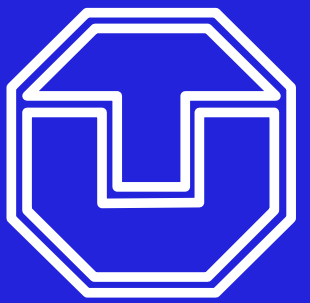
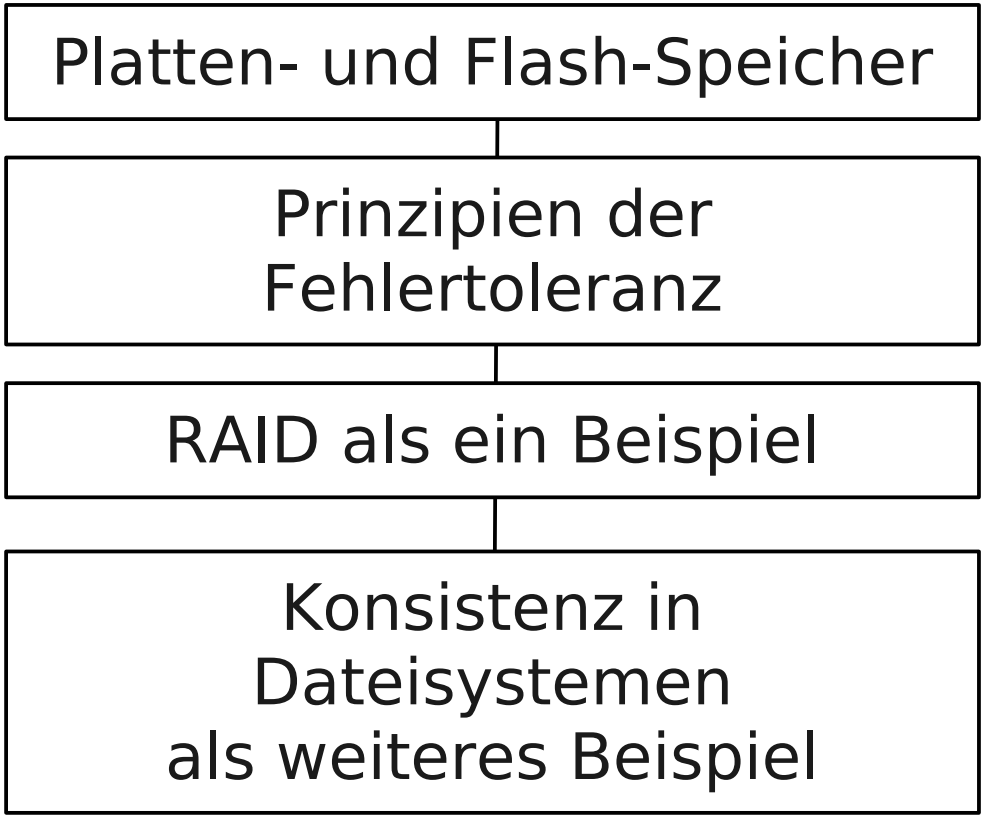
Hardware & Fehlertoleranz, Beispiel: Dateisysteme

Betriebssysteme

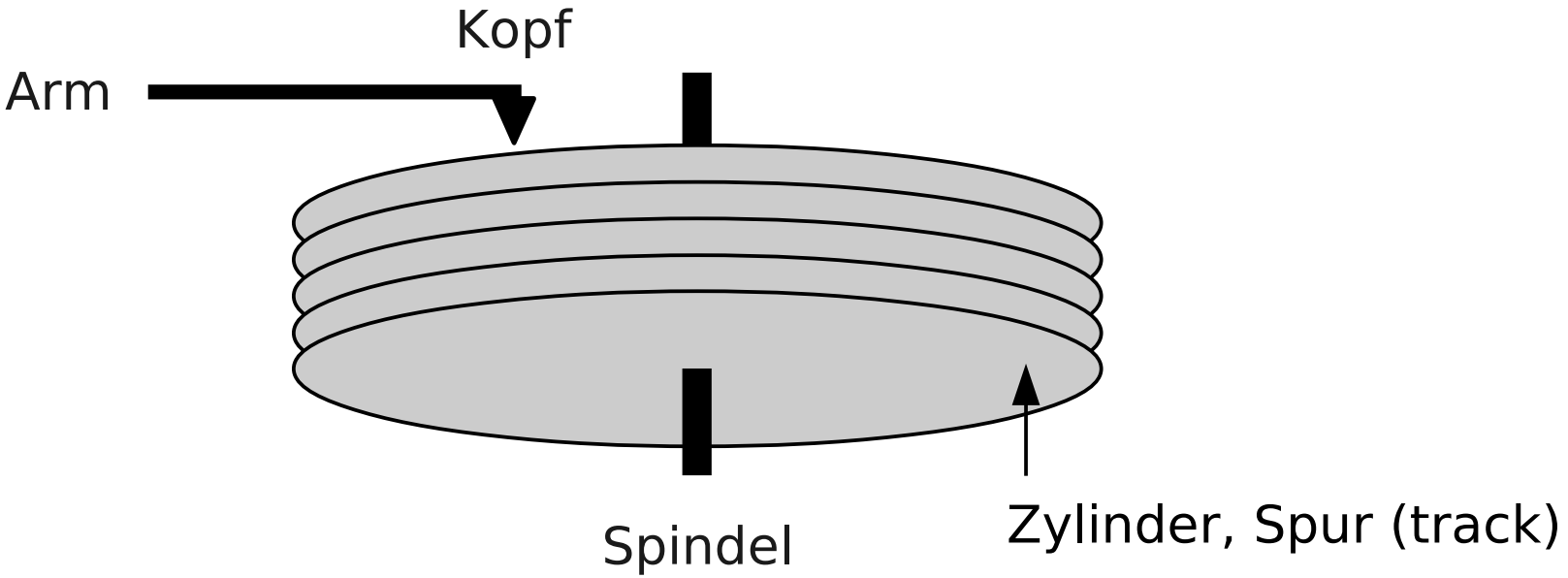
Hermann Härtig
TU Dresden



Wegweiser


Platten- und Flash-Speicher
Prinzipien der Fehlertoleranz
RAID als ein Beispiel
Konsistenz in Dateisystemen als weiteres Beispiel

Funktionsweise von Platten



Funktionsweise von Platten

Eigenschaften

- Persistente Speicherung
- Lesen/Schreiben in Einheiten von Blöcken (Sektoren) an beliebiger Stelle der Platte
- Beliebig häufiges Lesen und Schreiben
- Mechanik:
  Kopf verschieben / Scheiben rotieren

Block:

Header

Tail: error correction codes

Eigenschaften von Platten

Entwicklung der Plattentechnik

- Latenz: verbessert sich nur sehr allmählich
- Kapazität: wächst sehr schnell
- Optimierung: Flash-memory in Plattenlaufwerken

