5.Vorlesung Grundlagen der Informatik

Christian Baun

Hochschule Darmstadt
Fachbereich Informatik
christian.baun@h-da.de

10.11.2011

Wiederholung vom letzten Mal

- Rechnerarchitektur – Von-Neumann-Architektur
- Hardware-Komponenten eines Computers
- Speicher

Heute

- Ersetzungsstrategien
- Festplatten
- Solid State Drives
- Redundant Array of Independent Disks (RAID)

Cache-Datenverwaltung – Ersetzungsstrategien

- Es ist sinnvoll, immer nur die Datenblöcke (Seiten) im Cache zu halten, auf die häufig zugegriffen wird
- Gängige Ersetzungsstrategien orientieren sich neben der Größe auch am Zeitpunkt des letzten Aufrufs
- Einige wichtige **Ersetzungsstrategien**:
  - **OPT** (Optimale Strategie)
  - **LRU** (Least Recently Used)
  - **LFU** (Least Frequently Used)
  - **FIFO** (First In First Out)
  - **Clock**
  - **Climb**
  - **TTL** (Time To Live)
  - **Random**
  - **WS** (Working Set)

Optimale Strategie (OPT)

- Verdrängt den Datenblock, der am längsten nicht gebraucht, also auf den am längsten in der Zukunft nicht zugegriffen wird
- Der komplette Programmablauf, also die zukünftige Verwendung der Datenblöcke im Speicher muss im voraus bekannt sein
  - Das ist leider utopisch und nicht zu implementieren!
- Da niemand in die Zukunft sehen kann, müssen wir die Vergangenheit berücksichtigen
- Die Strategie hat ihre Berechtigung, um die Effizienz anderer Ersetzungsstrategien mit ihr zu vergleichen

Cache-Anfrage:	**1**	**2**	**3**	**4**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
1. Datenblock:	1	1	1	1	1	1	1	1	1	3	3	3
2. Datenblock:		2	2	2	2	2	2	2	2	2	4	4
3. Datenblock:			3	4	4	4	5	5	5	5	5	5

→ 7 Fehler

Least Recently Used (LRU)

- Verdrängt den Datenblock, auf den **am längsten nicht mehr** zugegriffen wurde
- Alle Datenblöcke werden in einer Warteschlange eingereiht
- Ist der Cache voll und es kommt zum Cache-Miss, wird der Datenblock am Ende der Warteschlange ausgelagert
- Wird ein Datenblock in den Cache geschrieben oder referenziert, wird er am Anfang der Warteschlange eingereiht
- Vorteil: Liefert gute Resultate und verursacht wenig Overhead
- Nachteil: Zugriffshäufigkeit wird nicht berücksichtigt

Cache-Anfrage:	**1**	**2**	**3**	**4**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
1. Datenblock:	1	1	1	2	3	4	1	2	5	1	2	3
2. Datenblock:		2	2	3	4	1	2	5	1	2	3	4
3. Datenblock:			3	4	1	2	5	1	2	3	4	5

→ 10 Fehler

Least Frequently Used (LFU)

- Verdrängt den Datenblock, auf den **am wenigsten** zugegriffen wurde
- Für jeden Datenblock im Cache wird in der Seitentabelle ein Referenzzähler geführt, der die Anzahl der Zugriffe speichert
  - Jeder Zugriff erhöht den Referenzzähler
- Ist der Cache voll und kommt es zum Cache-Miss, wird der Datenblock entfernt, dessen Referenzzähler den niedrigsten Wert hat
- Vorteil: Berücksichtigt die Zugriffshäufigkeit
- Nachteil: Datenblöcke, auf die in der Vergangenheit häufig zugegriffen wurde, können den Cache blockieren

Cache-Anfrage:	**1**	**2**	**3**	**4**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
1. Datenblock:	$_1$1	$_1$1	$_1$1	$_1$4	$_1$4	$_1$4	$_1$5	$_1$5	$_1$5	$_1$3	$_1$4	$_1$5
2. Datenblock:		$_1$2	$_1$2	$_1$2	$_1$1	$_1$1	$_1$1	$_2$1	$_2$1	$_2$1	$_2$1	$_2$1
3. Datenblock:			$_1$3	$_1$3	$_1$3	$_1$2	$_1$2	$_1$2	$_2$2	$_2$2	$_2$2	$_2$2

→ 10 Fehler

First In First Out (FIFO)

- Verdrängt den Datenblock, der sich am längsten im Speicher befindet
- Problem: Nur weil auf einen Datenblock länger nicht mehr zugegriffen wurde, heißt das nicht, dass er nicht in naher Zukunft verwendet wird
- Laszlo Belady zeigte 1969, dass unter ungünstigen Umständen FIFO bei einem größeren Speicher zu mehr Zugriffsfehlern (Miss) führt, als bei einem kleinen Speicher (Belady's Anomalie)
  - Ursprünglich ging man davon aus, dass eine Vergrößerung des Speichers zu weniger oder schlechtestenfalls gleich vielen Zugriffsfehlern führt
- Bis zur Entdeckung von Belady's Anomalie galt FIFO als gute Ersetzungsstrategie

Belady's Anomalie (1969)

Cache-Anfrage:	**1**	**2**	**3**	**4**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
1. Datenblock:	1	1	1	4	4	4	5	5	5	5	5	5
2. Datenblock:		2	2	2	1	1	1	1	1	3	3	3
3. Datenblock:			3	3	3	2	2	2	2	2	4	4

→ 9 Fehler

1. Datenblock:	1	1	1	1	1	1	5	5	5	5	4	4
2. Datenblock:		2	2	2	2	2	2	1	1	1	1	5
3. Datenblock:			3	3	3	3	3	3	2	2	2	2
4. Datenblock:				4	4	4	4	4	4	3	3	3

→ 10 Fehler

Clock

- Für jeden Datenblock wird in der Seitentabelle ein Referenzbit geführt
- Wird ein Datenblock geladen $\Longrightarrow$ Referenzbit $= 0$
- Wird auf einen Datenblock zugegriffen $\Longrightarrow$ Referenzbit $= 1$
- Ein Markierungsbit zeigt auf den zuletzt zugegriffen Datenblock
- Beim Miss wird der Cache ab dem Markierungsbit nach dem ersten Datenblock durchsucht, dessen Referenzbit den Wert 0 hat
  - Dieser Datenblock ersetzt
  - Bei allen bei der Suche durchgesehenen Datenblöcken, bei denen das Referenzbit den Wert 1 hat, wird das Referenzbit auf 0 gesetzt

Cache-Anfrage:	**1**	**2**	**3**	**4**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
1. Datenblock:	$_0 1^X$	$_0 1$	$_0 1$	$_0 4^X$	$_0 4$	$_0 4$	$_0 5^X$	$_0 5$	$_0 5$	$_0 3^X$	$_0 4^X$	$_0 4$
2. Datenblock:		$_0 2^X$	$_0 2$	$_0 2$	$_0 1^X$	$_0 1$	$_0 1$	$_1 1^X$	$_1 1$	$_1 1$	$_0 1$	$_0 5^X$
3. Datenblock:			$_0 3^X$	$_0 3$	$_0 3$	$_0 2^X$	$_0 2$	$_0 2$	$_1 2^X$	$_1 2$	$_0 2$	$_0 2$

$\longrightarrow$ 10 Fehler

Climb

- Datenblöcke werden immer hinten an den Cache angehängt
- Beim Zugriff auf einen Datenblock, steigt dieser eine Ebene nach oben
- Beim Miss wird der letzte Datenblock verdrängt
- Es ist kein Zähler wie bei LFU (Least Frequently Used) nötig
  - Dafür sind Kopiervorgänge nötig

Cache-Anfrage:	**1**	**2**	**3**	**4**	**1**	**2**	**5**	**1**	**2**	**3**	**4**	**5**
1. Datenblock:			1	1	1	2	2	1	2	2	2	2
2. Datenblock:		1	2	2	2	1	1	2	1	1	1	1
3. Datenblock:	1	2	3	4	4	4	5	5	5	3	4	5

