Flash-Speichermedien

Anwendungen und Dateisysteme

25.11.2008

Christian Seyda
(seydanator@web.de)

Seminar Speichermedien WS0809

Universität Heidelberg

Betreuung: Olga Mordvinova, Julian M. Kunkel

Inhalt

1. Flash-basierter Speicher

- Wiederholung
- Eigenschaften
- Arten von Flash-Speicher:
  - USB-Sticks/Speicherkarten
  - SSD

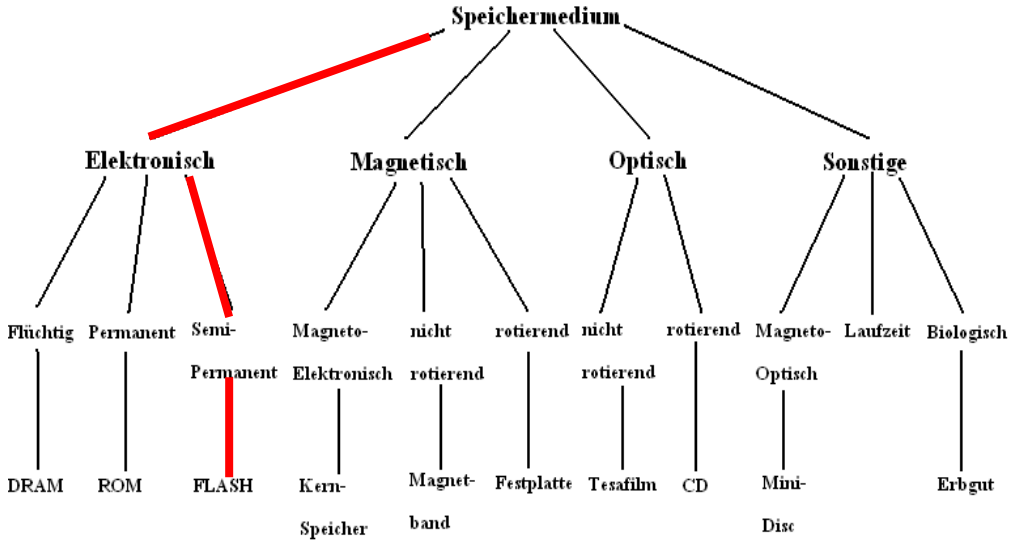
2. Dateisysteme für Flash

- JFFS(2)
- UBIFS

3. Ausblick in die Zukunft

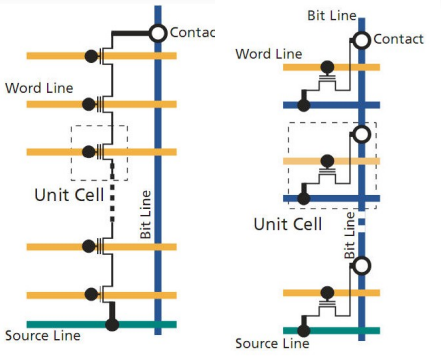
Flash-basierte Speichermedien

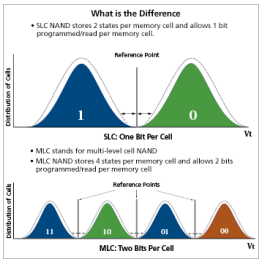
Einordnung Taxonomie



Wiederholung

- *Flasharten*
  - NAND
  - NOR
  - SLC/MLC
- *Endurance*
- *Wear-Levelling*
  - Statisch
  - Dynamisch





Endurance: Maximale Anzahl der Schreibzyklen pro Speicherzelle.

Wear-Levelling: Gleichmäßige Verteilung von Daten, so dass alle Blöcke gleichmäßig belastet werden.
Details im Vortrag Flash 1.

Wiederholung

	SLC NAND Flash	MLC NAND Flash		MLC NOR Flash	
Density	512Mbit to 4Gbit	1Gbit to 16Gbit	+	16Mbit to 1Gbit	-
Read Speed	24MB/s	18.6MB/s	-	103MB/s	+
Write Speed	8 MB/s	2.4 MB/s	+	0.47 MB/s	-
Erase Time	2.0 ms	2.0 ms	+	900 ms	-
Costs	cheap	Very cheap	+	expensive	-
Endurance	1 Mio	100.00 – 1 Mio	+	10.000 - 100.000	-

NAND für Massenspeicher
MLC-NAND für den Consumer-Markt
SLC-NAND für Firmen

NOR hauptsächlich für XiP-Anwendungen.

Allgemeine Eigenschaften von Flash-basierten Speichermedien

- Erschütterungsresistent
- Vernachlässigbare Zugriffszeiten (< 1ms)
- Geringer Stromverbrauch
- Lautlos
- Klein
- Geringe Wärmeentwicklung
- Magnetisch-resistent

Besonders im Unterschied zu Festplatten.

USB-Sticks/Speicherkarten

- Basieren auf NAND
- Bis zu 30MB/s lesend
- Bis zu 20MB/s schreibend
- Bis zu 64GB
- Kosten: 1-10€/GB
- Mobile Datenspeicherung
- Hat Diskette verdrängt

Hauptnachteil der Diskette: geringe Speicherplatz. Aber auch die anderen Nachteile, wie die physische Größe, die langsamen Transferraten, Datensicherheit,... verhalfen den Flash-Speichern im Bereich mobiler Datenspeicherung die Marktdominanz.

Lebenserwartung USB-Stick

- Testumgebung: 1GB Sony Microvault
- Stick bis auf einen Block vollschreiben
  → diesen Block immer wieder neu schreiben

- Ergebnis: ~90,5 Millionen Schreibvorgänge bis zum „Tod“
- Daten blieben nach „Tod“ noch lesbar

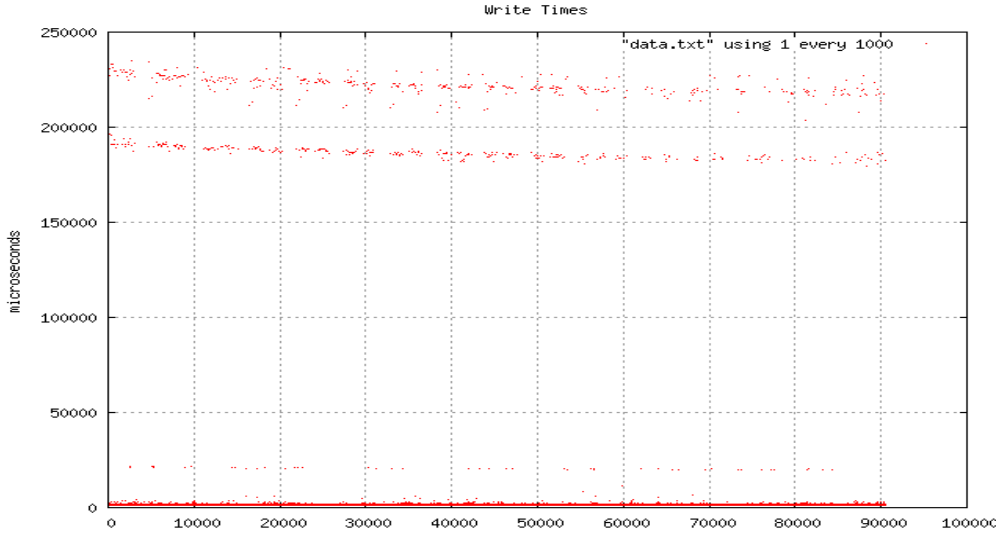
(http://www.bress.net/blog/archives/114-How-Long-Does-a-Flash-Drive-Last.html)

Mit „Tod“ ist hier gemeint, dass keine Daten mehr geschrieben werden konnten.