Im Vergleich zu anderen Komponenten

- CPU-Leistung wächst sehr schnell
- Netzleistung wächst sehr schnell

→ Dateisystem: außerordentlich kritisch für die Leistungsfähigkeit von Rechnern

Kenngrößen von Platten

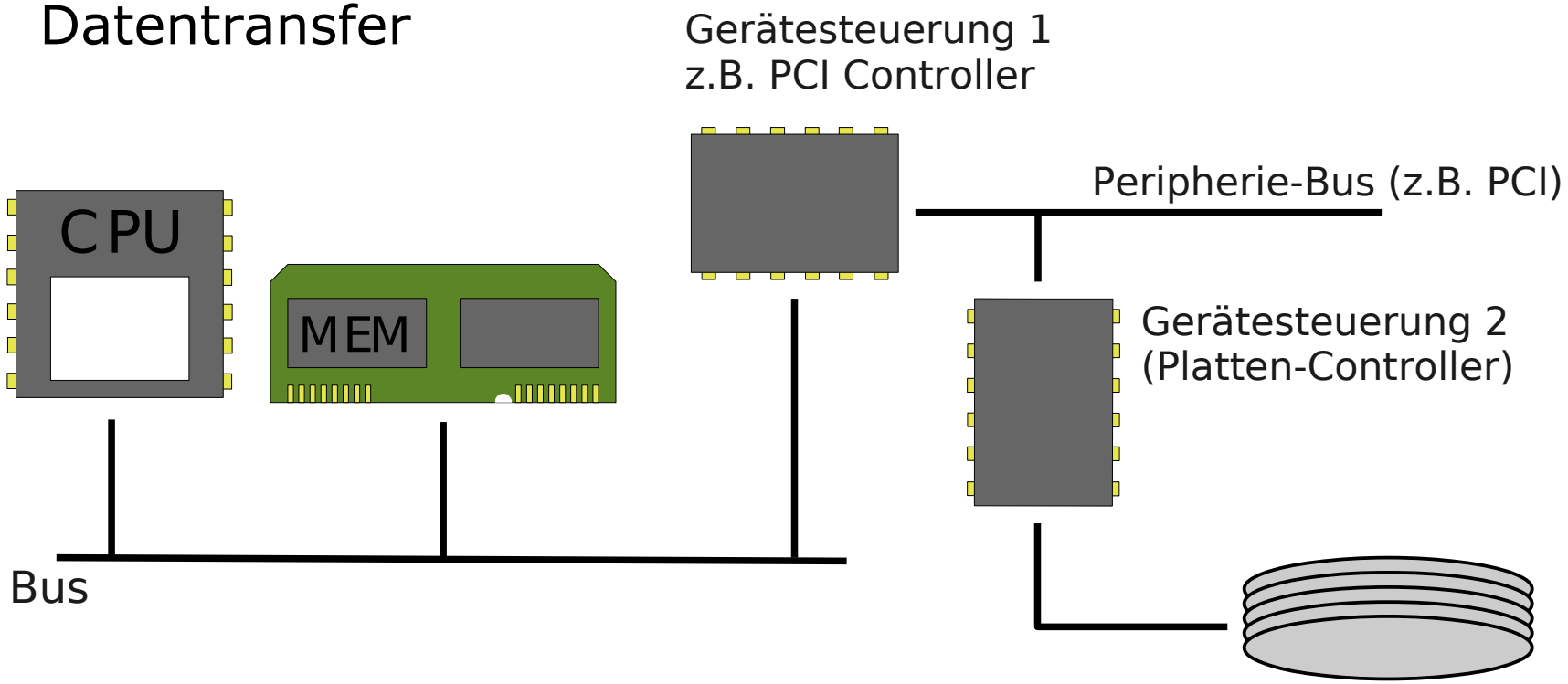
Größe, Kapazität und Zugriffsgeschwindigkeit sind bestimmt durch:

- Drehgeschwindigkeit
- Spurdichte (Zoning)
- Schreibdichte
- Beschleunigungs- und Bewegungszeiten
- Settle-Time (Kopfberuhigung, Kalibrierung)
- Caching
- Bus

Ablauf eines Plattenzugriffs (SCSI)

Rechner - Gerätesteuerung (1): Kommando, Adresse
Gerätesteuerung (2): Kommando, Adresse

- Umrechnung der Adresse in plattengeometrische Größen
- „Seek“: Start-Zeit, Bewegung, Stop-Zeit
- Abwarten Rotation
- Datentransfer



Zugriffszeit

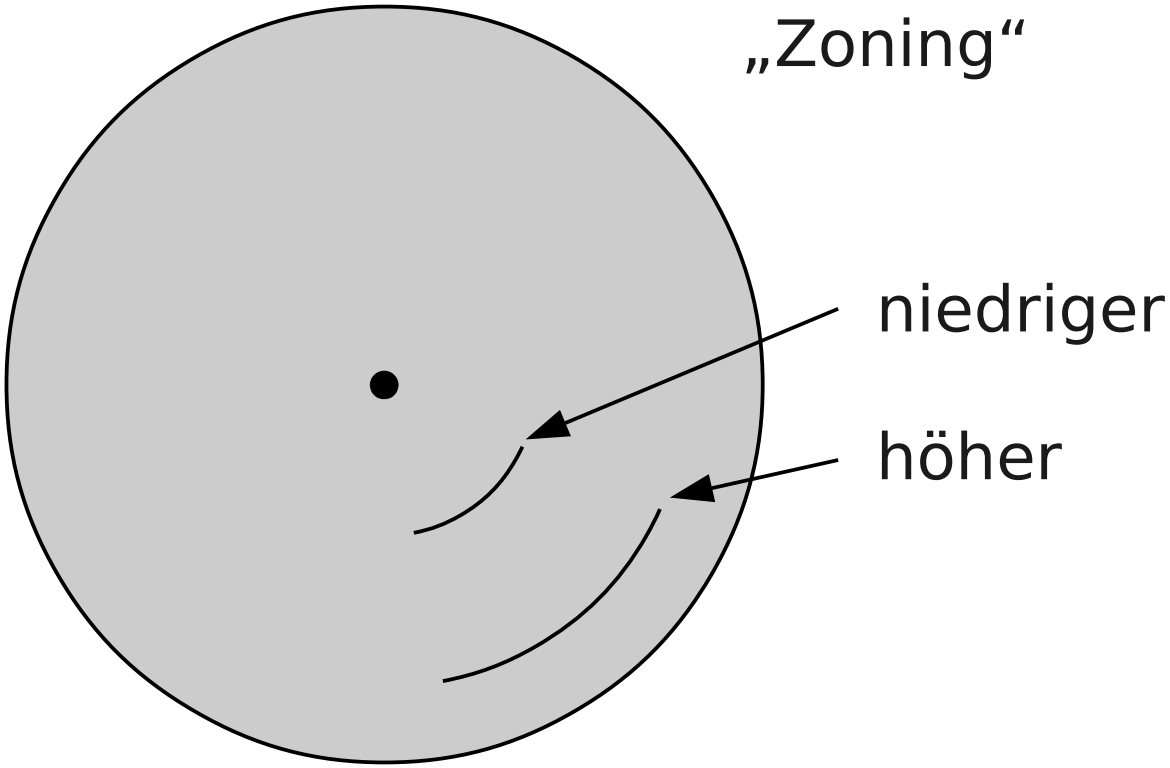
... setzt sich zusammen aus:

- Kommandogenerierung und -transfer
- seek(Positionieren):
  - Beschleunigung
  - Bewegung
  - Abbremsen
  - Kopfberuhigung
  - Spur-Identifikation
- Rotations-Latenz
- Daten-Transfer

Bandbreite

... ist die übertragene Informationsmenge pro Zeiteinheit (MByte/s) (Drehgeschwindigkeit konstant)

Laufwerk:



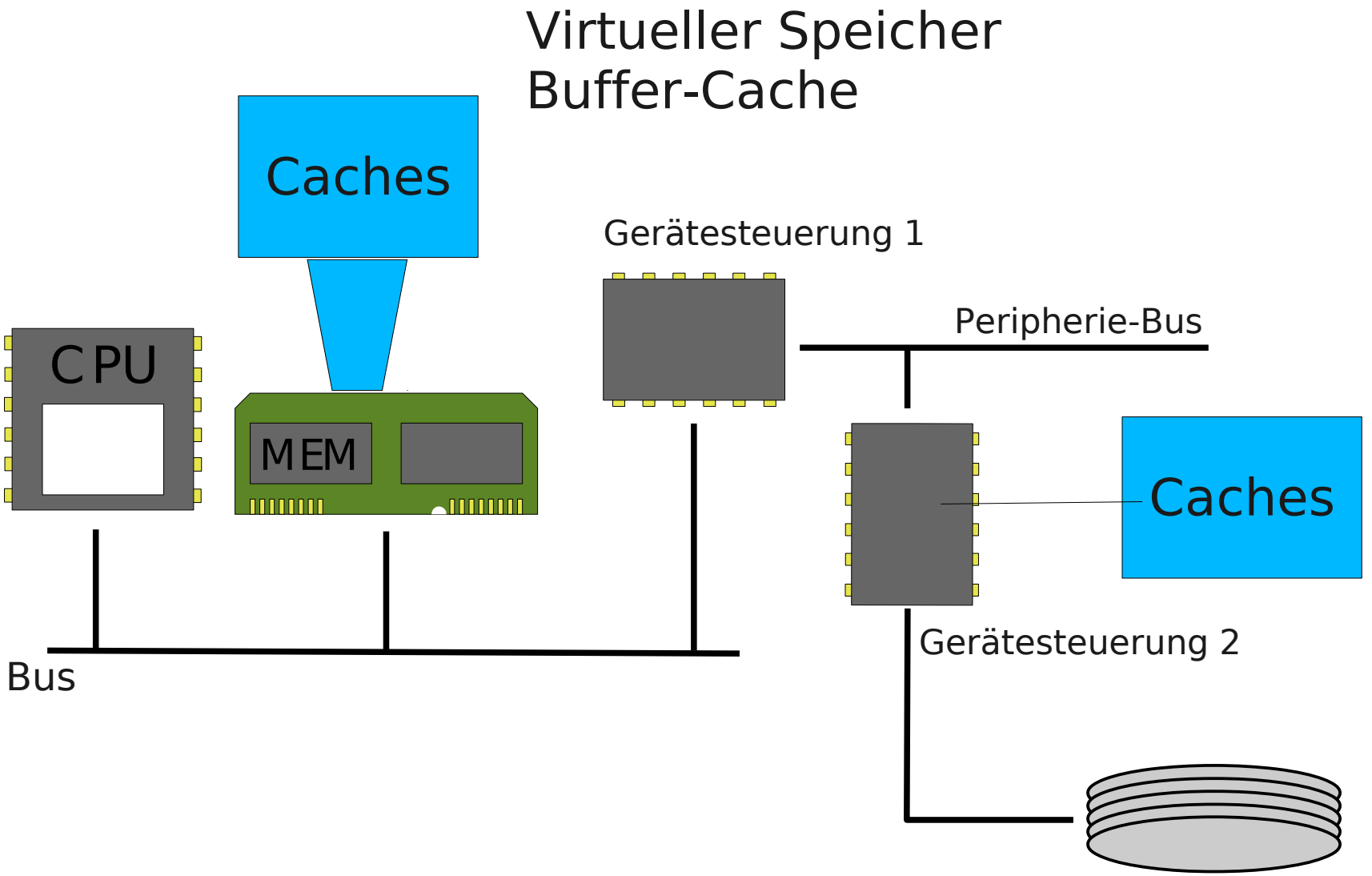
Beispiel

- Abstand von Spindel: 1 cm / 2 cm
- Spurumfang: 6 cm / 12 cm

Konsequenzen

- Allokation: Block → Dateien
  - Blöcke einer Datei in der „Nähe“ zueinander
  - Dateien mit großer Anforderung an Bandbreite nach „außen“
  - Ziel: möglichst große Lese-/Schreiboperationen
- Vorauslesen („read ahead“)
- Plattenaufträge sortieren
  - Nicht mehr FIFO
  - Minimiere mechanische Bewegungen (seek, rotate)
  - Auf Hardware- und Betriebssystem-Ebene

Caches


Virtueller Speicher
Buffer-Cache
Caches
Gerätesteuerung 1
Peripherie-Bus
CPU
MEM
Caches
Gerätesteuerung 2
Bus

Beispiel Serverplatte (leicht veraltet)

SCSI(3) Ultra320 (BUS):

- max. Bandbreite 320 MB/s

IBM ...

- Kapazität 73,4 GB
- Zonen 17
  - innen 29,8 MB/s
  - außen 58,0 MB/s
- Tracks per Inch 27.312
- Umdrehungen/Min. 10.000
- Speicherdichte 13,2 GB/square inch
- Average seek time 4,9 ms

Flash-Speicher

- Entwickelt aus EEPROM
  - EEPROM: einmal schreibbar, danach teurer Lösch-Zyklus vor Wiederbeschreiben
- Lese-Geschwindigkeiten ähnlich zu DRAM
- Im Gegensatz zu Festplatten weniger anfällig für
  - Magnetismus
  - Stöße
- Herstellungskosten höher als bei Festplatten
- Basis für moderne SSDs u.ä.

