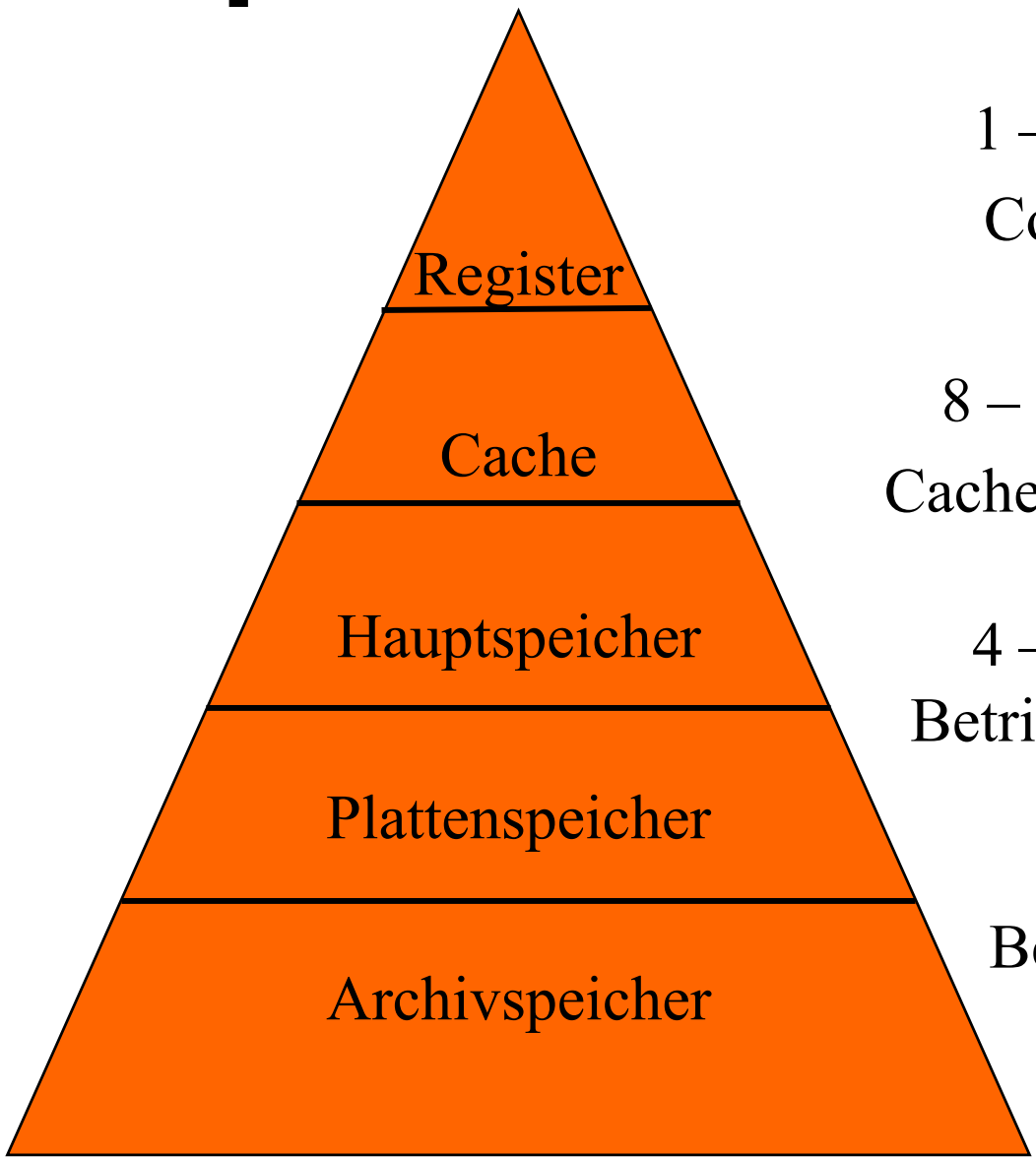
Physische Datenorganisation

- Speicherhierarchie
- Hintergrundspeicher / RAID
- (B-Bäume
- Hashing
- R-Bäume)