Zeiten für das Schreiben wurden protokolliert.

Lebenserwartung USB-Stick

(http://www.bress.net/blog/archives/114-How-Long-Does-a-Flash-Drive-Last.html)



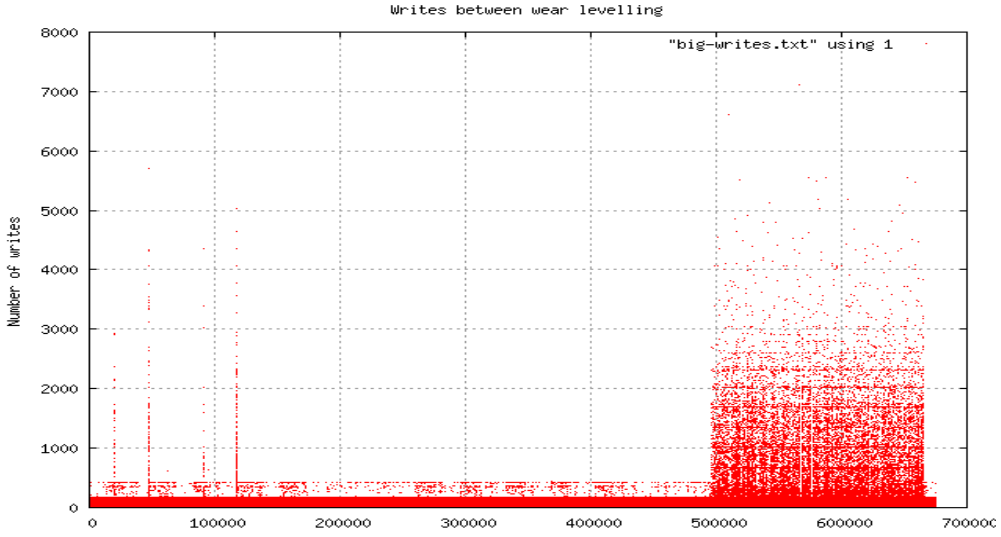
Zeit für jeden 1000. Schreibvorgang.

Beachte die längeren, „zufällig“ verteilten Schreibvorgänge.

Annahme: Stick „betreibt“ hier Wear-Levelling, kopiert Daten um

Lebenserwartung USB-Stick

(http://www.bress.net/blog/archives/114-How-Long-Does-a-Flash-Drive-Last.html)



Anzahl der Schreibvorgänge bis ein „langer“ Schreibvorgang passiert.

Ca. alle 50.000 Schreibvorgänge dauert ein Schreibvorgang länger.
Wahrscheinlich der Eraseblock.Counter zu hoch, so dass Daten verschieben werden.

SSD

- Solid State Drive
  - DRAM
  - Flashspeicher
- Erste SSD 1995 von M-Systems
- Seit 2006 auch für kommerzielle Zwecke
- Gedacht als Festplattenersatz
- Bisher nicht mal Ankündigungen von Festplattenhersteller
- Aber Modelle von Halbleiterherstellern

DRAM: Verbreitet bei Firmen, Datenbanken, sehr teuer. Näheres dazu in eigenem Vortrag.

Vorteile einer SSD auf Folie 7, Nachteile im Vergleich zu HDD sind der hohe Preis und die aktuell recht niedrigen Kapazitäten.

Intels SSD

- Zitat Linus Torwalds:

  „That thing absolutely rocks.[...]

  And the sad part is that other SSD's generally absolutely suck when it comes to especially random write performance.[...]

  So here's the deal: right now, don't buy any other SSD than the Intel ones“

- Daten: 2,5 Zoll / SATA2 / 80GB / ~600$

  Lesen: 250MB/s
  Schreiben: 70MB/s

Modell: Intel X25-M

Probleme der „billig“ SSDs

- Sehr lange Zugriffszeiten bei zufälligen Schreibvorgängen

100% random writes, IO queue depth 1	4 KB	16 KB	32 KB	64 KB	128 KB
OCZ Core (JMicron, MLC)	244ms	243ms	241ms	243ms	247ms
OCZ (Samsung, SLC)	9ms	14ms	21ms	28ms	29ms
Intel X25-M (Intel, MLC)	0.09ms	0.23ms	0.44ms	0.84ms	1.73ms
Seagate Momentus 7200.2	9ms	9ms	9ms	10ms	12ms

4KB writes, IO queue depth 1	100% S 0% R	90% S 10% R	50% S 50% R	0% S 100% R
OCZ Core (JMicron, MLC)	0.4ms	26ms	130ms	244ms
OCZ (Samsung, SLC)	0.16ms	2ms	5ms	9ms
Intel X25-M (Intel, MLC)	0.09ms	0.09ms	0.09ms	0.09ms
Seagate Momentus 7200.2	0.16ms	1ms	4ms	10ms

(S=Sequential, R=Random)

OCZ Core: billig Serie
OCZ: Server

Alle Angaben sind Mittelwerte:
Bei der Core gab es stellenweise Latenzen von 1000ms.

Zugriffszeiten führten unter anderem zum Absturz der Windows Vista Installation.