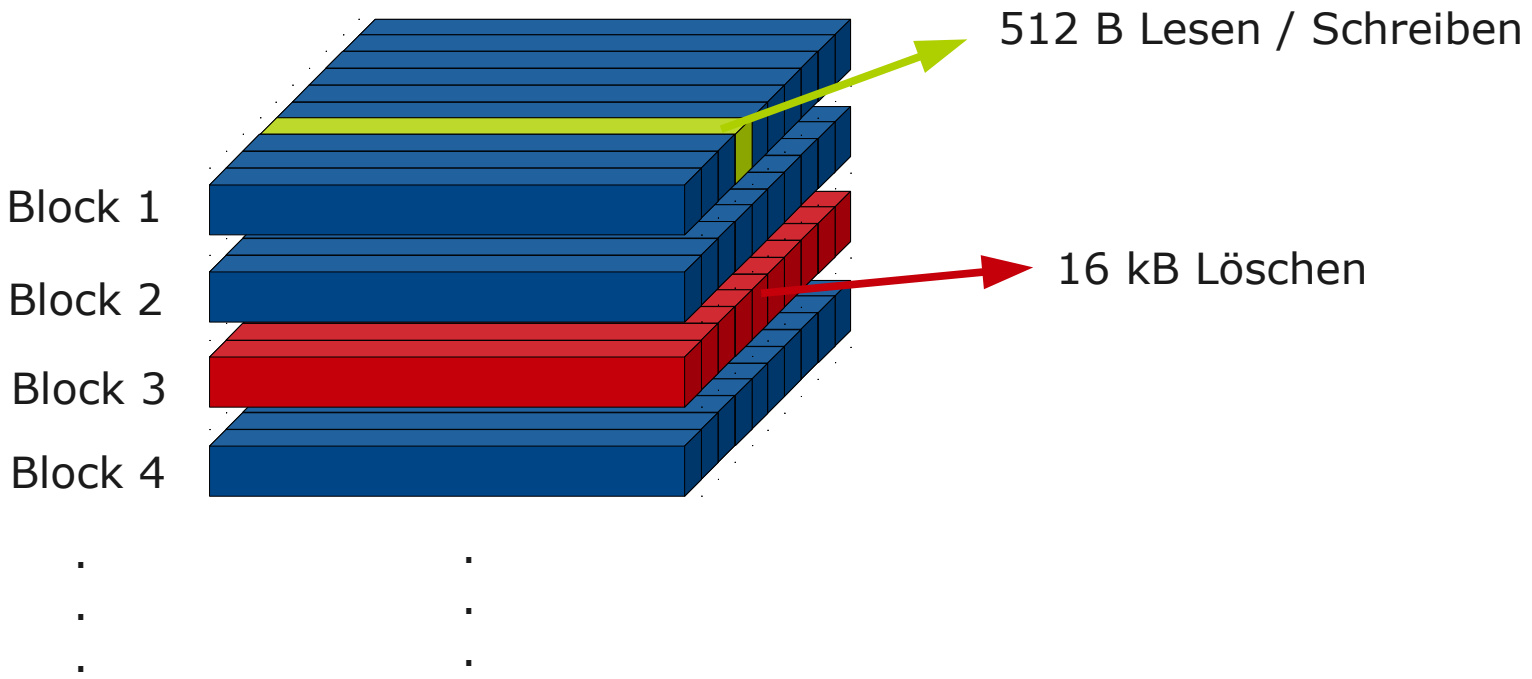
Flash-Speicher: Performance

Medium	Lesen	Schreiben	Löschen
DRAM	13.9 ns / 1 B 7.1 µs / 512 B	13.9 ns / 1 B 7.1 µs / 512 B	-
NOR Flash	45.0 ns / 1 B 23.0 µs / 512 B	14.0 µs / 1 B 7.2 ms / 512 B	18.0 ms / 128 kB
NAND Flash	15.0 µs / 1 B 35.0 µs / 8 kB	300 µs / 1 B AVG: 350 µs / 8 kB MAX: 500 µs / 8 kB	1.5 – 3 ms / 1 MB
Festplatte	AVG: 8.2 ms / 512 B	AVG: 9.2 ms / 512 B	-

T. Kuo: *„Challenges and Solutions for Consumer Flash-Memory Devices*“, National Taiwan University, CODES 2011, Taipeh 2011

Flash-Speicher: Aufbau

- Schreiben: seitenweise (1 page = 512 B)
- Überschreiben erst nach vorherigem Löschvorgang möglich (analog zu EEPROM)
  - Granularität: 1 Block = 32 Seiten = 16 kB



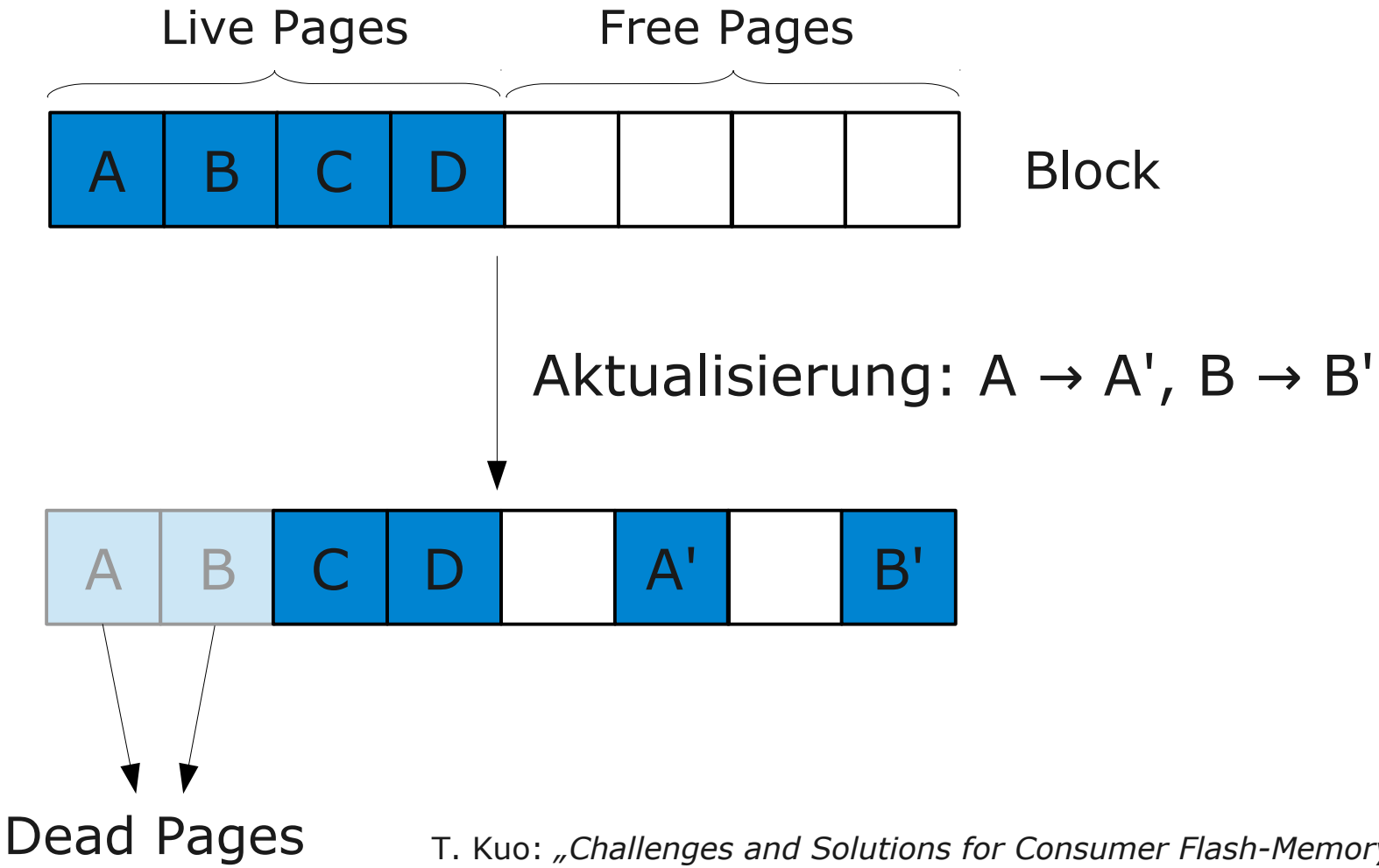
T. Kuo: *„Challenges and Solutions for Consumer Flash-Memory Devices"*, National Taiwan University, CODES 2011, Taipeh 2011

