Flashspeicher und Solid State Disks unter Linux

Steffen Bauer, Vortrag 25/10/2011 Linux User Group Frankfurt am Main

Teil 1

Grundlagen Flashspeicher

Historie: EPROM / EEPROM / Flash

EPROM

- *Frohman* (Intel 1971)
- Programmierung langsam (mehrere Minuten)
- Löschen über UV-Licht (~30 Minuten)

EEPROM

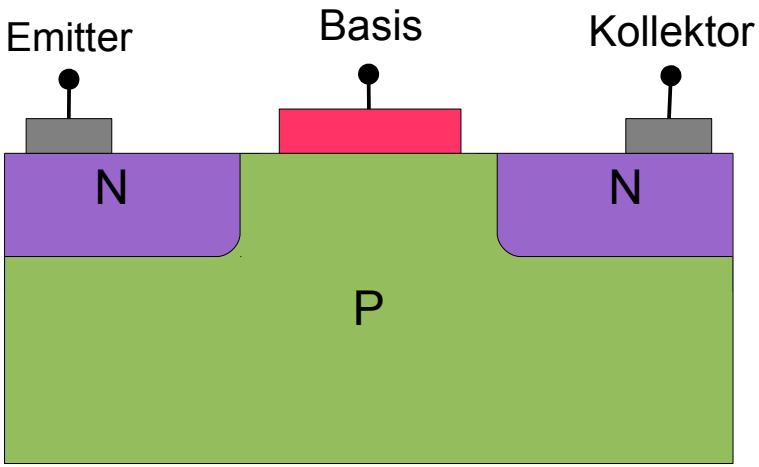
- *Perlegos* (Intel 1978)
- Programmierung ähnlich EPROM
- Löschen elektrisch (wenige Sekunden)

Flash

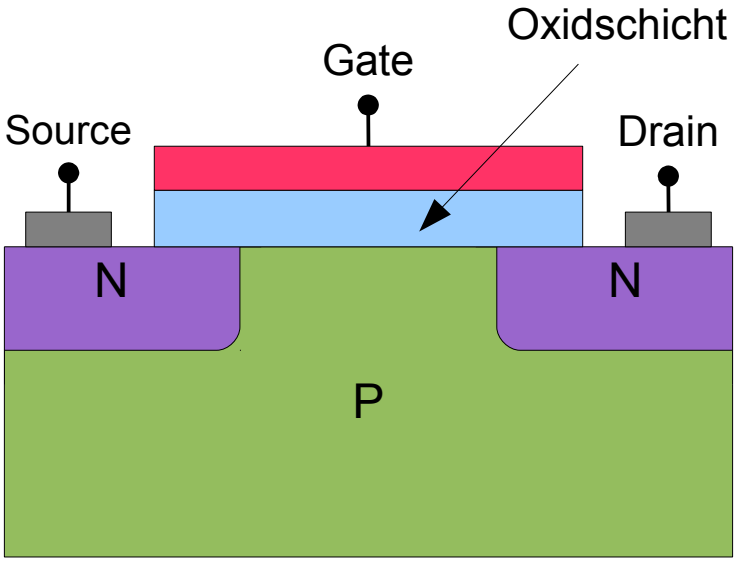
- *Masuoka* (Toshiba 1980)
- Aufteilung in Pages / Eraseblocks
- erheblich höhere Speicherdichte
- Löschen nur blockweise