Ursachen

- „billig“ SSD: 3-4€/GB
- Ursachen können nicht direkt ausgemacht werden, jedoch:
- Beruhen auf „Standard“-Design (JMicron-Chip)

  → Viele Firmen baugleiche SSD

  → Massenproduktion drückt Preis
- Intel:

  → Eigener Controller

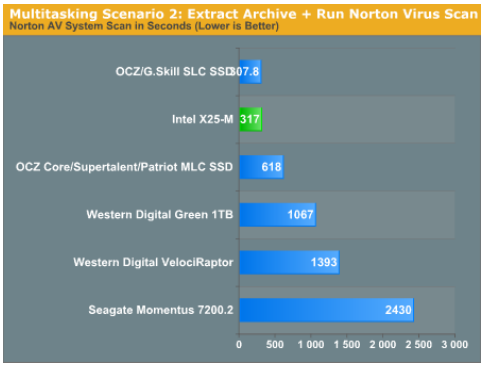
  → Erfahrungen mit Caches

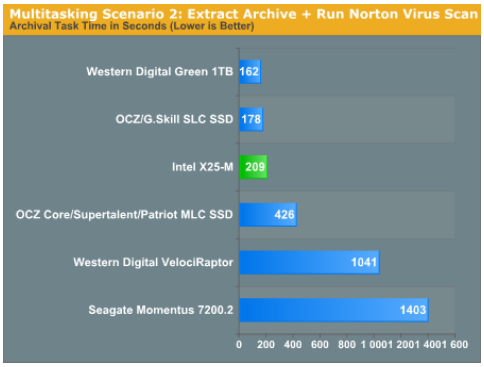
Controller von JMicron, möglicherweise Ursache

Die „guten“ SSD haben Eigenentwicklungen, zB eigene Flash-Controller; sind aber dementsprechend teurer.

Cache-Erfahrung: Zuweisungen zu Blöcken, Speichercontroller.

Multitasking





	Intel X-25M	OCZ SLC	WD Velociraptor	WD Green
AV Scan	317s	108s	1393s	1067s
Extracting	209s	178s	1041s	162s
Total	526s / ~9.5min	286s / ~4.75min	2434s / ~40.5min	1229s / 20.5 min

Multitasking in diesem Fall, gleichzeitige Zugriffe von verschiedenen Anwendungen auf verschiedene Daten.

Hier: Entpacken eines Archivs (Schreiben) und Viren Scannen (Lesen)

Werte der WD-Platten: Entweder vertauscht, oder Pfusch beim Test.

Lebensdauer SSD

- Bisher noch keine Studien
- OEMs wollen:

  20GB Daten pro Tag beschreiben

  5 Jahre lang
- Intel garantiert:

  100GB Daten pro Tag beschreiben

  5 Jahre lang

  (obwohl nur eine 3-Jahres-Garantie)

Zumindest fand ich noch keine Studien zu diesem Thema.

OEM: Original Equipment Manufacturer. Hersteller von Fertiggeräten (Fertigcomputer = Aldi, Dell,...)

Zusammenfassung Speichermedien

	USB-Stick	Speicherkarten	SSD
Write Speed	Up to 20MB/s	Up to 30MB/s	Up to 80MB/s
Read Speed	Up to 30MB/s	Up to 30MB/s	Up to 250MB/s
Price	1-5€/GB	2-10€/GB	5-10€/GB

- Flashspeicher in fast allen denkbaren Gebieten
- Preise fallen weiter
- Speichersticks verdrängten Disketten
- Speicherkarten hauptsächlich für Kameras
- Hersteller haben Endurance im Griff
- SSD auf dem Vormarsch

Embedded Devices nicht betrachtet, weil
ClosedSource → nicht genug dazu gefunden

SSD:
Leseperformance fast doppelt so schnell wie HDD
Zugriffe mindestens 10x so schnell
Preise noch astronomisch

Dateisysteme

Dateisysteme

Definition:

„Das Dateisystem ist Bestandteil des Betriebssystems und bildet die Schnittstelle zwischen diesem und den Laufwerken.
Es legt fest, wie der Computer Dateien auf den Datenträgern benennt, speichert, organisiert und verwaltet.“
(http://www.itwissen.info/definition/lexikon/Dateisystem-file-system.html)

Vereinfachung:

- Nicht jedes Flash-Dateisystem für NAND und NOR
- Aber: gleiche Prinzipien, nur andere Implementierung

NAND (128 KiB) hat größere eraseblocks als NOR (8 KiB)
NAND blocks weiter unterteilt:
Pages (512 bytes + 16 bytes „out of band“)

Bis Block neu gelöscht wird:
NOR: Erneutes Schreiben bis block voll
NAND: nur ~10x Schreiben in Page
Out of band: Speicher gedacht für Metadaten und Fehlercodes.

Wir betrachten nur ganze Eraseblocks, keine Pages

keine Unterscheidung NAND / NOR

Block vs. Flash Device

Block Device	Flash Device
Consists of **sectors**	Consists of **eraseblocks**
Sectors are small (512, 1024 bytes)	Eraseblocks are larger (128 KiB / 8 KiB)
2 main operations: **read sector** and **write sector**	3 main operations: **read from eraseblock**, **write to eraseblock** and **erase eraseblock**
Bad sectors are **re-mapped** and **hidden** by hardware	**Bad eraseblocks** are **not hidden** and should be dealt within software
Sectors are **devoid of the wear-out** property	Eraseblocks **wear-out** and become **bad and unusable** after about 10^3 – 10^5 erase cycles

(http://www.linux-mtd.infradead.org/faq/general.html)

➡ Flash Devices sind schwerer handhabbar

Block Devices sind zum Beispiel Festplatten, Disketten, CDs, DVDs, FTLs,...

MTD Devices sind Flashspeicher ohne FTL

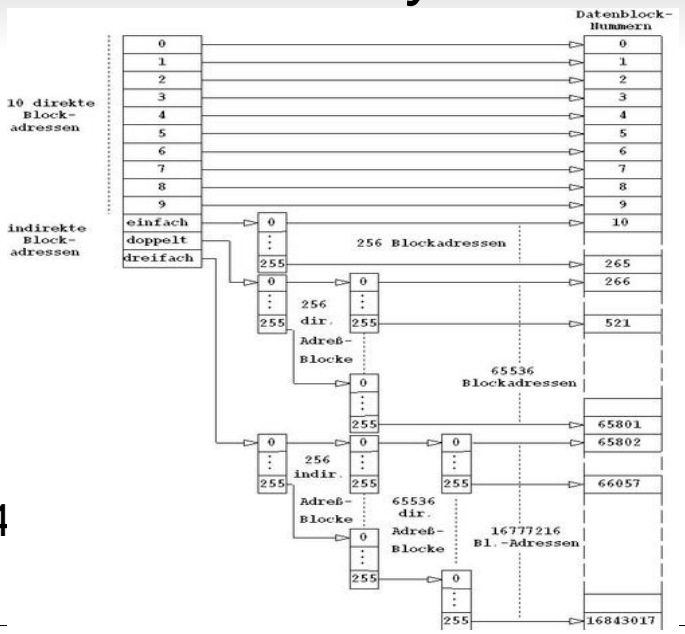
Write Eraseblock immer nur nach Erase Eraseblock

„Klassisches“ Blockdateisystem

Besteht aus:

(- Boot-Block)
- Super-Block
- Inode-Liste
- Datenblöcke

Bsp: Unix, ext2/3/4

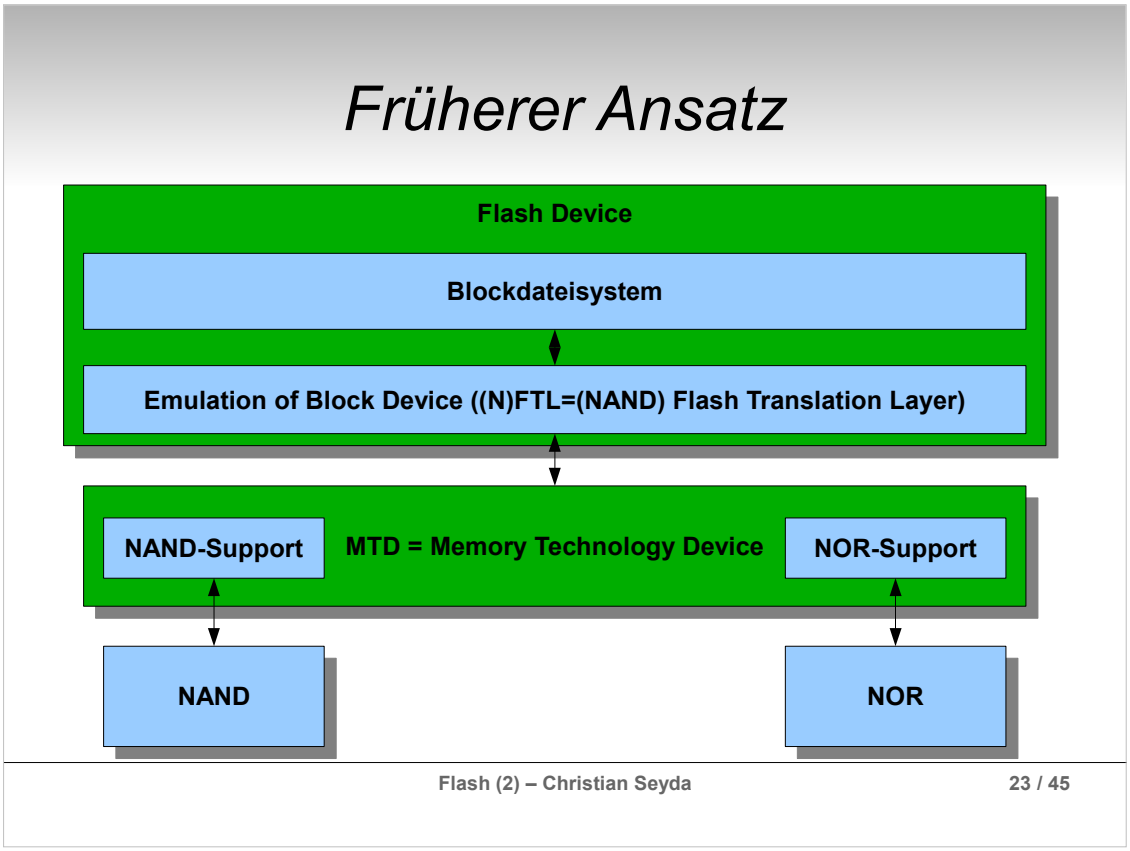


Boot-Block (-Sektor, MBR): Herstellername und Versionsnummer, Anzahl der Byte pro Sektor, Sektoren pro Cluster...

Super-Block: Größe des Dateisystems, Anzahl freier Blöcke, Liste der freien Blöcke, „Sperr-Felder“ für Liste der freien Blöcke / Inodes (z.B. für defekte Blöcke)

Inodes: Besitzer der Datei, Zugriffsrechte der Datei, Typ der Datei (einfache Datei, Verzeichnis, Link,…), Größe der Datei (in Bytes)
=> nennt man auch Metadaten

Kann nicht direkt für Flash benutzt werden, weil Schreiben zuerst ein Löschen erfordert, Änderungen immer an selben Stelle geändert werden (in-place updates) => wear-out



Emulation früher Software, heutzutage in Firmware.

Am Einfachsten: 1zu1 Mapping des Block Devices auf das Flash Device. Folglich kein Wear-Levelling und Datenkonsistenz bei zB. Stromausfall.

Verbreiteste und fortschrittlichste Emulationsschicht: (N)FTL: (NAND) Flash Translation Layer

Warum nicht einfach FTL benutzen?

1. Flash Translation Layer ist nicht OpenSource

2. Nur rudimentäre Wear-Levelling Ansätze

3. Zitat David Woodhouse (RedHat):

> „This sucks. Obviously you need a journalling file system on your emulated block device, which is itself a kind of journalling pseudo filesystem. Two layers of journalling on top of each other aren't the best way to ensure efficient operation."

1. Linux unterstützt zwar FTL, aber der Gebrauch davon ist patentrechtlich missbilligt

3. Eigentlich nur bei Embedded Devices genutzt zur Kostenersparnis. Folglich ist das Wear-Levelling nur für kleine Speichergrößen konzipiert.

JFFS

- Journaling Flash File System
- OpenSource
- Basiert auf:
  - Journaling
  - Sequenziellen Logs
- Garbage Collection

Journalling Flash File System

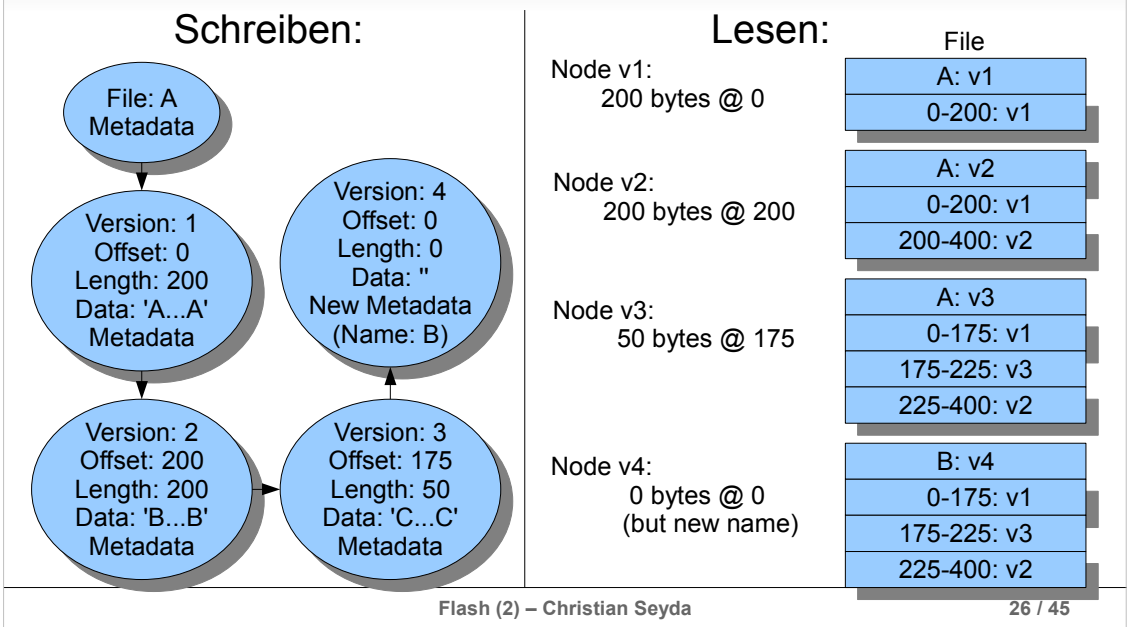
Journalling: Um Dateikonsistenz auch bei Stromausfall oder anderen Unterbrechungen zu gewährleisten, werden Änderungen zuerst in ein „Journal“ geschrieben, bevor sie ausgeführt werden. Das erspart langes Überprüfen.