→ 8 Fehler

Weitere Ersetzungsstrategien

- **TTL** (Time To Live): Jedem Datenblock wird bei der Erzeugung eine Lebenszeit zugeordnet
  - Ist die TTL überschritten, kann der Datenblock aus dem Cache verdrängt werden
- **Random**: Zufälliger Datenblock wird aus dem Cache verdrängt
- **WS** (Working Set): Alle Datenblöcke, die in einem bestimmten Zeitfenster von einem Prozess verwendet wurden, das sogenannte Working Set, werden verdrängt

Festplatten

- Festplatten sind ca. Faktor 100 preisgünstiger pro Bit als Hauptspeicher und bieten ca. Faktor 100 mehr Kapazität
  - Nachteil: Zugriffe auf Festplatten sind um ca. Faktor 1000 langsamer
- Grund für die geringere **Zugriffsgeschwindigkeit**:
  - Festplatten sind mechanische Geräte die eine oder mehrere mit 4200, 5400, 7200, 10800 oder 15000 Umdrehungen pro Minute rotierende Scheiben enthalten
- Für jede Seite jeder Platte existiert ein Schwungarm mit einem **Schreib-/Lesekopf**, der Bereiche der Scheibenoberfläche unterschiedlich magnetisiert und so die Daten auf die Festplatte schreibt bzw. von ihr liest
  - Zwischen Platte und Kopf ist ein Luftpolster von ca. 20 Nanometern
- Auch Festplatten haben einen Cache, der Schreib- und Lesezugriffe auf den Datenträger puffert
  - Der Festplattencache befindet sich auf der Steuerplatine und ist zwischen 1 und 32 MB groß

Logischer Aufbau von Festplatten (1)

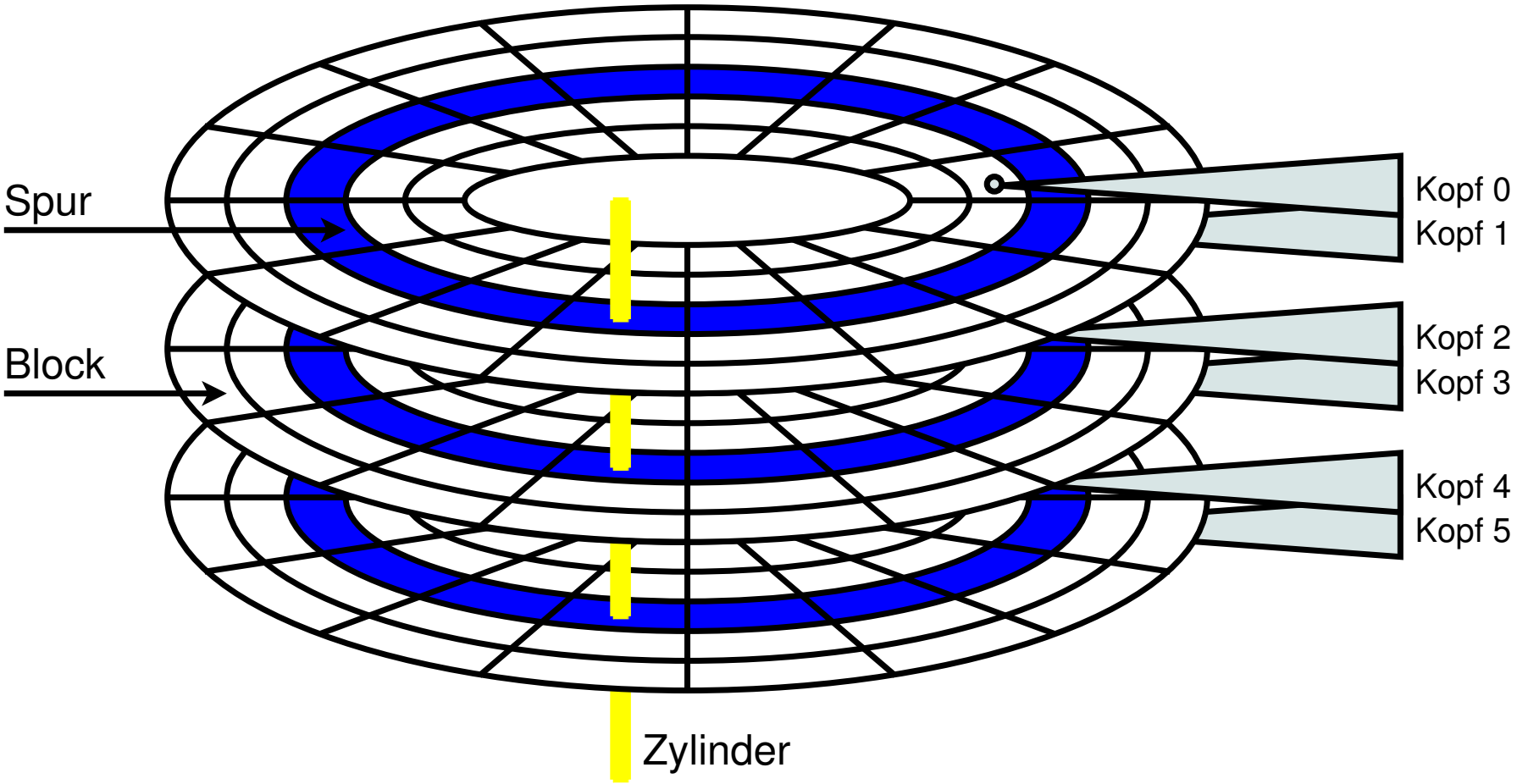
- Die Magnetisierung der Scheiben wird auf kreisförmigen, konzentrischen **Spuren** (*Tracks*) von den Köpfen auf beiden Seiten aufgetragen
- Alle gleichen, also übereinander befindlichen Spuren der einzelnen Plattenoberflächen bezeichnet man als **Zylinder** (*Cylinder*)
- Die Spuren werden in kleine logische Einheiten (Kreissegmente) unterteilt, die man **Blöcke** oder **Sektoren** nennt
  - Typischerweise enthält ein Block 512 Byte Nutzdaten
  - Blöcke sind die kleinsten adressierbaren Einheiten auf Festplatten
  - Müssen Daten in einem Block geändert werden, muss der ganze Block gelesen und neu geschrieben werden
- Heute werden auf Softwareseite **Cluster** angesprochen
  - Cluster sind Verbünde von Blöcken mit fester Größe, z.B. 4 oder 8 KB
  - Bei modernen Betriebssystemen sind Cluster die kleinste Zuordnungseinheit