Source: Wikicommons

Bipolartransistor vs. Feldeffekttransistor

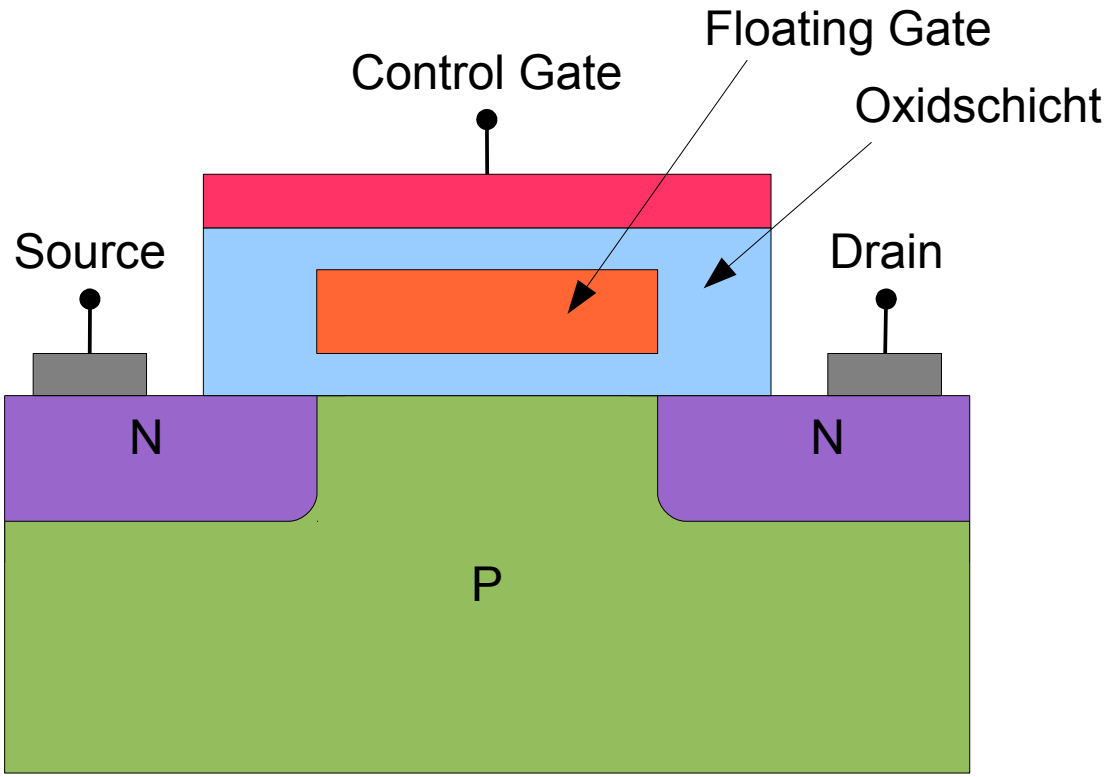


- *Shockley* (1947)
- **Strom**gesteuertes Schalten
- Allgemeine Analogtechnik
- Kleinsignalverstärker („Transistorradio“)

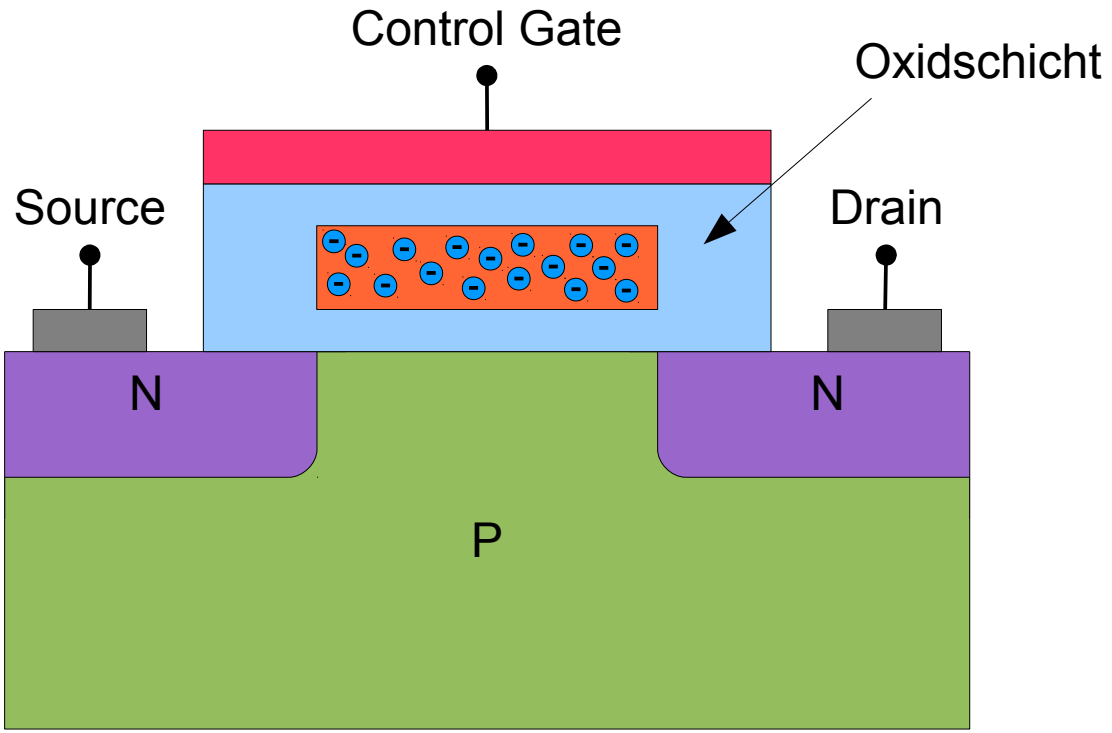


- *Lilienfeld* (1928)
- Ökonomisch erst ab den 60ern
- **Spannungs**gesteuertes Schalten
- Fast leistungsloses Schalten
- Digitaltechnik (CMOS)