Kapitel 7

Überblick: Speicherhierarchie



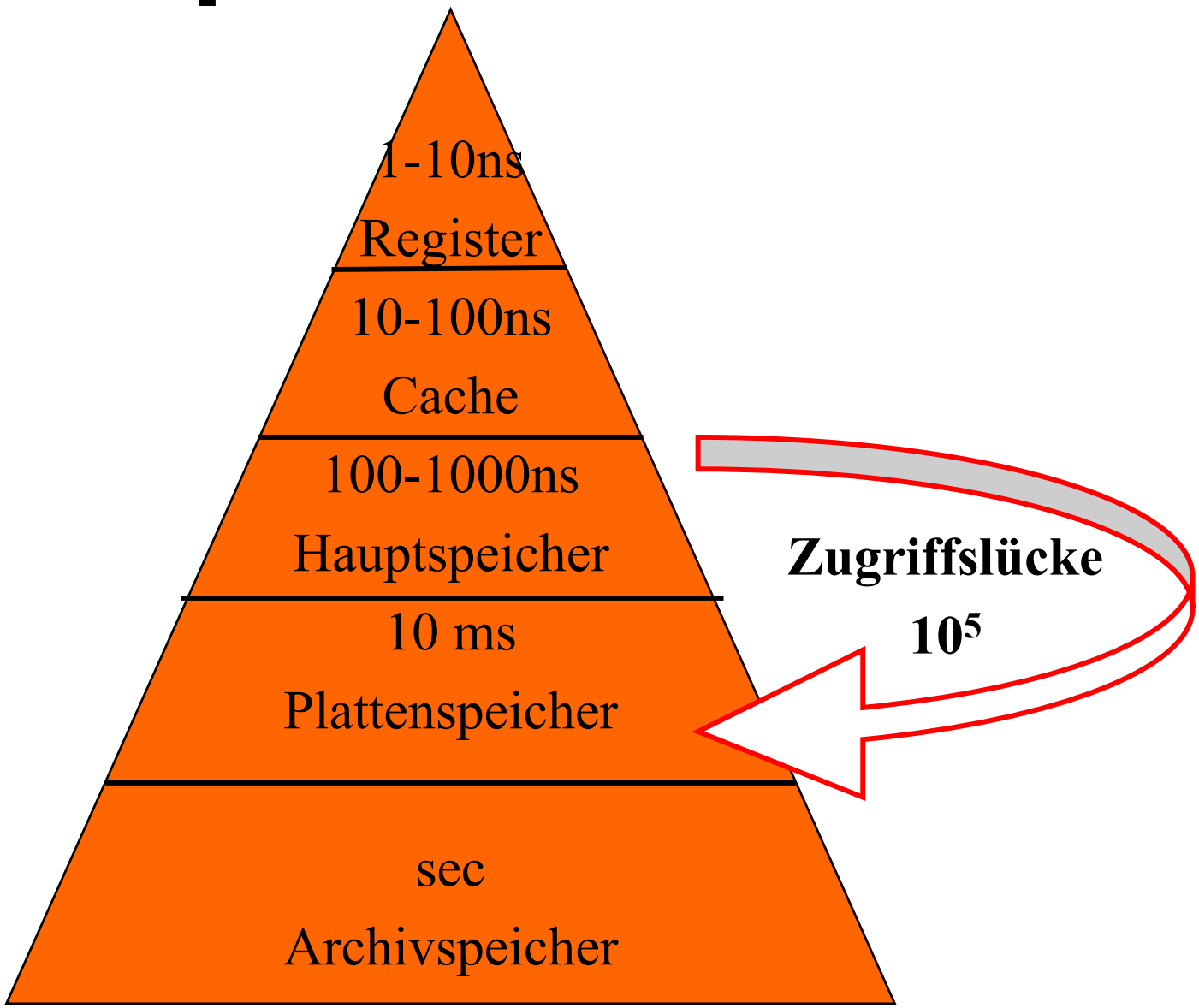
1 – 8 Byte
Compiler

8 – 128 Byte
Cache-Controller

4 – 64 KB
Betriebssystem

Benutzer

Überblick: Speicherhierarchie