Offene Festplatte

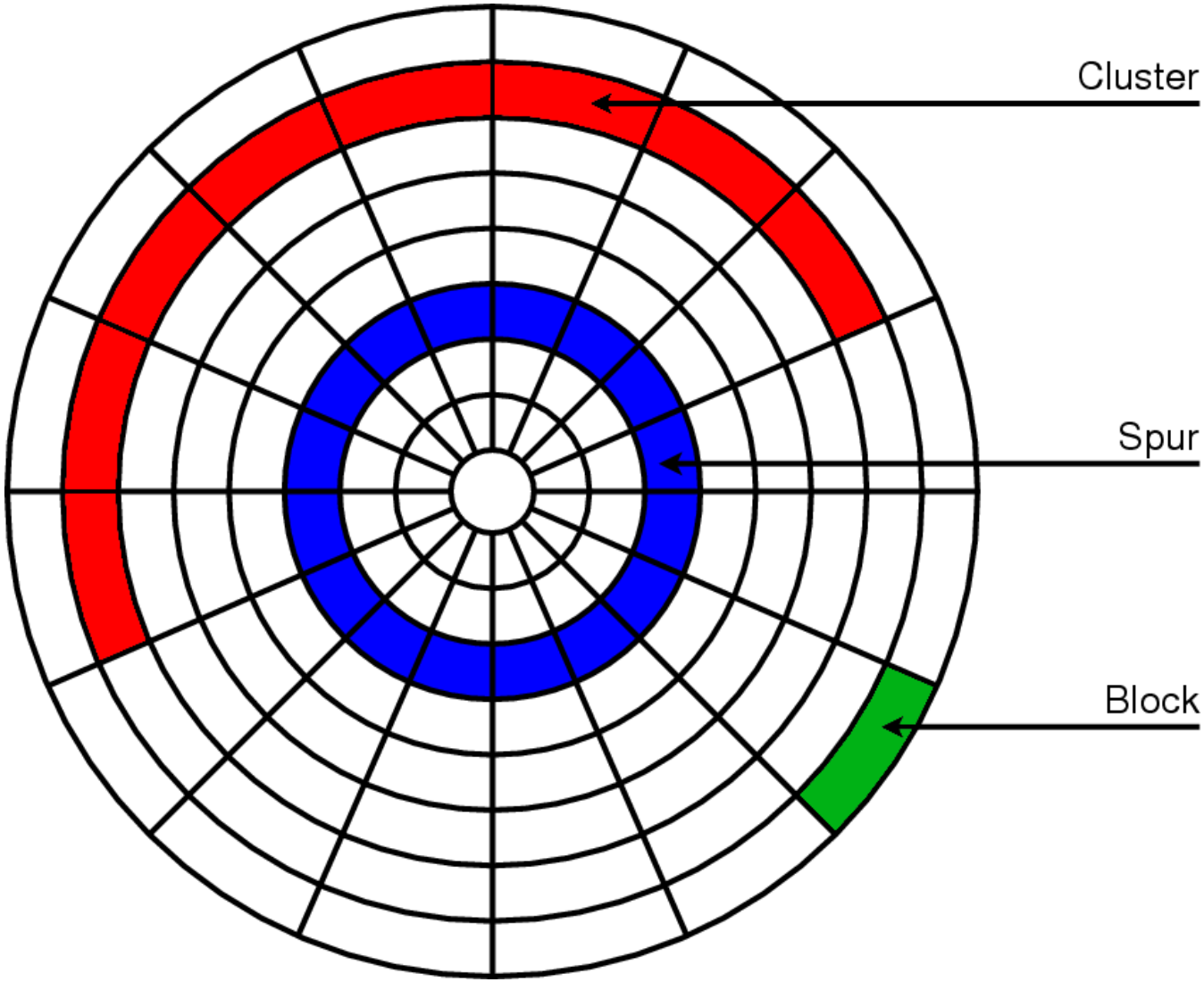
Bildquelle:
`http://www.hitechreview.com`

Logischer Aufbau von Festplatten (2)

- Alle Spuren auf allen Platten bei einer Position des Schwungarms bilden einen Zylinder



Logischer Aufbau von Festplatten (3)



Adressierung der Daten (1)

- Bei Festplatten mit einer Kapazität ≤ 8 GB werden die einzelnen Sektoren mit der sogenannten Zylinder-Kopf-Sektor-Adressierung (**C***ylinder*-**H***ead*-**S***ector-Adressierung*) durchgeführt
- Jeder Sektor lässt sich mit CHS klar lokalisieren und adressieren
- CHS unterliegt aber mehreren Einschränkungen:
  - Die Schnittstelle zwischen IDE und BIOS reserviert nur 16 Bit für die Zylinder (maximal 65.536), 4 Bit für die Köpfe (maximal 16) und 8 Bits für die Sektoren/Spur (maximal 256)
  - Das BIOS hat 10 Bits für die Zylinder zur Verfügung (maximal 1.024), 8 Bits für die Köpfe (maximal 256) und 6 Bit für die Sektoren/Spur (maximal 63, da ab 1 gezählt wird)
- Bei den Grenzen ist der jeweils niedrigere Wert entscheidend
- Darum können alte BIOS-Versionen maximal 504 MB adressieren
  - 1.024 Zylinder * 16 Köpfe * 63 Sektoren/Spur * 512 Byte/Sektor = 528.482.304 Byte
  - 528.482.304 Byte / 1024 / 1024 = 504 MB

Adressierung der Daten (2)

- Spätere BIOS-Versionen stocken beim sogenannten erweiterten CHS per Mapping die Anzahl der Schreib-/Leseköpfe auf 255 auf
- Eine 2,5" oder 3,5" Festplatte hat natürlich nicht 16 oder mehr Köpfe
- Es wird eine andere Geometrie (Zylinder/Köpfe/Sektor) verwendet als die physische Geometrie
- Dadurch sind Kapazitäten bis 7.844 GB möglich
  - 1.024 Zylinder * 255 Köpfe * 63 Sektoren/Spur * 512 Byte/Sektor = 8.422.686.720 Byte
  - 8.422.686.720 Byte / 1.024 / 1.024 / 1.024 = 7,844 GB

Adressierung der Daten (3)

- Seit der Einführung der logischen Blockadressierung (*Logical Block Addressing (LBA)*) 1995 werden alle Sektoren auf der Festplatte von 0 beginnend durchnummeriert
  - Durch LBA wurde das BIOS- und CHS-Korsett umgangen und es kann damit jede derzeit gängige Festplatte voll adressiert werden
- Die Zahlen zu Zylinder, Köpfen und Sektoren auf der Festplatte haben heute nichts mehr mit der tatsächlichen Lage der Sektoren auf der Festplatte zu tun
  - Es handelt sich um eine logische Plattengeometrie, die nur aus Kompatibilitätsgründen existiert
- Aus Kompatibilitätsgründen enthält die CHS-Information bei allen Platten $\geq 8\,\text{GB}$ gemäß ATA-Spezifikation folgende Werte: 16.383 Zylinder, 16 Köpfe (manchmal 15 Köpfe) und 63 Sektoren pro Spur
  - Mehr lässt sich im CHS-Modus nicht ansprechen
- Nur in der LBA-Information offenbaren diese Platten ihre wahre Größe

Berechnung der Größe bei Festplatten

- Auf einer Festplatte befinden sich folgende Informationen:
  - 16.383 Zylinder, 16 Köpfe, 63 Sektoren/Spur
- Kapazität einer Scheibe der Festplatte:
  - 16.383 Zylinder * 63 Sektoren/Spur * 512 Byte/Sektor = 528.450.048 Byte
- Größe einer Spur:
  - 63 Sektoren/Spur * 512 Byte/Sektor = 32.256 Byte
- Die Gesamtkapazität der Festplatte ist:
  - 528.450.048 Byte/Scheibe * 16 Köpfe = 8.455.200.768 Byte
- Teilt man diesen Wert drei mal durch 1024 erhält man die GB der Festplatte:
  - 8.455.200.768 Byte / 1024 / 1024 / 1024 = 7,874 GB
- Die Hersteller teilen den Byte-Wert in der Regel durch 1000
  - Dadurch ergibt sich eine höhere Kapazitätsangabe:
  - 8.455.200.768 Byte / 1000 / 1000 / 1000 = 8,455 GB

CHS-Adressen in LBA-Adressen umrechnen

- CHS-Adressen, bestehend aus einem (c_A, h_A, s_A) Tupel können in LBA-Adressen umgerechnet werden:

$$A = (c_A * h_n * s_n) + (h_A * s_n) + s_A - 1$$

$$\begin{aligned} A &= \text{LBA-Adresse} \\ c_n &= \text{Anzahl der Zylinder} \\ h_n &= \text{Anzahl der Köpfe} \\ s_n &= \text{Anzahl der Sektoren/Spur} \\ (c_A, h_A, s_A) &= \text{CHS-Adresse} \end{aligned}$$

Zugriffszeit bei Festplatten

- Die Zugriffszeit ist ein wichtiges Kriterium für die Geschwindigkeit
- 2 Faktoren sind für die Zugriffszeit einer Festplatte verantwortlich
  1. **Suchzeit** (*Average Seek Time*)
     - Die Zeit, die der Schwungarm braucht, um eine Spur zu erreichen
     - Die Suchzeit moderner Festplatten ist zwischen 5 und 15 ms
  2. **Zugriffsverzögerung durch Umdrehung** (*Average Rotational Latency Time*)
     - Verzögerung der Drehgeschwindigkeit bis der Schreib-/Lesekopf den gewünschten Block erreicht
     - Dieser Wert ist ausschließlich von der Drehgeschwindigkeit der Scheiben abhängig
     - Die Verzögerung durch die Drehgeschwindigkeit bei modernen Festplatten liegt zwischen 2 und 7,1 ms