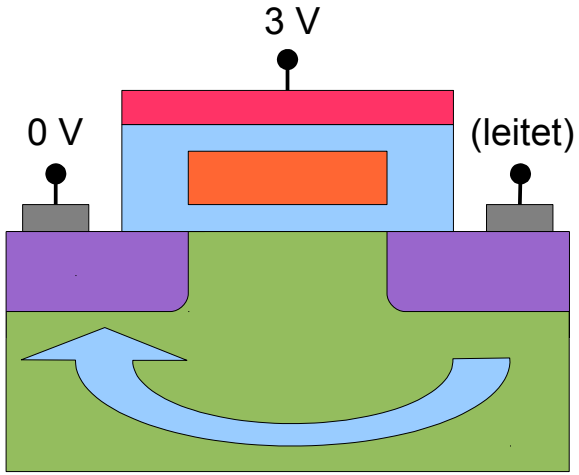
Floating Gate Transistor



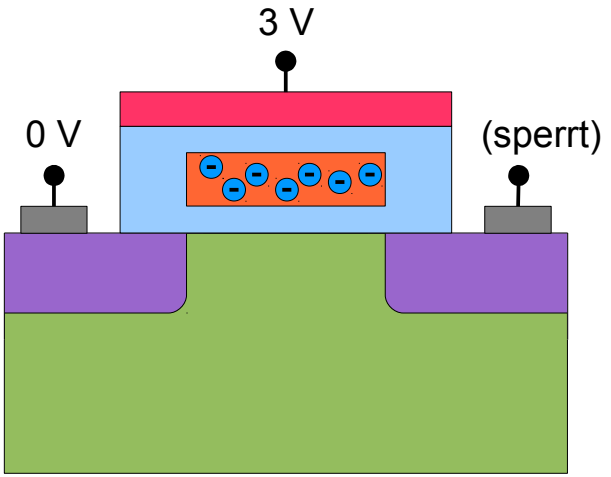
Floating Gate Transistor



Grundlegende Operationen: Lesen

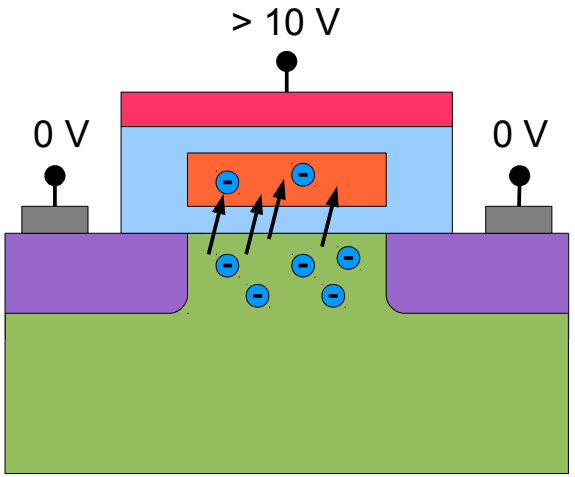


Auslesen ('1')

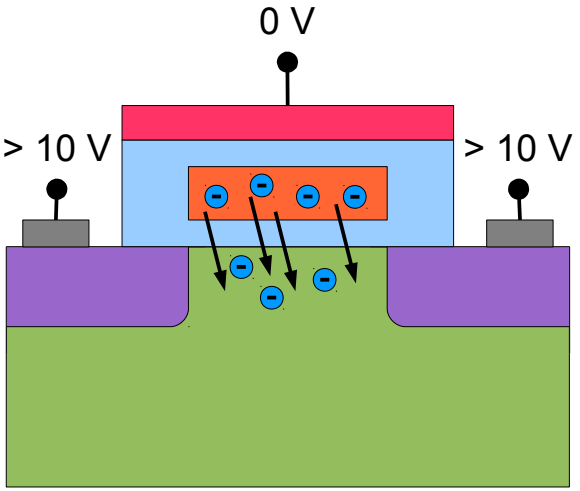


Auslesen ('0')

Grundlegende Operationen: Schreiben / Löschen

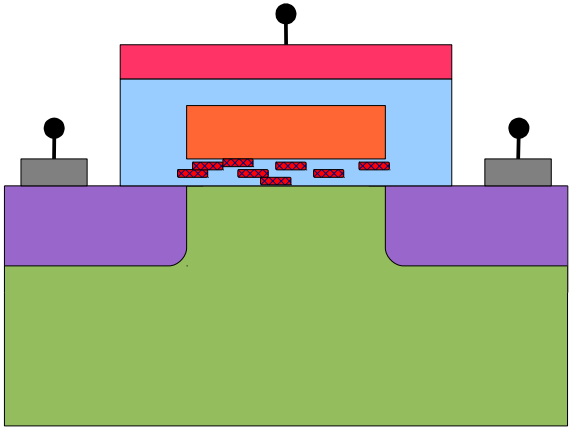


Schreiben ('1' -> '0')