Log-basiert: Um hohe Schreibgeschwindigkeiten zu erreichen, werden alle Updates bzgl. Dateien und Metadaten in einen sequenziellen Stream, genannt Log, geschrieben.

1991 von Axis Communications AB unter GNU General Public License

Erstes Dateisystem, dass sich dieser Problematik annahm.

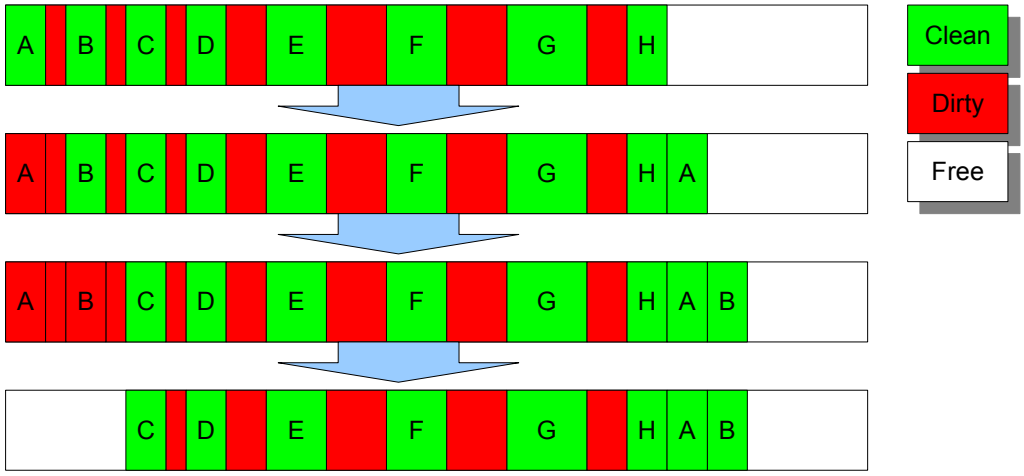
Schreiben und Lesen



Nodes werden sequenziell geschrieben. Für jede Änderung neuer Node.

Beim Lesen wird Stück für Stück der aktuelle Zustand hergestellt.

Garbage Collection



=> statisches Wear-Levelling

Garbage Collecting: sobald zuwenig Speicher zum Schreiben frei ist, werden solange gültige Daten ans Ende geschrieben, bis ein Eraseblock gelöscht werden kann.

Clean: Gültige Daten.
Dirty: Durch spätere Schreibvorgänge ungültig gewordene Daten.

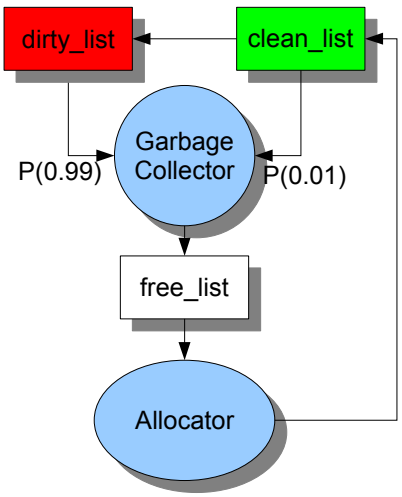
Begrenzungen von JFFS

- Keine Kompression
- Metadaten in jedem Knoten gespeichert:
  - Verhindert Hard-Links
  - Speicherplatzverbrauch
- Garbge Collection:
  - Linear
  - Nicht effizient bei wenig freiem Speicherplatz

Metadaten: zb Name der Datei, Vaterknoten,...

Verbesserungen in JFFS2

- Einführung von Kompression
- Verschiedene Knotenarten
  - Ermöglicht Hard-Links
  - Effizientere Metadatenhaltung
- Beachtet Eraseblockgröße
  - Logs nicht mehr sequenziell
- Verschiedene Kontrolllisten
  - free_list, clean_list, dirty_list,...
  - Effizientere Garbage Collection



Weiterentwickelt von RedHat, weil ein Kunde Kompression wollte. Da das ohne weiteres nicht ging, hat man das FS neu geschrieben und sich den Kritikpunkten angenommen.

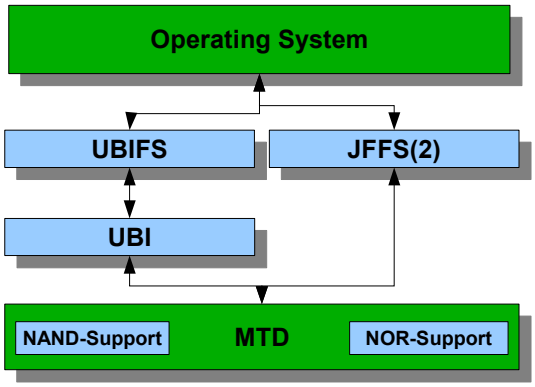
Knotenarten: Metadaten werden nicht mehr komplett in jeden Node geschrieben, Knotenart für Verzeichnisse ermöglicht Hard-Links

Kontrolllisten: freie Eraseblocks, ungültige Eraseblocks, gültige Eraseblocks,...

Garbage Collection muss nicht mehr sequenziell durch das Medium suchen.
Garbage Collection löscht zu 99% blocks aus der dirty_list, zu 1% aus der clean_list
=> statisches wear-levelling

Verbesserungen in UBIFS

- Index auf Flash
→ Schnelleres Mounten
→ Weniger Speicherverbrauch
→ Besser skalierbar
→ Komplizierter handzuhaben
- Unterstützt write-back
- Läuft auf UBI Volumes



University of Szeged, Nokia Mitarbeitern
Um JFFS2 zu verbessern

JFFS2 rekonstruiert den Index bei jedem mounten neu und hält ihn dann im Speicher
→ lineare Mountingzeit und Speicherverbrauch im Verhältnis zur Größe des Flashspeichers

Write-back: Daten werden so lange nicht geschrieben, bis es unbedingt nötig wird.
→ höhere Schreibgeschwindigkeit. Ähnlich dem Festplatten-Cache.