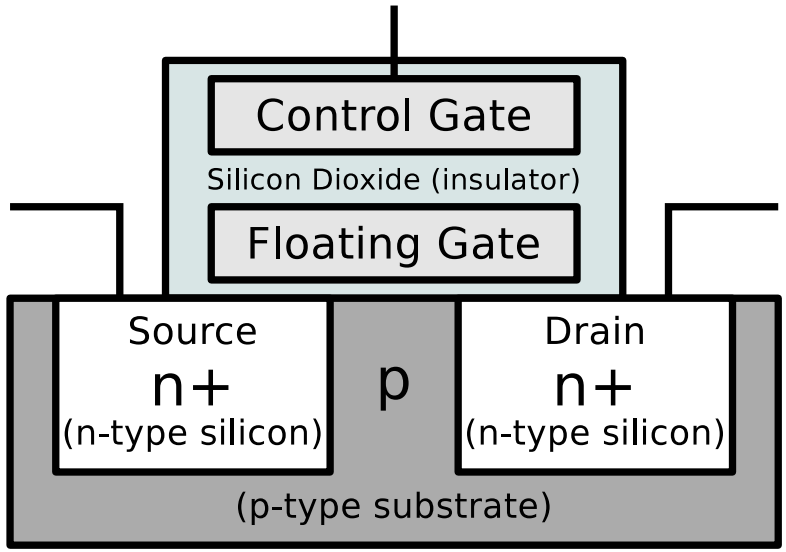
$$\text{Zugriffsverzögerung durch Umdrehung} = \frac{30.000}{\text{Drehgeschwindigkeit}}$$

Solid State Drives

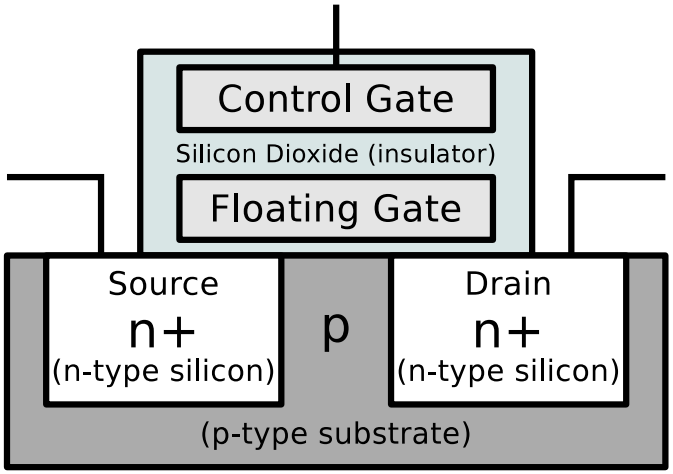
- Solid State Drives (SSDs), manchmal auch fälschlicherweise als Solid State Disks (SSDs) bezeichnet, sind Halbleiterlaufwerke
  - Datenspeicherung findet auf Halbleiterspeicherbausteinen statt
  - Keine beweglichen Teile enthalten
- Vorteile
  - Kurze Zugriffszeiten
  - Geringer Energieverbrauch
  - Keine Geräuschentwicklung
  - Mechanische Robustheit
  - Geringes Gewicht
  - Position der Daten ist irrelevant $\Longrightarrow$ Defragmentieren unnötig
- Nachteile
  - Höherer Preis im Vergleich zu Festplatten gleicher Kapazität
  - Sicheres Löschen bzw. Überschreiben ist schwierig
  - Eingeschränkte Anzahl an Schreib-/Löschzyklen

Arbeitsweise von Flash-Speicher

- Daten werden in Flash-Speicher als elektrische Ladungen gespeichert
- Im Gegensatz zum Hauptspeicher ist kein Strom nötig, um die Daten im Speicher zu halten
- Eine Flash-Speicherzelle besteht aus 3 Schichten Silizium
  - Gate
  - Drain (Elektrode)
  - Source (Elektrode)

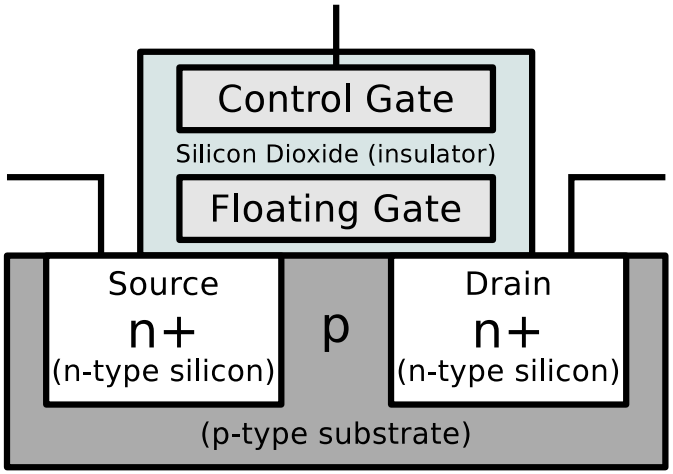


Daten in Flash-Speicherzellen schreiben



- Hinter dem Control-Gate liegt das Floating-Gate (die Ladungsfalle)
  - In der Ladungsfalle werden die Daten gespeichert
  - Das Floating-Gate ist komplett von einem Isolator umgeben
  - Üblicherweise bleibt die Ladung in der Ladungsfalle über Jahre stabil
- Der quantenmechanische Prozess des Fowler-Nordheim-Tunnels ermöglicht es Elektronen durch den Nichtleiter zu wandern
- Durch das Ansetzen einer hohen Spannung (10–20V) werden einige Elektronen von Source durch den Isolator in das Floating-Gate getunnelt

Daten in Flash-Speicherzellen löschen



- Um den Inhalt einer Flash-Speicherzelle zu löschen, wird eine hohe hohe negative Löschspannung angesetzt
- Die isolierende Schicht, das das Floating-Gate umgibt, leidet bei jedem Löschvorgang
  - Irgendwann ist die isolierende Schicht nicht mehr ausreichend, um die Ladung im Floating-Gate zu halten
  - Aus diesem Grund sind bei Flash-Speicher nur eine eingeschränkte Anzahl Schreib-/Löschzyklen möglich

Arbeitsweise von Flash-Speicher

- In einem Flash-Speicher sind die Speicherzellen in Gruppen zu Seiten und Blöcken angeordnet
  - Je nach dem Aufbau eines Flash-Speichers ist immer eine feste Anzahl an Seiten zu einem Block zusammengefasst
- Schreib- und Löschoperationen können nur für komplette Blöcke durchgeführt werden
  - Aus diesem Grund sind Löschvorgänge bei Flash-Speicher aufwendiger als Leseoperationen
- Wenn eine Seite verändert werden soll, die bereits Daten enthält, muss der gesamte Block zuerst gelöscht werden
  - Dafür muss der gesamte Block in einen Pufferspeicher kopiert werden
  - Im Pufferspeicher werden die Daten verändert
  - Danach wird der Block im Flash-Speicher gelöscht
  - Anschließend wird der komplette Block in die gelöschten Speicherzellen geschrieben

Unterschiede beim Flash-Speicher

- Es existieren 2 Arten von Flash-Speicher:
  - **NOR-Speicher**
  - **NAND-Speicher**
- Das Schaltzeichen bezeichnet die interne Verbindung der Speicherzellen
- Die interne Verbindung beeinflusst Kapazität und Zugriffsgeschwindigkeit

NOR-Speicher

- Alle Speicherzellen sind mit eigenen Verbindungsleitungen angeschlossen
  $\Longrightarrow$ Wahlfreier Zugriff
  - Aufbau ist dementsprechend komplex und kostspielig
- Typische Blockgrößen: 64, 128 oder 256 KB
- Üblicherweise geringe Kapazitäten (bis zu 32 MB)
- Primär im industriellen Umfeld zu finden

NAND-Speicher

- Speicherzellen sind zu Seiten zusammengefasst
  - Typische Seitengröße: 2.048 bis 4.096 Byte
- Mehrere Seiten umfassen einen Block
  - Typische Blockgröße: 128, 256, 256 oder 512 KB
  - Jeder Block ist mit einer Datenverbindung angeschlossen
    - Daher kein wahlfreier Zugriff
- Benötigt wegen der reduzierten Anzahl an Datenleitungen nur ca. halb so viel Platz wie NOR-Speicher
- Herstellung ist viel preisgünstiger im Vergleich zu NOR-Flash-Speicher
- USB-Sticks und SSDs bestehen heute immer aus NAND-Speicher
- Es existieren 2 Arten von NAND-Flash-Speicher
  - **Single-Level Cell (SLC)**
  - **Multi-Level Cell (MLC)**