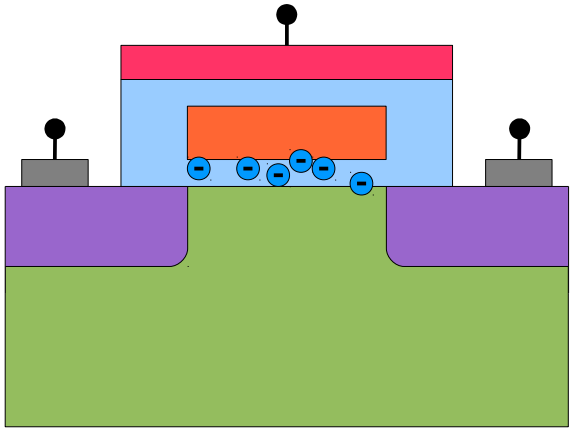


Löschen ('0' -> '1')

Alterung von Flashzellen

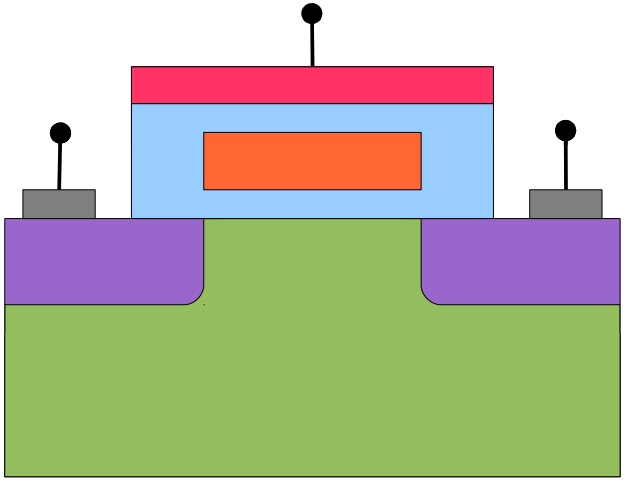


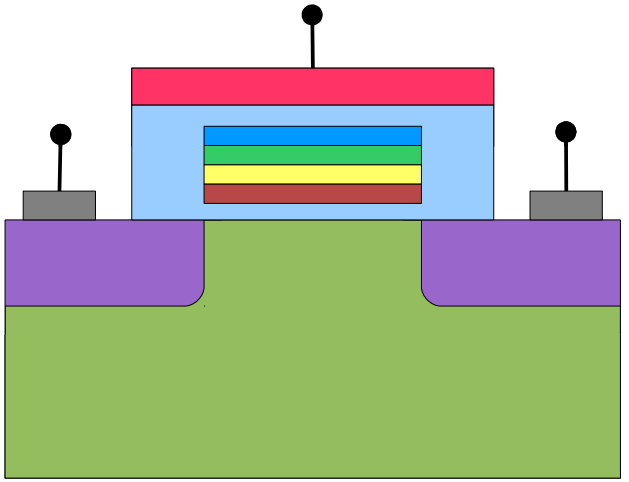
- Degration der Oxidschicht
- Floating Gate kann Elektronen nicht mehr halten
- Flashzelle ist dauerhaft auf '1'



- 'Electron trapping'
- Eingefangene Elektronen 'kleben' in der Oxidschicht
- Flashzelle ist dauerhaft auf '0'

Single Level Cell vs. Multi Level Cell (SLC / MLC)