Flash-Speicher: Anbindung

- Standard-Dateisystem (ext4...) auf Flash-Gerät:
  - Optimiert für Festplatten mit hohen Seek-Zeiten → Flash-Seek-Zeit quasi 0
  - „Löschen durch Überschreiben“ nicht möglich wegen blockweisem Löschen bei Flash
- Spezielles Flash-Dateisystem (im BS) oder Flash Translation Layer in Hardware:
  - Fehlerkorrektur, Garbage Collection
  - Begrenzte Anzahl von Schreiboperationen pro Flash-Zelle möglich (10.000 – 100.000)
    - **Wear Leveling**: gleichmässige Verteilung von Schreiboperationen über alle Blöcke

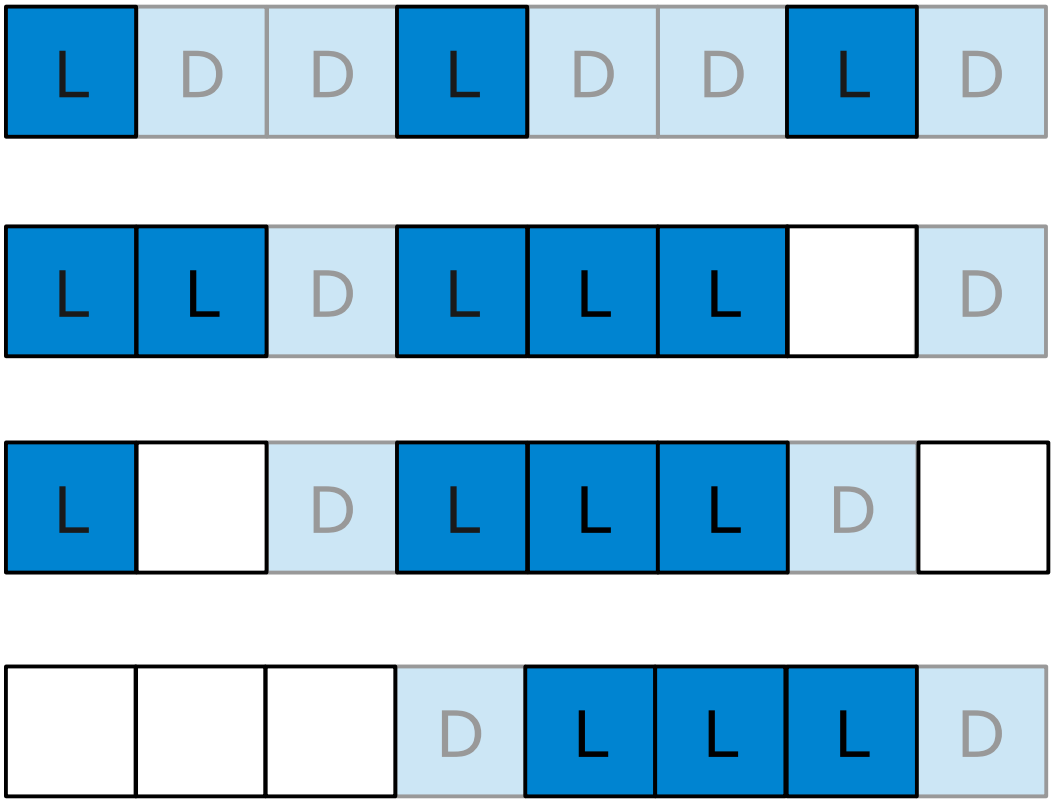
Flash-Speicher: Aktualisierung

Aktualisierung durch Schreiben in freie Seite
→ kein Löschaufwand



T. Kuo: *„Challenges and Solutions for Consumer Flash-Memory Devices"*, National Taiwan University, CODES 2011, Taipeh 2011