Single-Level Cell (SLC) und Multi-Level Cell (MLC)

- Jede SLC-Speicherzelle kann 1 Bit speichern
- MLC-Speicherzellen können 2 oder 4 Bit speichern
- SLC-Speicher ist teurer
- SLC-Speicher bietet eine höhere Schreibgeschwindigkeit
- SLC-Speicher hat eine höhere Lebensdauer (mehr Schreib-/Löschzyklen)
  - SLC-Speicher: Üblicherweise 100.000 bis 300.000 Schreib-/Löschzyklen
  - MLC-Speicher: Üblicherweise ca. 10.000 Schreib-/Löschzyklen
- Es existieren auch Speicherzellen, die mehrere Millionen Schreib-/Löschzyklen ermöglichen
- Wear-Level-Algorithmen verteilen Schreib-/Löschzugriffe gleichmäßig
- Dateisysteme, die speziell für Flash-Speicher ausgelegt sind, und Schreibzugriffe minimieren sind u.a. JFFS, JFFS2, YAFFS und LogFS
  - JFFS enthält eigene Wear-Level-Algorithmen
    - Das ist bei eingebetteten Systemen häufig notwendig, wo Flash-Speicher direkt angeschlossen wird

Zugriffszeiten bei Festplatten

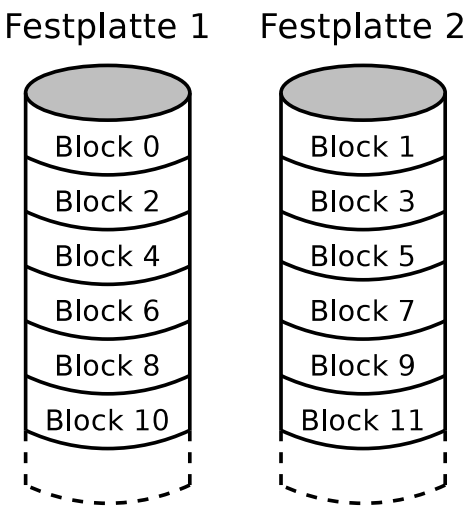
- In den letzten 40 Jahren ist die Geschwindigkeit von Prozessoren, Cache und Hauptspeicher deutlich schneller gewachsen als die Zugriffsgeschwindigkeit der Festplatten:
  - **Festplatten**
    1973: IBM 3340, 30 MB Kapazität, 30 ms Zugriffszeit
    1989: Maxtor LXTI00S, 96 MB Kapazität, 29 ms Zugriffszeit
    1998: IBM DHEA-36481, 6 GB Kapazität, 16 ms Zugriffszeit
    2006: Maxtor STM320820A, 320 GB Kapazität, 14 ms Zugriffszeit
    2011: Western Digital WD30EZRSDTL, 3 TB Kapazität, 8 ms Zugriffszeit
  - **Prozessoren**
    1971: Intel 4004, 740 kHz Taktfrequenz
    2007: AMD Opteron Santa Rosa F3, 2,8 GHz Taktfrequenz
    2010: Core i7 980X Extreme (6 Cores), 3,33 Ghz Taktfrequenz
- Der Abstand vergrößert sich in Zukunft weiter
- Kann man die Zugriffszeit und Datensicherheit bei Festplatten erhöhen?

Redundant Array of independent Disks (RAID)

- Die Geschwindigkeit der Festplatten lässt sich nicht beliebig verbessern
- Festplatten bestehen aus beweglichen Teilen
- Physikalische und materielle Grenzen müssen akzeptiert werden
- Eine Möglichkeit, die gegebenen Beschränkungen im Hinblick auf Geschwindigkeit, Kapazität und Datensicherheit zu umgehen, ist das gleichzeitige Verwenden mehrerer Komponenten
- Ein RAID-System besteht aus mehreren Festplatten
  - Diese werden vom Benutzer und den Prozessen als eine einzige große Festplatte wahrgenommen
- Die Daten werden über die Festplatten eines RAID-Systems verteilt
- Wie die Daten verteilt werden, wird durch das RAID-Level spezifiziert
- Die gebräuchlichsten RAID-Level sind RAID 0, RAID 1 und RAID 5

RAID 0 – Striping – Beschleunigung ohne Redundanz

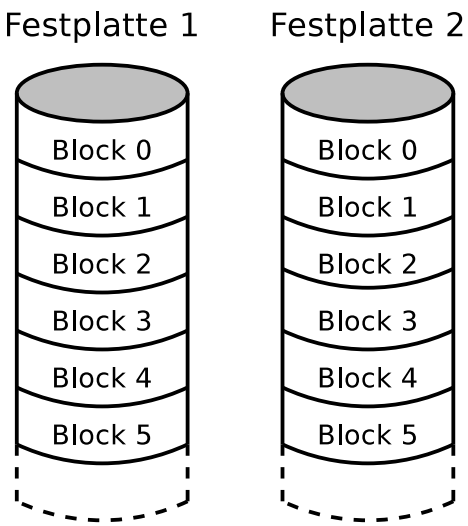
- RAID 0 ist eigentlich kein echtes RAID, da es keine Redundanz vorsieht
- Es wird nur die Datentransferrate gesteigert
- Aufteilung der Festplatten in zusammenhängende Blöcke gleicher Größe
- Sind die Ein-/Ausgabeaufträge groß genug ($>$ 4 oder 8 KB), können die Zugriffe auf den Verbund parallel auf mehreren oder allen Festplatten durchgeführt werden



- Fällt eine Festplatte aus, können die Daten nicht mehr vollständig rekonstruiert werden
  - Nur kleinere Dateien, die vollständig auf den verbliebenen Festplatten gespeichert sind, können gerettet werden
- RAID 0 eignet sich nur, wenn die Sicherheit der Daten bedeutungslos ist oder eine andere geeignete Form der Datensicherung vorhanden ist

RAID 1 – Mirroring – Spiegelung

- Mindestens zwei Festplatten gleicher Kapazität enthalten exakt die gleichen Daten
  - Sind die Festplatten unterschiedlich groß, bietet ein Verbund mit RAID 1 höchstens die Kapazität der kleinsten Festplatte
- Der Ausfall einer Festplatte führt nicht zu Datenverlust, da die übrigen Festplatten die identischen Daten vorhalten
- Zum Totalverlust kommt es nur beim Ausfall aller Festplatten