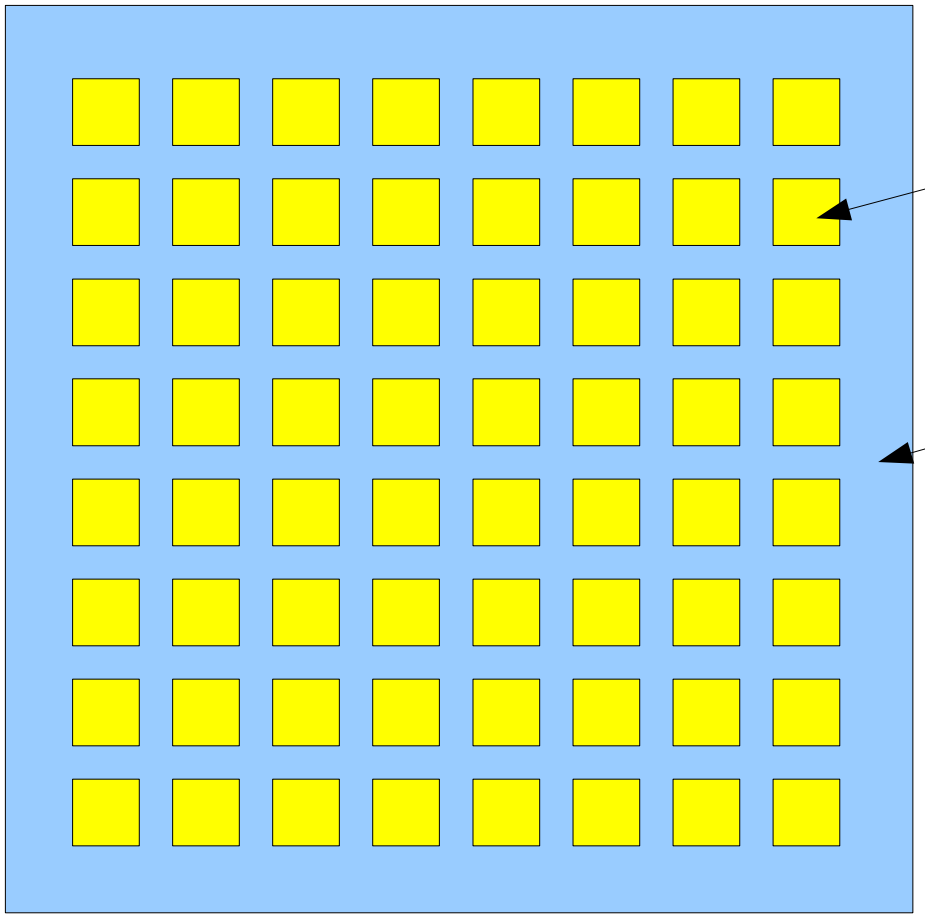
SLC

- Speichert 1 Bit / Zelle
- ~ 100000 Schreib/Löschzyklen
- Schneller als MLC
- Teurer als MLC
  - → High-End Einsatz
  - → Server-SSD

MLC

- Speichert 2-4 Bit / Zelle
- < 10000 Schreib/Löschzyklen
- Langsamer als SLC
- Höhere Speicherdichte als SLC
- Günstiger
  - → USB-Sticks
  - → Consumer-SSD
  - → Mobile Geräte

Eraseblock / Pages



Page (meist 4 kB):
Kleinste *schreibbare* Einheit

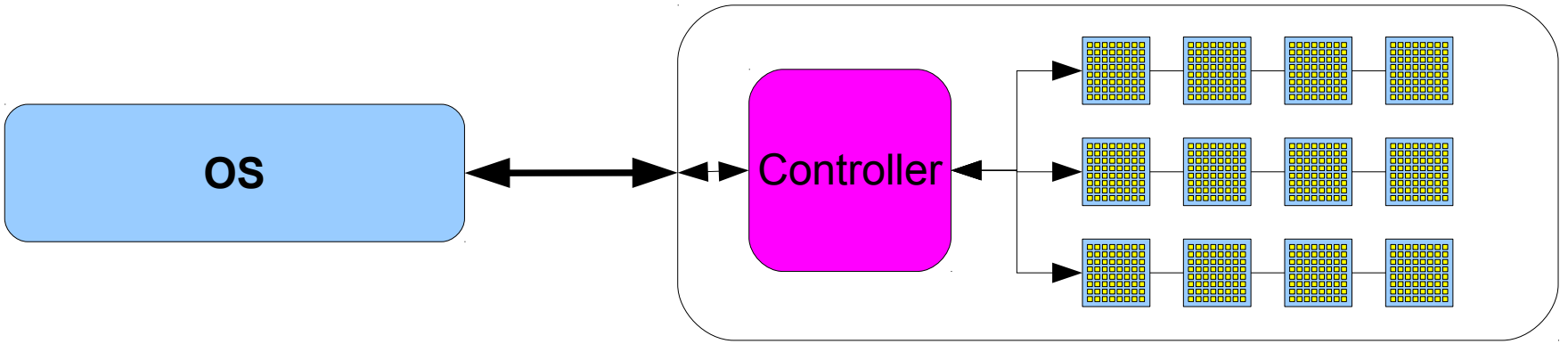
Block (meist 512 kB):
Kleinste *löschbare* Einheit

- Pages/Blockstruktur erlaubt hohe Speicherdichten von Flash (erheblich weniger Steuertransistoren)
- Herkunft des Namens „Flash“ (Löschvorgang erinnert an Kamerablitz)
- Braucht aber anspruchsvolle Algorithmen (Garbage Collection, Wear Leveling ...)