UBI

- Unsorted Block Images

MTD Partition	UBI Volume
Physical Eraseblocks (PEB)	Logical Eraseblocks (LEB)
No wear-levelling	Static wear-levelling across whole medium
No Bad Block Managment	Has Bad Block Managment

→ einfacher handhabbar als MTD

→ einfachere Dateisysteme

- Dynamische Änderung von UBI-Volumes
- UBI ist kein FTL!

Schicht zwischen MTD und FS

Bietet Dateisystemen logische UBI-Volumes (statisch: nur lesen / dynamisch: lesen und schreiben)

Dateisystem muss sich nicht mehr um Wear-Levelling, Bad Block Management kümmern

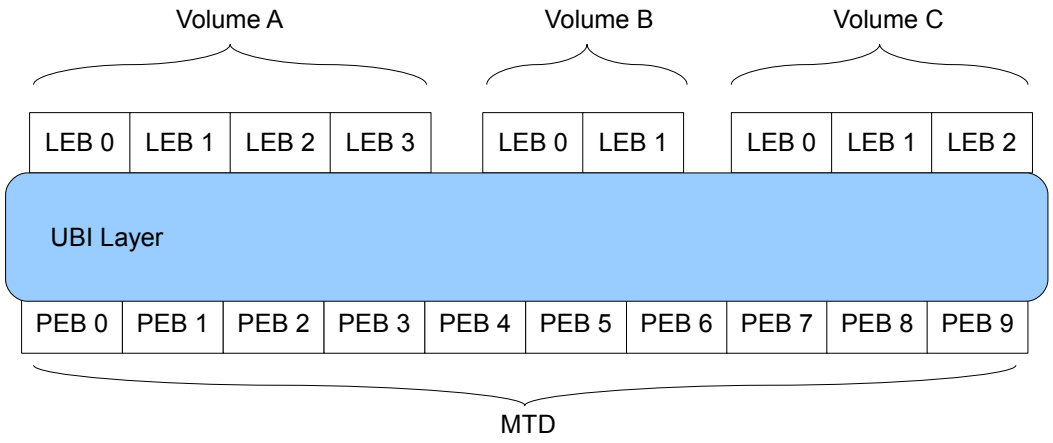
Änderung: Erzeugung, Löschem, Größenänderung, on the fly

FTL emuliert Blockdevice, UBI nicht! In UBI immer noch Eraseblocks

FTL kann aber auf UBI-Layer laufen

UBI

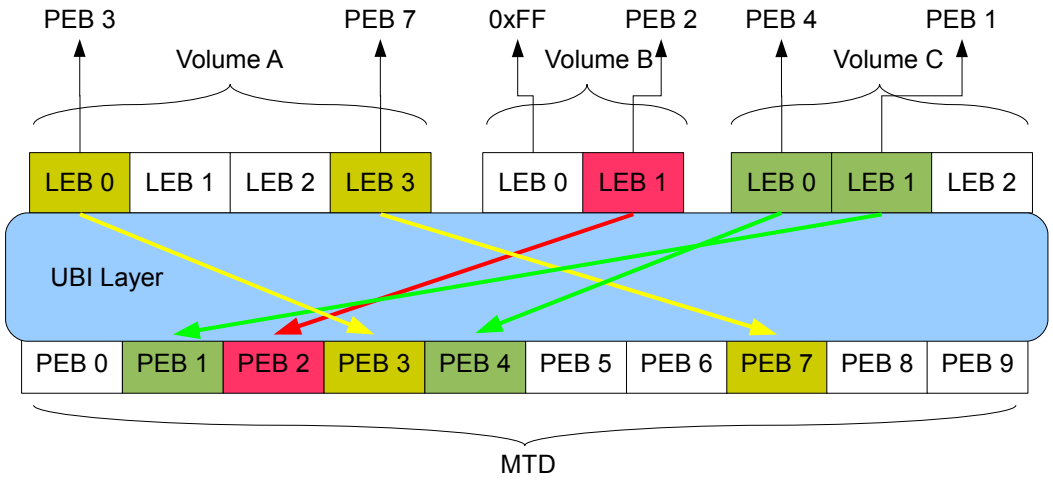
Schreiben:



UBI-Volumes werden erzeugt

UBI

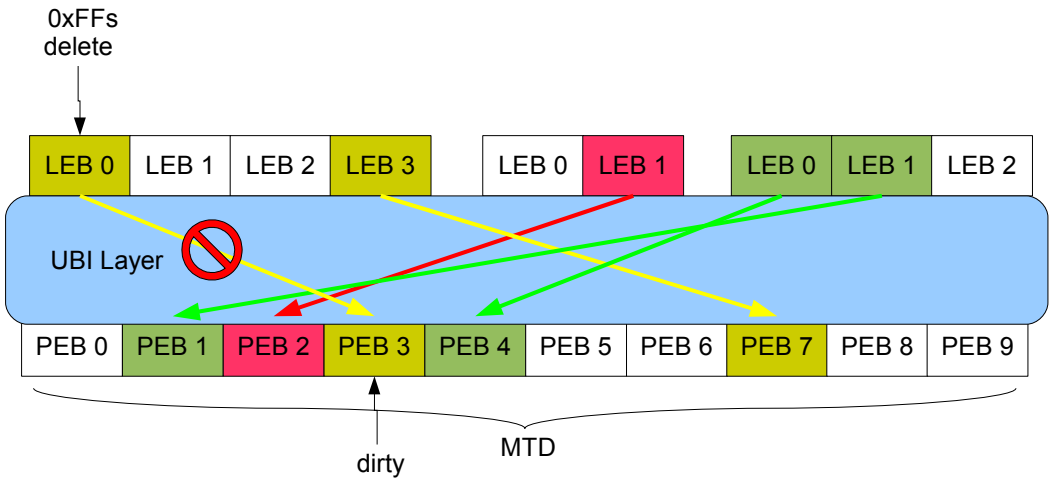
Schreiben: dynamisches Wear-Levelling



LEB werden auf PEB gemappt.
Leder LEB kann auf jeden PEB gemappt werden.