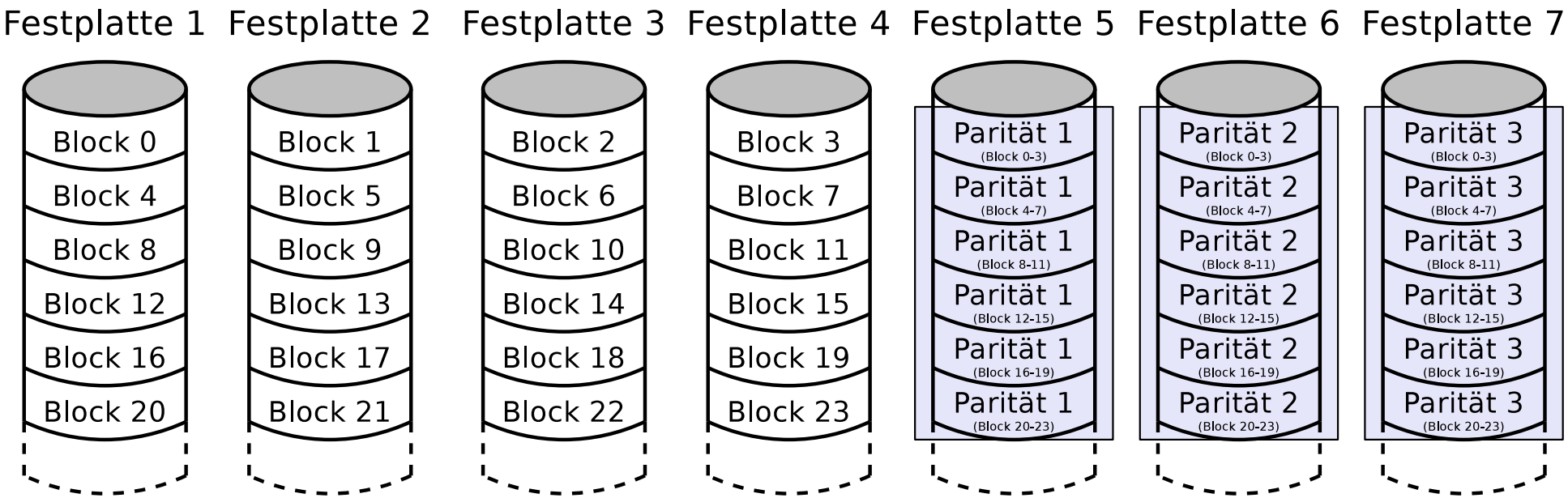


- Jede Datenänderung wird auf allen beteiligten Festplatten geschrieben
- RAID 1 ist kein Ersatz für Datensicherung
  - Fehlerhafte Dateioperationen oder Virenbefall finden auf allen Festplatten statt
- Die Lesegeschwindigkeit kann durch intelligente Verteilung der Zugriffe auf die angeschlossenen Platten gesteigert werden

RAID 2 – Bit-Level Striping mit Hamming-Code-Fehlerkorrektur

- Die Nutzdaten werden hierbei Blöcke gleicher Größe zerlegt und mittels eines Hamming-Codes auf größere Blöcke abgebildet
  - Die Bits, die Potenzen von 2 sind (1, 2, 4, 8, 16, usw.) sind Prüfbits
- Die einzelnen Bits des Hamming-Codeworts werden über die Platten verteilt $\Longrightarrow$ Datendurchsatz wird durch RAID 2 gesteigert
- Ein mögliches Szenario:
  - 7 Festplatten im Verbund
  - Davon 4 Platten für die Nutzdaten und 3 für die Paritätsinformationen
- Der kleinste RAID-2-Verbund benötigt 3 Festplatten und entspricht einem RAID 1 mit zweifacher Spiegelung
- In der Realität besteht ein RAID 2 aus mindestens 10 Festplatten
- Wurde nur bei Großrechnern verwendet und spielt heute keine Rolle mehr

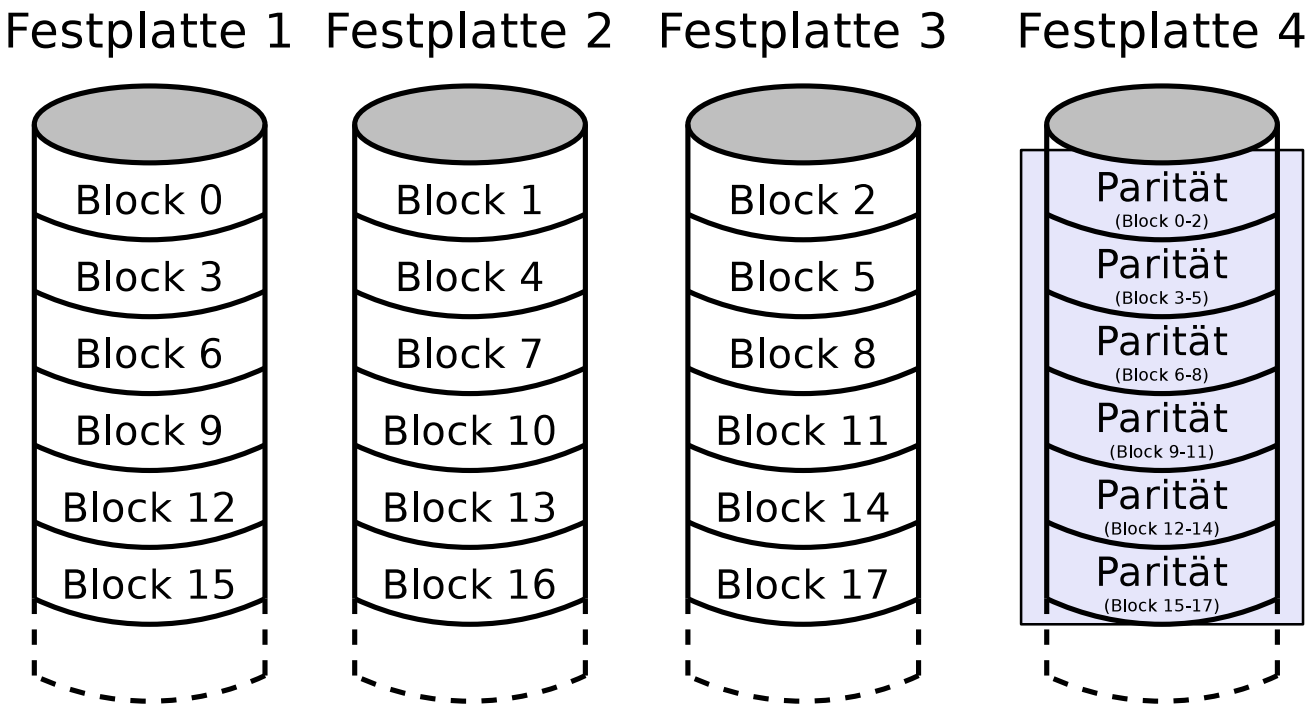
Prinzip von RAID 2



RAID 3 – Byte-Level Striping mit Paritätsinformationen

- Speicherung der Nutzdaten auf einer oder mehreren Datenplatten
- Speicherung der Paritätsinformationen auf einer Paritätsplatte
- Berechnung der Paritätsinformationen
  - Die Bits der Datenplatten werden zusammengezählt und die errechnete Summe darauf untersucht, ob sie eine gerade oder ungerade ist
  - Gerade Summe wird auf der Paritätsplatte mit Bit 0 gekennzeichnet
  - Ungerade Summe wird mit dem Bit-Wert 1 gekennzeichnet
- Die Paritätsplatte enthält nur die Summeninformationen
- Man kann beliebig viele Datenplatten verwenden und braucht für die Paritätsinformationen trotzdem nur eine einzige Platte
- Die Paritätsplatte wird bei jeder Operation benötigt $\Longrightarrow$ Flaschenhals
  - Auf die Paritätsplatte wird bei jeder Schreiboperation zugegriffen
- Bei RAID 2 mit 2 Festplatten hat die Paritätsplatte den gleichen Inhalt wie die Datenplatte $\Longrightarrow$ Identisch zu RAID-1 mit 2 Festplatten
- Ist vom Markt verschwunden und wurde durch RAID 5 ersetzt