Flash Transaction Layer

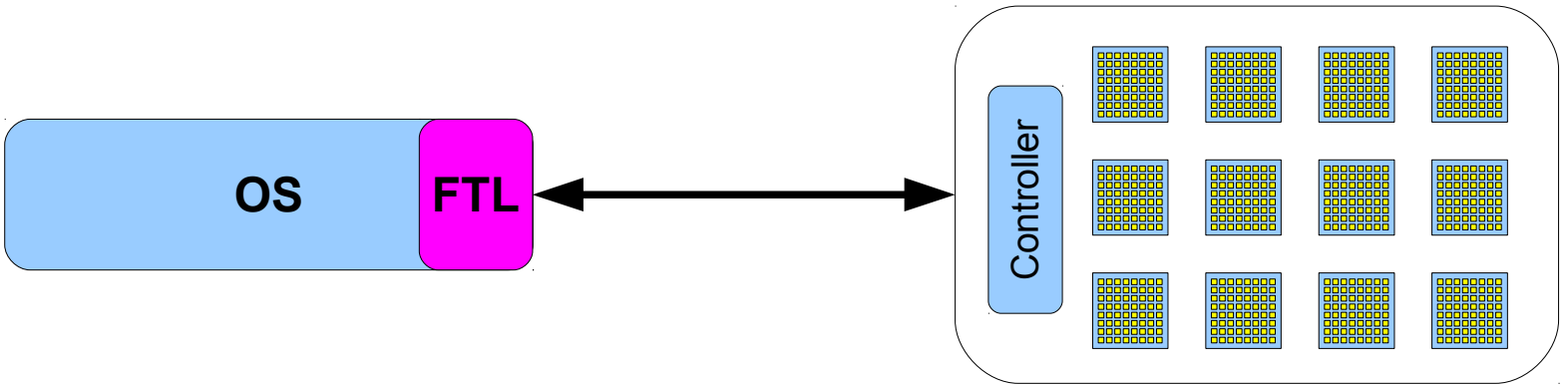
Flash als **Block Device**

(USB-Stick, Solid State Disk, Speicherkarte, ...)



Flash als **Memory Technology Device**

(Embedded system, interner Speicher Smartphones, ...)



Write Amplification

$$\text{write amplification} = \frac{\text{Daten real vom SSD-Controller geschrieben}}{\text{Daten vom Hostsystem geschrieben}}$$

Faktor	Beschreibung	Effekt auf WA
Over-provisioning	Unalloziierter Speicher für Controlleralgorithmen	Positiv
TRIM	Explizites Kennzeichnen von gelöschten Pages	Positiv
Secure erase	Komplettes Zurücksetzen gebrauchter SSDs	Positiv
Sequential writes	Wenige große Schreibzugriffe	Neutral
Random writes	Viele kleine Schreibzugriffe	Negativ

Teil 2

SSD als Systemplatte für Laptops unter Linux

SSD vs. Harddisk als Systemplatte

	Solid State Disk	Hard Disk
Pros	• Erschütterungsfest • Geringer Stromverbrauch • Zugriffszeit Lesen • Lautlos	• Preis (< 10ct / GB) • Beliebig oft beschreibbar
Cons	• Preis (> 1€ pro GB) • Begrenzte Schreibzyklen	• Geräuschentwicklung • Erschütterungsempfindlich • Höherer Stromverbrauch • Positionierungszeit

Checkliste SSD als Systemplatte unter Linux

- Gebrauchte SSD mit Secure Erase löschen
- Partitionierung
  - ➔ Sektoralignment beachten
  - ➔ evtl. Spare Area einrichten
- Filesystem wählen und spezifische Mountoptionen setzen
- I/O Scheduler wählen
- Volatile temporäre Verzeichnisse in *tmpfs* mounten
- Virtual Memory Subsystem anpassen
- Writeback Cache aktivieren
- I/O Vorgänge überwachen
  - ➔ Prozesse mit hoher I/O optimieren / in Ramdisk auslagern

hdparm / smartmontools

Read timings bestimmen:

```
> hdparm -t /dev/sdX        # Reading speed directly from device
> hdparm -T /dev/sdX        # Reading speed through cache
```

Disk features bestimmen:

```
> hdparm -I /dev/sdX        # Identification directly from device
> hdparm -i /dev/sdX        # Device identification from kernel
```

SMART-Werte auslesen:

```
> smartctl -a /dev/sdX      # Extended S.M.A.R.T. information
```

Secure Erase: Komplette SSD löschen

(genaue Prozedur siehe z.B. https://wiki.archlinux.org/index.php/SSD_Memory_Cell_Clearing)

```
> hdparm --user-master u --security-set-pass <password> /dev/sdX
> hdparm --user-master u --security-erase <password> /dev/sdX
```

Partitionierung / Alignment

a) **Spare Area** einrichten / vergrößern (Over-Provisioning)

- Zusätzlicher Platz für Wear Levelling usw.
- Erhöht Lebensdauer und I/O Performance
- Bis zu max. 27% Spare Area der Gesamtkapazität sind sinnvoll

1) Bei Partitionierung unpartitionierten Bereich frei lassen
2) Oder mit **hdparm -N** eine Host Protected Area einrichten

b) **Alignment** beachten

- Sektoren sollten nicht die Grenze zweier Pages überschreiten
- Heutige Linux Installer richten meist Sektoren an 1024 kB-Vielfachen aus
- Bei manueller Partitionierung siehe folgende Anleitungen:

  https://wiki.archlinux.org/index.php/Solid_State_Drives#Partition_Alignment
  http://www.thomas-krenn.com/de/wiki/Partition_Alignment
  http://wiki.ubuntuusers.de/SSD/Alignment