Flash-Speicher: Verwaltung



Schreiben des Blockes nicht mehr möglich

→ Garbage Collection

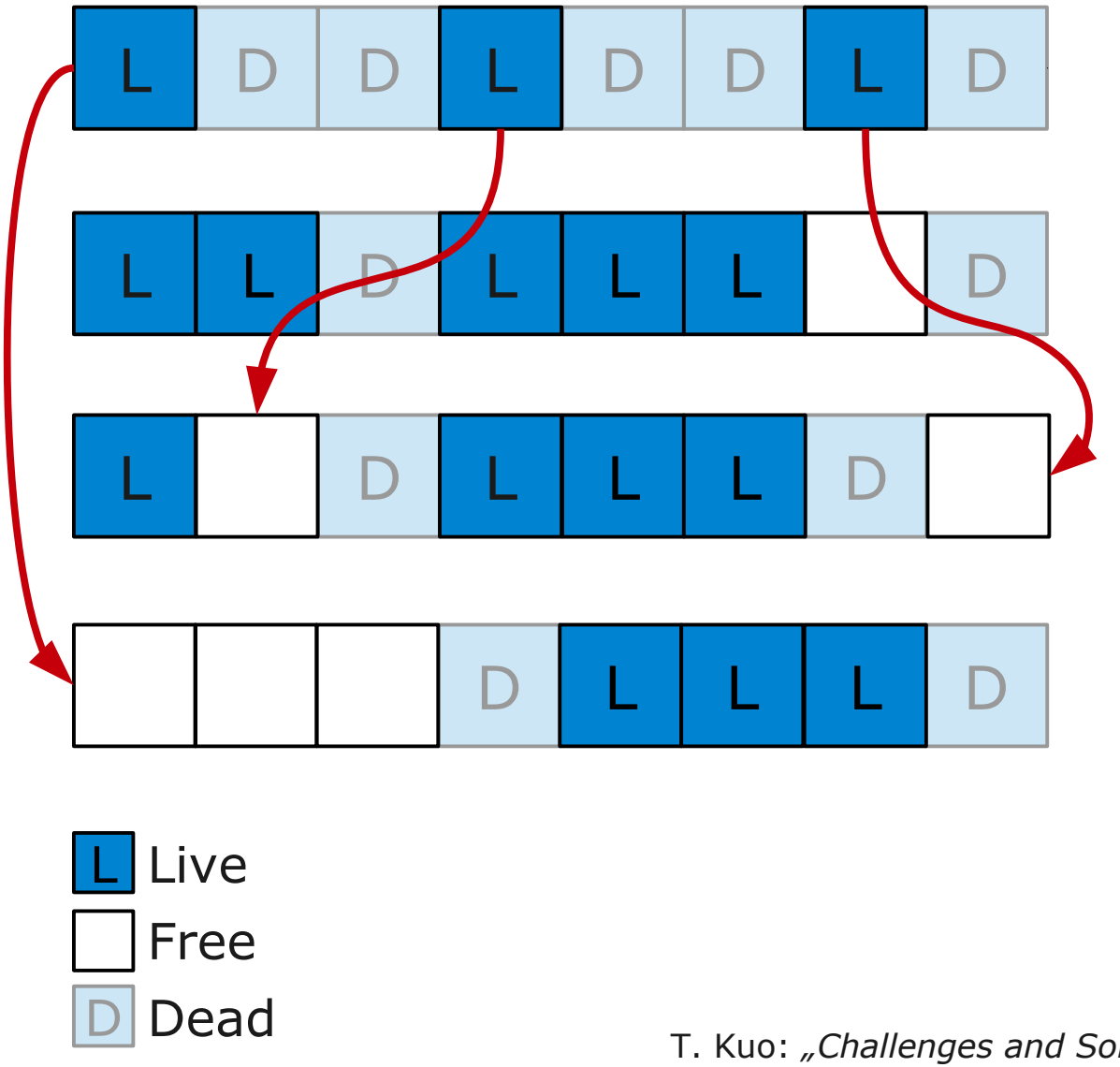
L Live

Free

D Dead

T. Kuo: *„Challenges and Solutions for Consumer Flash-Memory Devices"*, National Taiwan University, CODES 2011, Taipeh 2011

Flash-Speicher: Verwaltung

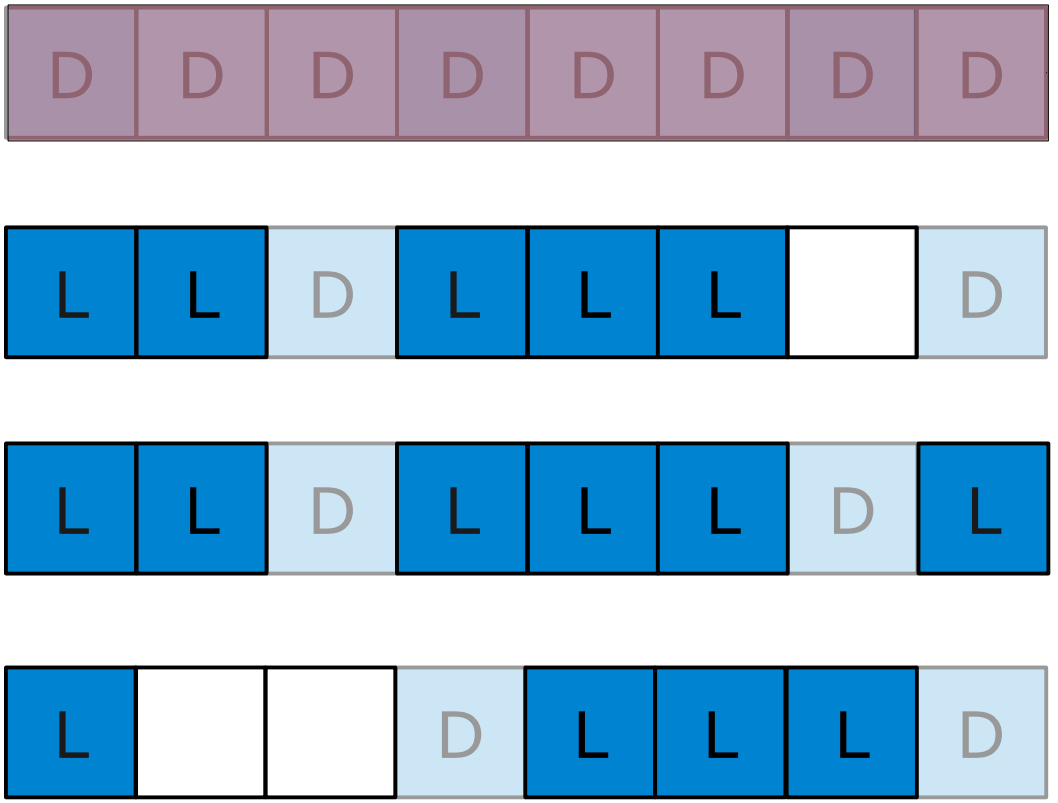


Garbage Collection:

(1) Kopieren der lebendigen Seiten in freie Seiten

T. Kuo: *„Challenges and Solutions for Consumer Flash-Memory Devices"*, National Taiwan University, CODES 2011, Taipeh 2011