Prinzip von RAID 3

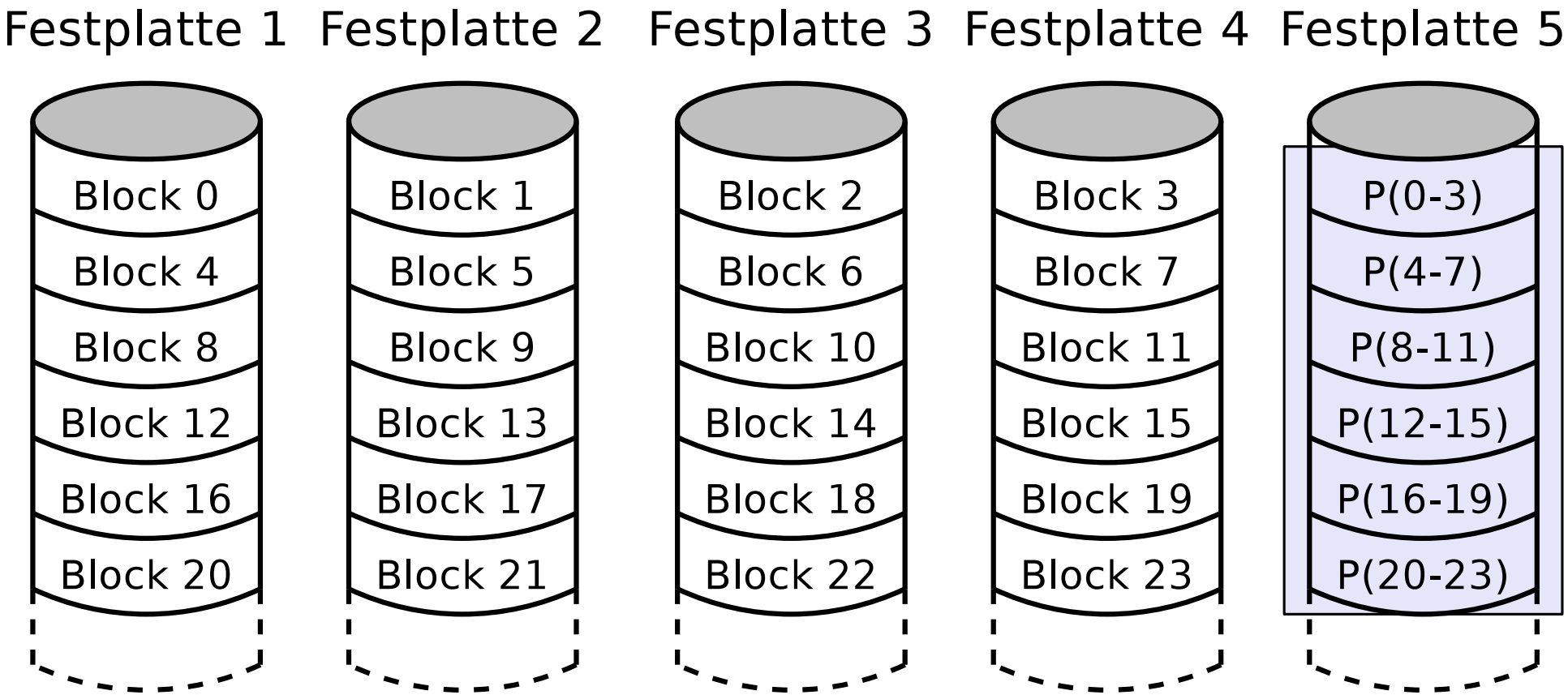


Datenplatten		Summe		gerade/ungerade		Paritätsplatte
Bits sind $0+0+0$	$\Longrightarrow$	0	$\Longrightarrow$	Summe ist gerade	$\Longrightarrow$	Summen-Bit 0
Bits sind $1+0+0$	$\Longrightarrow$	1	$\Longrightarrow$	Summe ist ungerade	$\Longrightarrow$	Summen-Bit 1
Bits sind $1+1+0$	$\Longrightarrow$	2	$\Longrightarrow$	Summe ist gerade	$\Longrightarrow$	Summen-Bit 0
Bits sind $1+1+1$	$\Longrightarrow$	3	$\Longrightarrow$	Summe ist ungerade	$\Longrightarrow$	Summen-Bit 1
Bits sind $0+1+0$	$\Longrightarrow$	1	$\Longrightarrow$	Summe ist ungerade	$\Longrightarrow$	Summen-Bit 1
Bits sind $0+0+1$	$\Longrightarrow$	1	$\Longrightarrow$	Summe ist ungerade	$\Longrightarrow$	Summen-Bit 1

RAID 4 – Block-Level Striping mit Paritätsinformationen

- Speicherung der Nutzdaten auf einer oder mehreren Datenplatten
- Speicherung der Paritätsinformationen auf einer Paritätsplatte
- Unterschied zu RAID 3:
  - Die Einheiten, die geschrieben werden, sind größere Datenblöcke (**Chunks**) und nicht einzelne Bytes
- Die Paritätsplatte wird bei jeder Operation benötigt $\Longrightarrow$ Flaschenhals
  - Auf die Paritätsplatte wird bei jeder Schreiboperation zugegriffen
  - Die Paritätsplatte fällt häufiger aus
  - Lösung: RAID 5
- Anstatt RAID 4 wird fast immer RAID 5 verwendet
- Die Firma NetApp verwendet in ihren NAS-Servern RAID 4
  - z.B. NetApp FAS2020, FAS2050, FAS3040, FAS3140, FAS6080

Prinzip von RAID 4

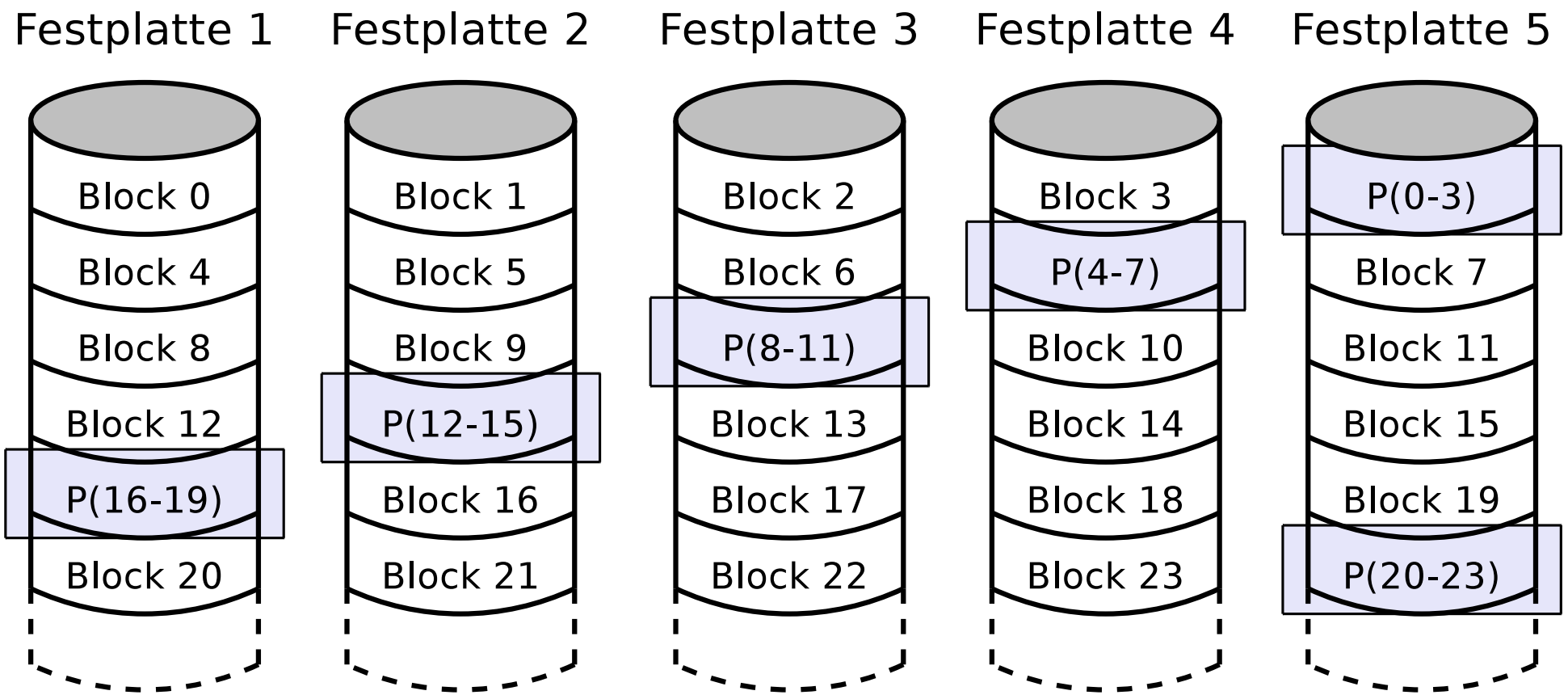


- P(16-19) = Block 16 XOR Block 17 XOR Block 18 XOR Block 19

RAID 5 – Block-Level Striping mit verteilten Paritätsinformationen

- Bietet erhöhten Datendurchsatz und höhere Datensicherheit
- Ein Verbund besteht aus mindestens 3 Festplatten
- Nutzdaten werden wie bei RAID 0 auf alle Festplatten verteilt
- Es werden Paritätsinformationen der Nutzdaten berechnet, mit denen beim Ausfall maximal einer Festplatte die Nutzdaten vollständig rekonstruiert werden können
  - Die Berechnung der Paritätsinformationen beim Ausfall einer Platte kann abhängig von der Größe des Verbunds sehr lange dauern
    - Während dieser Zeit ist der Verbund ungeschützt
- Die Berechnung der Paritätsinformationen durch XOR erfordert zusätzliche Rechenleistung bei Schreibzugriffen
- Durch die Verteilung der Paritätsinformationen auf alle Festplatten (*Rotating Parity*) wird verhindert, dass eine einzelne Paritätsplatte zu einem möglichen Engpass (Flaschenhals) wird
- Die Nettokapazität ist $n - 1$, wobei n die Anzahl der Festplatten ist