Filesysteme für SSDs

EXT2 / EXT3 / EXT4

Mountoptionen:

• *noatime*	alternativ *relatime*	EXT2 / EXT3 / EXT4
• *journal=writeback*	mit *tune2fs* aktivieren	EXT3 / EXT4
• *discard*	aktiviert TRIM	EXT4 (ab 2.6.33)

BTRFS (noch experimentell)

Mountoptionen:

• *noatime*	
• *discard*	
• *ssd*	für High-End SSDs
• *ssd_spread*	für ältere SSDs

JFFS2, YAFFS, UBIFS, LogFS

Nur für MTD! Nicht mit Block Devices verwendbar

Volatile Verzeichnisse in Ramdisk mounten

Voraussetzung: Kernel mit TMPFS Unterstützung, genügend physisches RAM

- /tmp
- /var/run
- /var/lock

Falls keine Log-Files benötigt werden (Laptop):

- /var/log
  - ➔ Unterverzeichnisse müssen manuell angelegt werden

```
# /etc/fstab

tmpfs                /tmp      tmpfs     defaults,relatime,mode=1777            0         0
tmpfs                /var/run  tmpfs     defaults,relatime,mode=1777            0         0
tmpfs                /var/lock tmpfs     defaults,relatime,mode=1777            0         0

tmpfs                /var/log  tmpfs     defaults,relatime,mode=1777            0         0
```

```
# /etc/rc.local

# ...
mkdir -p /var/log/cups;  chmod 0755 /var/log/cups;
mkdir -p /var/log/exim4; chmod 2750 /var/log/exim4; chown Debian-exim:adm /var/log/exim4;
mkdir -p /var/log/fsck;  chmod 0755 /var/log/fsck;
mkdir -p /var/log/gdm3;  chmod 1770 /var/log/gdm3;  chown root:Debian-gdm /var/log/gdm3;
# ...
```

I/O Scheduler

Der I/O Scheduler regelt und gruppiert anstehende I/O Requests:

- **Noop** FIFO scheduler (first in, first out)
- **Deadline** Jeder I/O request erhält Deadline zur Ausführung
- **Anticipatory** Gruppiert multiple synchrone Requests (nur bis 2.6.33)
- **CFQ** 'Completely Fair Queuing', arbeitet mit Timeslices

CFQ ist der Linux Standard-Scheduler, hat Optimierungen für rotierende Medien

Noop oder **Deadline** kann Vorteile bei SSDs bringen (keine Positionierungszeit, Schreiben großer Blöcke auf einmal)

Aber: Ausprobieren und Kernel-Dokumentation lesen!
I/O Scheduler neuerer Kernel bekommen zunehmend SSD-Unterstützung.

```
# I/O scheduler auf 'noop' setzen
echo noop > /sys/block/sdX/queue/scheduler
```

Virtual Memory Subsystem anpassen

a) Laptop Modus aktivieren (Geräte mit Batterieversorgung)

```
> echo 5    > /proc/sys/vm/laptop_mode
> echo 6000 > /proc/sys/vm/dirty_writeback_centisecs
```

b) Swappiness vermindern (bei genug Arbeitsspeicher)

```
> echo 1    > /proc/sys/vm/swappiness
> echo 50   > /proc/sys/vm/vfs_cache_pressure
```

c) Ramzswap aktivieren (Komprimierte Ramdisk als Swapspace)
Noch experimentelles Kernelfeature!

```
> modprobe ramzswap
> rzscontrol /dev/ramzswap0 --init
> mkswap /dev/ramzswap0
> swapon /dev/ramzswap0
```

Beispiel: Generische für SSD optimierte *fstab*

```
# /etc/fstab: static file system information

# <file system>  <mount>      <type>   <options>                          <dump> <pass>

/dev/sda1        /boot        ext2     noatime                            0        2
/dev/sda2        none         swap     sw                                 0        0
/dev/sda3        /            ext4     noatime,data=writeback,discard     0        1
/dev/sda4        /home        ext4     noatime,data=writeback,discard     0        2

tmpfs            /tmp         tmpfs    defaults,relatime,mode=1777        0        0
tmpfs            /var/run     tmpfs    defaults,relatime,mode=1777        0        0
tmpfs            /var/lock    tmpfs    defaults,relatime,mode=1777        0        0

# Optional: Log Verzeichnis in tmpfs mounten. Nur falls Logs nicht benötigt werden!
tmpfs            /var/log     tmpfs    defaults,relatime,mode=1777        0        0
```