UBI

Löschen:

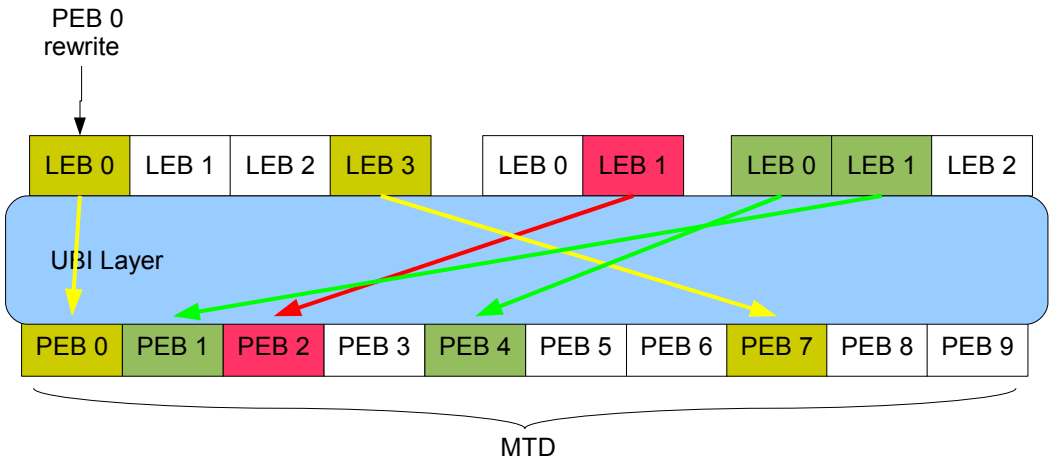


Daten aus LEB0 werden gelöscht: „Link“ zu PEB3 entfernt, PEB3 als dirty markieren

Garbage Collection löscht sofort EraseBlock in einem asynchron ausgeführten Thread.

UBI

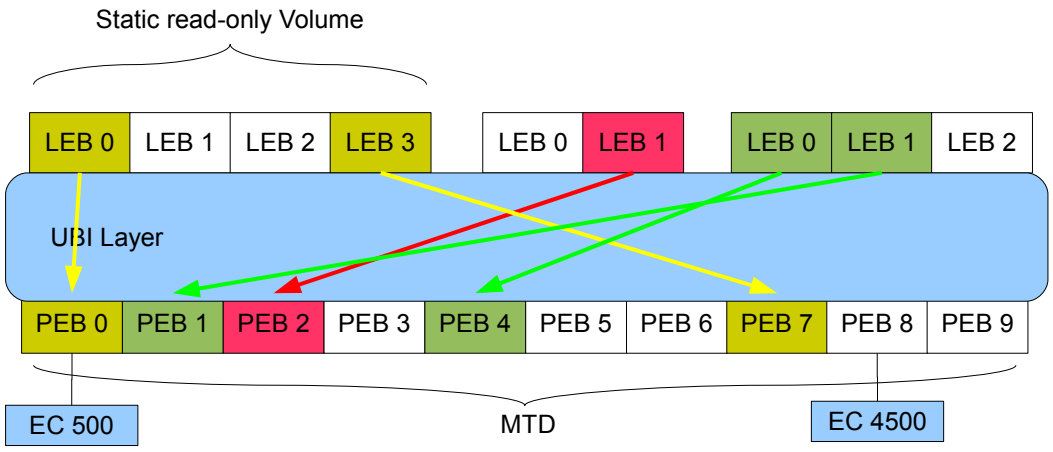
Löschen:



Neues Schreiben auf LEB 0.
LEB 0 wird diesmal auf PEB 0 gemappt.

UBI

Statisches Wear-Levelling:



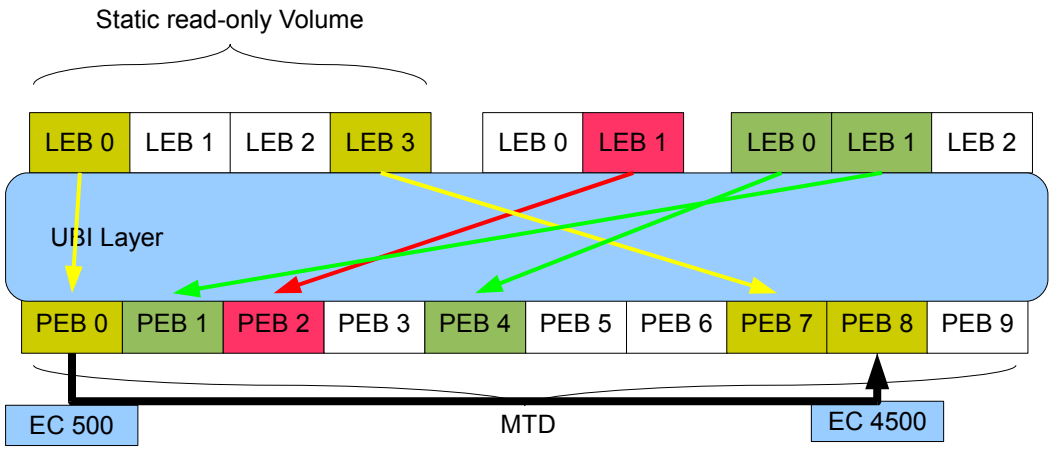
Statisches UBI-Volume: read only → static Wear-Levelling

Dynamisches UBI-Volume: read and write → dynamisches Wear-Levelling

Falls Grenzwert lowest – highest EraseCount überschritten wird

UBI

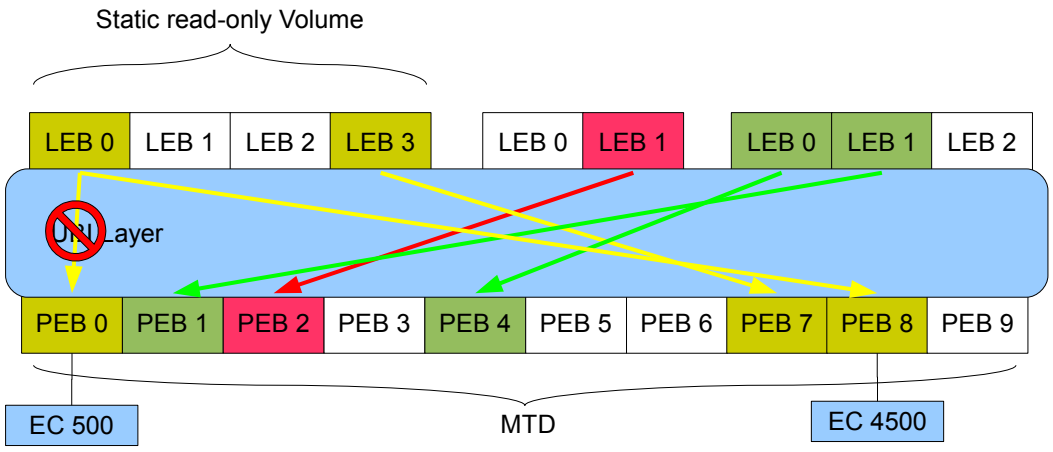
Statisches Wear-Levelling:



Falls Grenzwert lowest – highest EraseCount überschritten wird
→ kopiere Daten

UBI

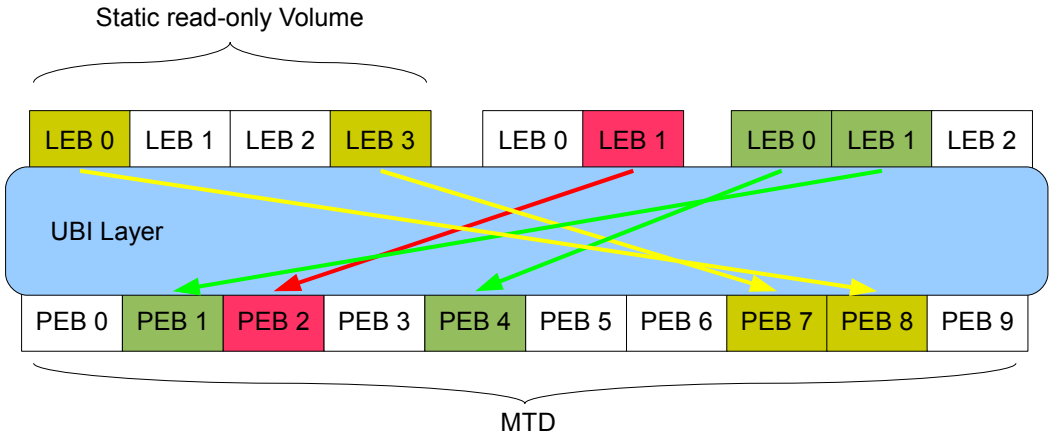
Statisches Wear-Levelling:



Falls Grenzwert lowest – highest EraseCount überschritten wird
→ kopiere Daten
→ verlinke neu

UBI

Statisches Wear-Levelling:



Falls Grenzwert lowest – highest EraseCount überschritten wird
→ kopiere Daten
→ verlinke neu

Zusammenfassung Dateisysteme

	Dateisysteme für Blockspeicher	Dateisysteme für Flashspeicher
Write / Change File	in-place	out-of-place differences to nodes
Devices	Block-Devices FTL-Devices	MTD (bare flash = mostly embedded devices)

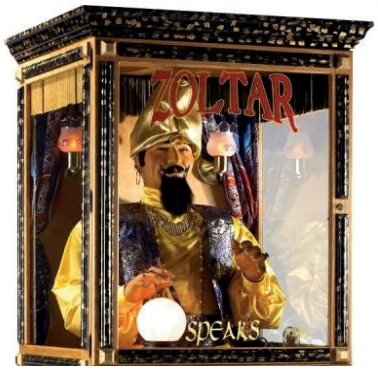
- Wear-Levelling
- Garbage Collection
- Bad Block Management

Weitere Dateisysteme für Flashspeicher:

- LogFS / YAFFS / CramFS / SquashFS

CramFS / SquashFS für ROM

Ausblick in die Zukunft



Zukunft

- Kleinere Strukturbreiten → mehr Speicherplatz
- SanDisk: ExtremeFFS

  „Flash-Performance“ um Faktor 100 steigern

  Cache auf SSD

  Controller lernt Gebrauchsmuster
- „Performance-Gewinn“ um Faktor 4 in 2009
- JEDEC: Standards für SSD in 2009

Strukturbreite: momentan 40nm

Moore's Law: alle 18 Monate Verdopplung der Transistoren / Halbierung der Preise

Entwicklung wohl schneller

JEDEC = Joint Electron Device Engineering Council.
Die JEDEC Solid State Technology Association (kurz JEDEC) ist eine US-amerikanische Organisation zur Standardisierung von Halbleitern.

Zukunft

- Intel will Moore's Law einhalten:

  ~ 200GB in einem Jahr

  ~ 500GB in drei Jahren

  oder: 80GB SSD für 100€ in drei Jahren
- Intels X25-E mit 32GB/64GB

  250MB/s lesend, 210MB/s schreibend
- Samsung SSD mit 256GB

  220MB/s lesend, 200MB/s schreibend

Gegenwart

- Flash-Speicher in fast allen Embedded-Devices
- Als Portabler Massenspeicher unverzichtbar
- Keine Endurance-Probleme mehr
- Gute SSD noch zu teuer und wenig Speicher
- Flash-Dateisysteme:

  Wear-Levelling, Bad Block Management, Garbage Collection

  bisher leider nur für embedded-Devices

Portabler Massenspeicher = wiederbeschreibbare CDs/DVDs sind langsamer, und man braucht einen Brenner. USB-Sticks brauchen nur USB-Port, haben sogar Nettops ^^

Titel durch Klicken hinzufügen

- Fragen?
- Danke für eure Aufmerksamkeit!

Quellen

Web:

http://www.linux-mtd.infradead.org

http://sourceware.org/jffs2/

http://www.ibm.com/developerworks/linux/library/l-flash-filesystems/?S_TACT=105AGX54&S_CMP=B0522&ca=dnw-920

http://torvalds-family.blogspot.com/2008/10/so-i-got-one-of-new-intel-ssds.html

Quellen

Web:

http://www.tomshardware.com/de/Festplatten-Loeschen-Neodym-Eisen-Bor-Magnet,testberichte-239962.html

http://www.bress.net/blog/archives/114-How-Long-Does-a-Flash-Drive-Last.html
(How Long Does A Flash Drive Last)

http://www.anandtech.com/cpuchipsets/intel/showdoc.aspx?i=3403&p=1
(Test der Intel X-25M)

Quellen

Web:

http://www.computerbase.de/news/hardware/laufwerke/massenspeicher/2008/november/jedec_standards_ssds_2009/

http://www.computerbase.de/news/hardware/laufwerke/flashspeicher/2008/november/erste_benchmarks_intels_x25-e_ssd/

http://www.computerbase.de/news/hardware/laufwerke/flashspeicher/2008/november/samsung_25-zoll-ssd_256_gb/

http://ht4u.net/news/2594_ueberarbeitetes_dateisystem_von_sandisk_soll_geschwindigkeit_von_flash-speicher_erhoehen/

Quellen

Papers/Folien:

http://rtg.informatik.tu-chemnitz.de/docs/vortraege/def-sa-mdae.pdf
(Echtzeitanalyse von Flash-Dateisystemen)

http://sourceware.org/jffs2/jffs2-slides-transformed.pdf
(The Journalling Flash File System)

http://www.datalight.com/products/download.php?type=public&resourceid=715
(A Comparison of Commercial and Open-Source Flash File Systems for Linux)

Quellen

Papers/Folien:

http://free-electrons.com/redirect/celf-ubi.html
(An examination of UBI)

http://klabs.org/richcontent/MAPLDCon98/Papers/c4_miyahira.doc
(Evaluation of Radiation Effects in Flash Memories)