Prinzip von RAID 5

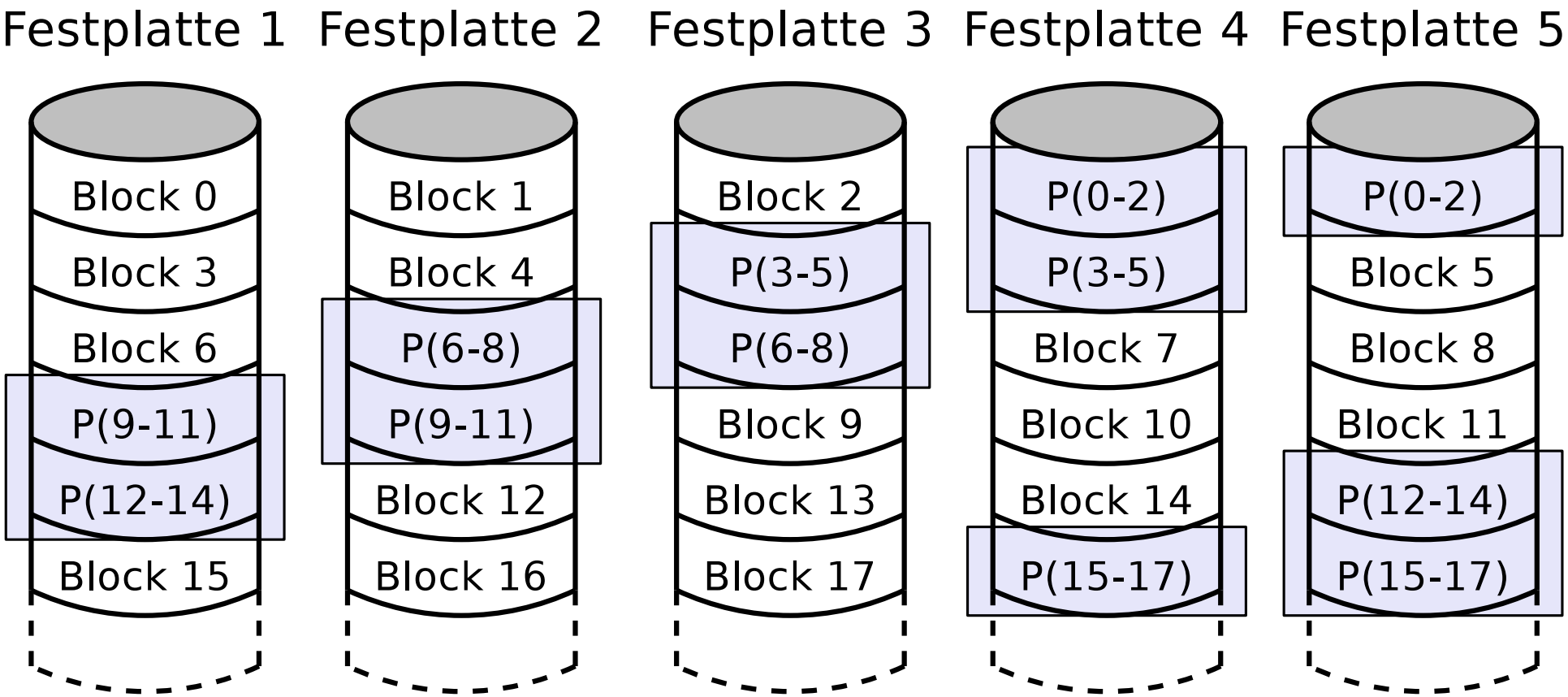


- P(16-19) = Block 16 XOR Block 17 XOR Block 18 XOR Block 19

RAID 6 – Block-Level Striping mit doppelt verteilten Paritätsinformationen

- Funktioniert ähnlich wie RAID 5
- Verkraftet aber den gleichzeitigen Ausfall von bis zu zwei Festplatten
- RAID 6 ist Striping mit doppelten, auf Block-Level verteilten Paritätsinformationen
- Im Gegensatz zu RAID 5 gibt es mehrere mögliche Implementierungsformen, die sich insbesondere in der Schreibleistung und dem Rechenaufwand unterscheiden
  - Bessere Schreibleistung wird durch erhöhten Rechenaufwand erkauft
  - Im einfachsten Fall wird eine zusätzliche XOR-Operation berechnet
- Im Gegensatz zu RAID 5 ist die Verfügbarkeit höher, aber der Datendurchsatz insgesamt niedriger

Prinzip von RAID 6



- Das Beispiel ist nur eine mögliche Realisierung von RAID 6

Übersicht über die RAID-Level

RAID	n (Anzahl Festplatten)	k (Netto-kapazität)	Ausfall-sicherheit	Leistung (Lesen)	Leistung (Schreiben)
0	≥ 2	n	0 (keine)	$n * X$	$n * X$
1	≥ 2	Kapazität der kleinsten Platte	$n-1$ Platten	$n * X$	X
2	≥ 3	$n - [\log_2 n]$	1 Platte	variabel	variabel
3	≥ 3	$n-1$	1 Platte	$(n-1) * X$	$(n-1) * X$
4	≥ 3	$n-1$	1 Platte	$(n-1) * X$	$(n-1) * X$
5	≥ 3	$n-1$	1 Platte	$(n-1) * X$	$(n-1) * X$
6	≥ 4	$n-2$	2 Platten	$(n-2) * X$	$(n-2) * X$

- X ist die Leistung einer einzelnen Platte beim Lesen bzw. Schreiben
- Die maximale theoretisch mögliche Leistung wird häufig vom Controller bzw. der Rechenleistung des Hauptprozessors eingeschränkt

RAID-Kombinationen

- RAID 0, 1 und 5 finden die weitaus größte Verwendung
- Zusätzlich zu den bekannten RAID-Standards (*Leveln*) gibt es noch verschiedene RAID-Kombinationen
  - Dabei wird ein RAID nochmals zu einem zweiten RAID zusammengefasst
- Beispiel: aus mehreren RAID 0-Verbünden wird ein RAID 5-Array gebildet $\Longrightarrow$ RAID 05 (0+5)
- Weitere Beispiele
  - RAID 00: Großes RAID 0 mit mindestens 4 Festplatten
  - RAID 01: RAID 1, das aus mehreren RAID 0 besteht
  - RAID 10: RAID 0 über mehrere RAID 1
  - RAID 03 bzw. RAID 30: RAID 3 über mehrere RAID 0
  - RAID 05: RAID 5, das aus mehreren RAID 0 besteht
  - RAID 15: RAID 5, das aus mindestens drei RAID 1 besteht
  - RAID 50: RAID 0, das aus mehreren RAID 5 besteht
  - RAID 51: RAID 1, das aus mehreren RAID 5 besteht
  - RAID 60: RAID 0, das aus mehreren RAID 6 besteht

Hardware-RAID, Host-RAID und Software-RAID

- **Hardware-RAID**
  - Ein RAID-Controller kümmert sich um die Zusammenarbeit den Festplatten
- **Host-RAID**
  - Es werden entweder preiswerte RAID-Controller oder einfach der Chipsatz auf dem Mainboard eingesetzt
  - Berechnung der Paritätsinformationen übernimmt die CPU
- **Software-RAID**
  - Moderne Betriebssysteme ermöglichen das Zusammenschließen von Festplatten zu einem RAID auch ohne RAID-Controller
    - Linux und Windows können RAID 0, RAID 1 und RAID 5 bereitstellen
    - MacOS X kennt RAID 0 und RAID 1
  - Vorteile: Preiswerter als Hardware-RAID und flexibler als Host-RAID
  - Nachteile: Betriebssystemabhängigkeit und zusätzliche Belastung der CPU

Nächste Vorlesung

Nächste Vorlesung:

17.11.2011