Beispiel: Generische für SSD optimierte *rc.local*

```
#!/bin/sh -e
# /etc/rc.local

# Writeback cache aktivieren
hdparm -W1 /dev/sda

# I/O scheduler auf 'noop' setzen
echo noop > /sys/block/sda/queue/scheduler

# Virtual memory subsystem optimieren
echo 5    > /proc/sys/vm/laptop_mode
echo 6000 > /proc/sys/vm/dirty_writeback_centisecs
echo 1    > /proc/sys/vm/swappiness
echo 50   > /proc/sys/vm/vfs_cache_pressure

# Unterverzeichnisse in /var/log anlegen und Berechtigungen setzen
# (nur falls /var/log in tmpfs gemounted)
mkdir -p /var/log/apt;            chmod 0755 /var/log/apt;
mkdir -p /var/log/ConsoleKit;     chmod 0755 /var/log/ConsoleKit;
mkdir -p /var/log/cups;           chmod 0755 /var/log/cups;
mkdir -p /var/log/exim4;          chmod 2750 /var/log/exim4;
mkdir -p /var/log/fsck;           chmod 0755 /var/log/fsck;
mkdir -p /var/log/gdm3;           chmod 1770 /var/log/gdm3;
mkdir -p /var/log/installer;      chmod 0755 /var/log/installer;
mkdir -p /var/log/iptraf;         chmod 0755 /var/log/iptraf;
mkdir -p /var/log/news;           chmod 0755 /var/log/news;
mkdir -p /var/log/sysstat;        chmod 0755 /var/log/sysstat;
chown Debian-exim:adm /var/log/exim4;
chown root:Debian-gdm /var/log/gdm3;
```

I/O überwachen / analysieren

iostat -xk	I/O Überwachung Devices
iotop -oa	I/O Überwachung Prozesse/Threads
cat 1 > /proc/sys/vm/block_dump tail -f /var/log/syslog	Loggt alle I/O Zugriffe in syslog (nach Analyse wieder deaktivieren!)
smartctl -a <*device*>	Auslesen SMART-Speicher der SSD ➔ *Reallocated_Sector_Count* (> 0: SSD wechseln) ➔ *Media_Wearout_Indicator* (100%: fabrikneu, 0%: Ende Lebensdauer)
tune2fs -l <partition>	Lifetime writes (ungenau wg. Write amplification)

Firefox Cache in Ramdisk auslagern (schnell & einfach)

Voraussetzung: `/tmp` ist auf `tmpfs` gemounted

Firefox Konfiguration mit *about:config* in Adresszeile aufrufen:

neuen Key (`string`) anlegen (right-click -> Neu -> String)

- Name: `browser.cache.disk.parent_directory`
- Wert: `/tmp/firefox-cache`

Firefox Profil in Ramdisk auslagern (aufwändiger aber konsequent)

Neues Firefox Profil anlegen

- Name: `ramdisk`
- Ordner: `/tmp/firefox-ramdisk`

```
> mkdir /tmp/firefox-ramdisk
> firefox -ProfileManager
  ...
> cd /tmp
> tar -czf firefox.ssd.tgz firefox-ramdisk
> mv firefox.ssd.tgz ~/.mozilla/firefox/
```

Firefox ab dann über Skript starten:

```
#!/bin/sh
# firefox.sh: Script to start firefox with ramdisk profile

if [ -f ~/.mozilla/firefox/firefox.ssd.tgz ] ; then
     mv ~/.mozilla/firefox/firefox.ssd.tgz ~/.mozilla/firefox/firefox.ssd.tgz.bak
fi

cp ~/.mozilla/firefox/firefox.ssd.tgz.bak /tmp

cd /tmp; tar -xzf firefox.ssd.tgz.bak; rm firefox.ssd.tgz.bak

firefox -P ramdisk

cd /tmp; tar -czf firefox.ssd.tgz firefox-ramdisk
mv /tmp/firefox.ssd.tgz ~/.mozilla/firefox/
```

Links & Referenzen

Guidelines und Anleitungen SSDs unter Linux konfigurieren:

http://wiki.ubuntuusers.de/SSD
https://wiki.archlinux.org/index.php/Solid_State_Drives
http://de.gentoo-wiki.com/wiki/Solid_State_Drive
http://docs.fedoraproject.org/en-US/Fedora/14/html/Storage_Administration_Guide/index.html
http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Storage_Administration_Guide/newmds-ssdtuning.html

http://www.thomas-krenn.com/de/wiki/Kategorie:SSDs

c't Magazin 2011/22, 152-154 „Meldevorgang. Linux für SSDs konfigurieren“

Ramzswap Projekt:

http://code.google.com/p/compcache/

Linux MTD Subsystem Project:

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

... und die Manpages von **hdparm, smartctl, tune2fs, iostat, mount**