Flash-Speicher: Verwaltung



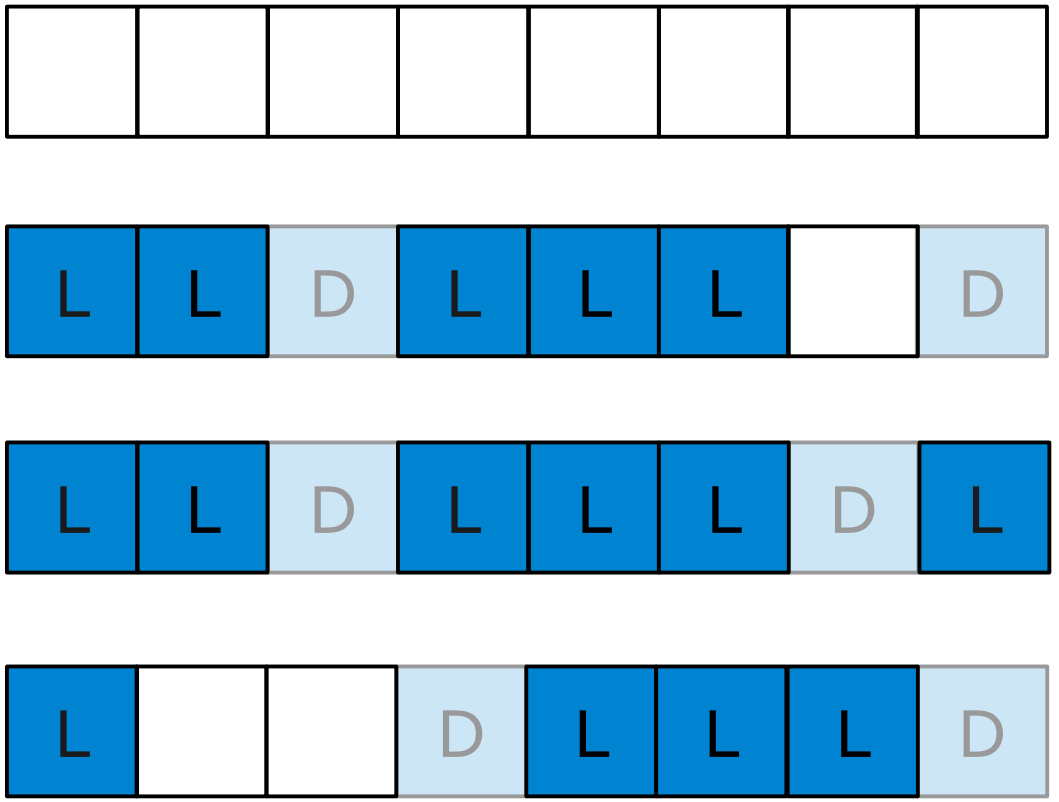
Garbage Collection:

(1) Kopieren der lebendigen Seiten in freie Seiten

(2) Löschen des Original-Blocks

T. Kuo: *„Challenges and Solutions for Consumer Flash-Memory Devices"*, National Taiwan University, CODES 2011, Taipeh 2011

Flash-Speicher: Verwaltung



Garbage Collection:

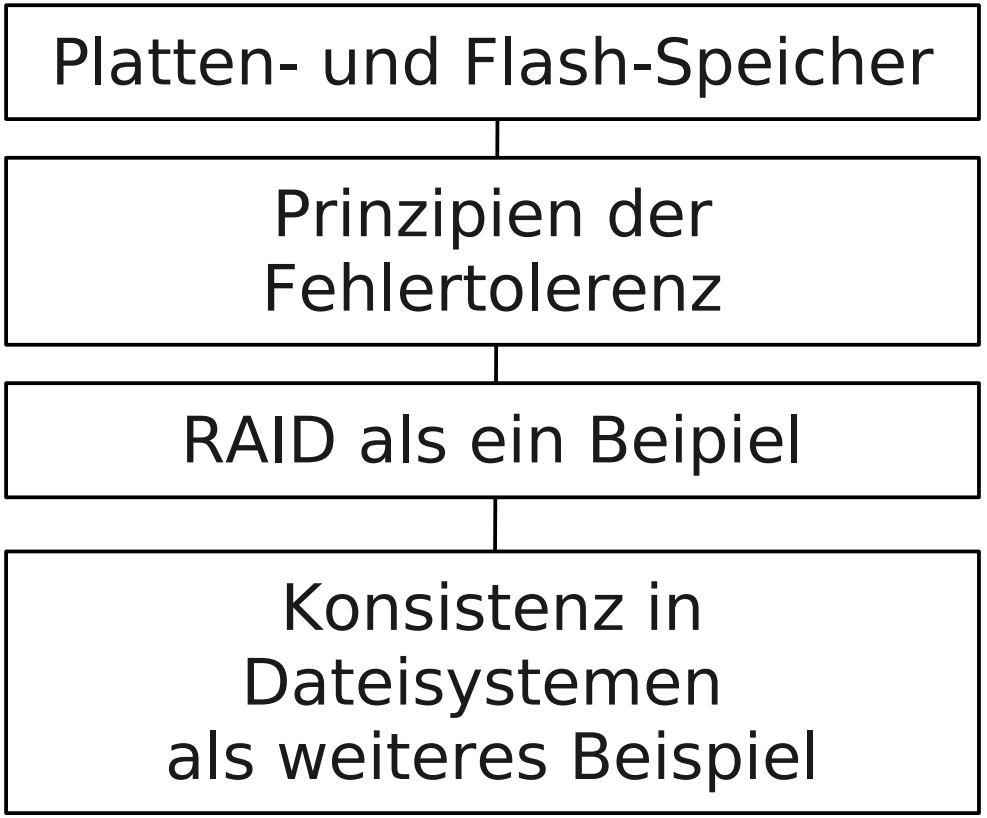
(1) Kopieren der lebendigen Seiten in freie Seiten

(2) Löschen des Original-Blocks

(3) Wiederverwendung der freigegebenen Seiten

T. Kuo: *„Challenges and Solutions for Consumer Flash-Memory Devices"*, National Taiwan University, CODES 2011, Taipeh 2011

Wegweiser


Platten- und Flash-Speicher
Prinzipien der Fehlertolerenz
RAID als ein Beipiel
Konsistenz in Dateisystemen als weiteres Beispiel

Fehlertoleranz: Aufgaben

- Fehlertypen festlegen („Fehlermodell“)
- Fehler erkennen („Detection“)
- Ausbreitung beschränken („Fault Isolation“)
- Fehler beheben („Recovery“)
- Reparatur

→ Redundanz als Mittel zur Fehlertoleranz

Fehlertypen (am Beispiel Festplatte)

- Zur Laufzeit
  - **Transient:**
    - „Überschreiben“ der Daten in einzelnen Blöcken
    - z.B. durch Erdmagnetismus, Strahlung, …
    - Reparatur: neu schreiben
  - **Permanent:**
    - Nicht korrigierbarer Blockfehler (defekter Block)
    - Ausfall eines gesamten Laufwerkes
    - Systemabsturz (Inkonsistenz von Datenstrukturen auf Platte)
    - Reparatur: Festplatte austauschen
- Entwurfsfehler

Formen der Redundanz

- **Zeit**
  → Mehrfache Ausführung von Operationen, Vergleich der Ergebnisse

- **Struktur**
  → Nutzung mehrfach ausgelegter Funktionseinheiten (z.B. Speicherung von Daten auf 2 Festplatten)

- **Information**
  → Fehler-korrigierende Codes als Bestandteil der gespeicherten Daten

Zusätzliche Funktionalität fügt auch zusätzliche Fehlerquellen hinzu!

Transiente Lesefehler

- Jeder Block auf Platte hat Head/Tail mit redundanter Codierung
  - Error Correcting Codes (ECC)

- Fehlererkennung und -korrektur durch Auswertung dieses Codes

- Buchführung zu Wartungszwecken
  - Statistik: Festplatten mit höheren (transienten) Fehlerraten fallen auch bald komplett aus

Permanente Lesefehler: Defekte Blöcke

Erkennung durch redundante Codes
→ häufig schon bei Herstellung

Vermeidung statt Behebung:

- Plattenspuren haben „Bad Sector"-Markierungen
- In Software:
  - Datei mit Liste aller defekten Blöcke (Vorsicht bei Backup!)
  - Blöcke als „allokiert" markiert
- Heute eher schon im Festplatten-Controller

Ausfall von Laufwerken

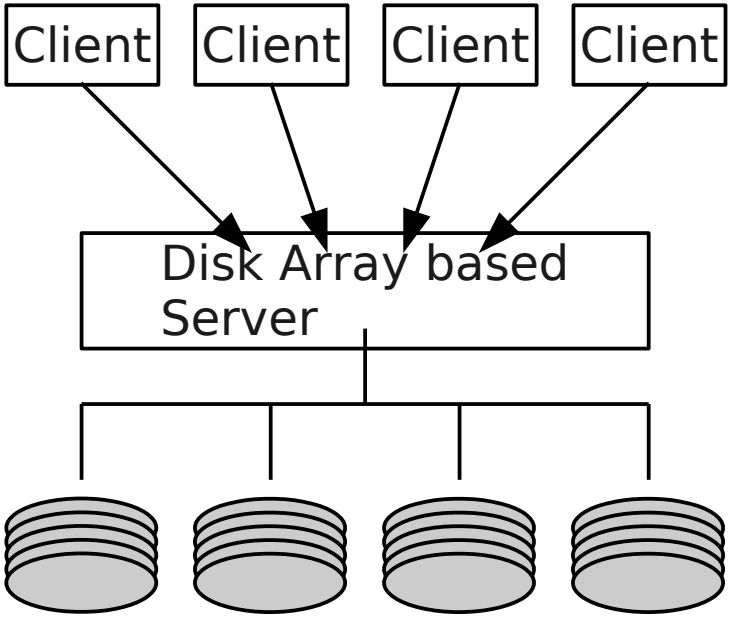
- Entdeckung:
  - Timeout
  - sich häufende Lesefehler
- Recovery: Redundanz durch RAID
- Reparatur: Plattentausch

RAID - Redundant Array of Independent Disks

Zusammenbau mehrere unabhängig ansteuerbarer Platten zu einem Array

Ziel

- Bessere Leistung durch parallele Zugriffe
- Ausgleich der Lücke zwischen immer schnelleren Prozessoren/Speichern und nach wie vor langsamen Platten

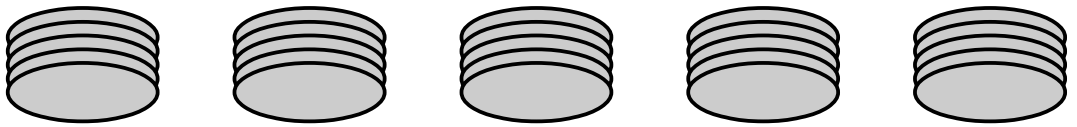


Nachteil

- Mittlere Zeit bis zum Ausfall des Feldes ist kleiner
- ➔ Maßnahmen zur Fehlertoleranz !

Ohne Redundanz (RAID 0)

- Disk Array aus n Platten
- Keine Kosten für Redundanz (trivial)
- Keine Fehlertoleranz



Disk 1 . . . Disk n

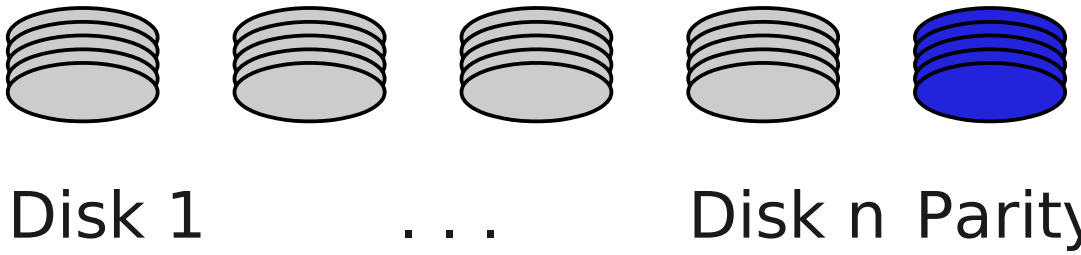
Spiegelplatten (RAID 1)

- Disk Array aus 2n Platten
- Jede Platte gespiegelt (identische Kopien)
- Schreiben: alle Daten zweimal
- Lesen: einmal, von schnellerer Platte
- Fehlertolerant (ein beliebiger Ausfall)
- Hohe Speicherkosten

Disk 1 . . . Disk n Disk 1' . . . Disk n'

Einzelne Paritätsplatte (RAID 4)

- Nutzerdaten werden blockweise auf n Platten verteilt (striping)
  Dateiblock 1 auf Disk 1, 2 auf 2, ...
- Blockweise Paritätsbildung
  (Disk 1, Block 1) XOR (Disk 2, Block 1) ... → (Disk P, Block 1)
- Fehlererkennung durch jede Platte separat
  Rekonstruktion durch Paritätsbildung



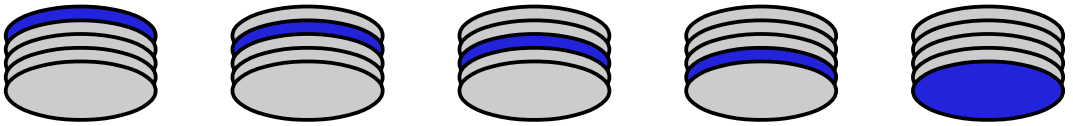
RAID Level 5

Probleme mit RAID 4:

- bei hohem Leseanteil: Paritäts-Platte nicht richtig ausgelastet
- bei vielen kleinen Dateien und Schreiboperationen: Engpass an Paritätsplatte

Block Interleaved Distributed Parity

- Paritätsblöcke über gesamtes Array verteilt



- beste Leistung bei kleinen und großen Leseanforderungen sowie bei großem Anteil Schreiboperationen
- kleine Schreibanforderungen immer noch ineffizienter, da Paritätsblock jedesmal lokalisiert und geschrieben werden muss

Zusammenfassung

Aufgabe : Verwaltung und persistente Speicherung großer Datenmengen

Problem: Lücke in der Zugriffsgeschwindigkeit

Weiterführendes

Konsistenzmechanismen in Dateisystemen
(Journaling, Logging, etc. → im Januar)

Verteilte Dateisysteme