Überblick: Speicherhierarchie

Literaturrecherche

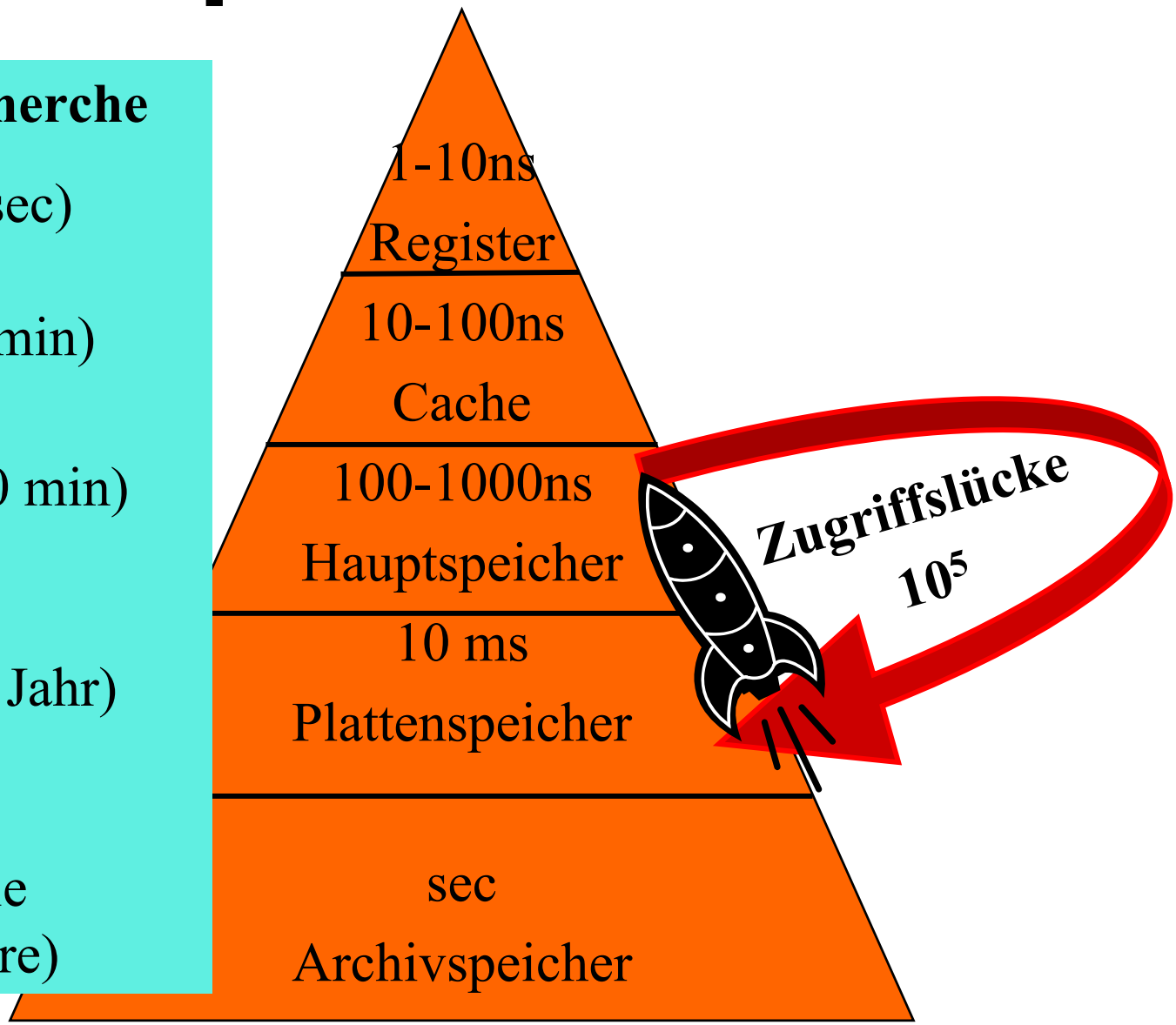
Kopf (30 sec)

Internet (5 min)

Uni-Bibl. (50 min)

Fernleihe (1 Jahr)

Archäologie
(1000 Jahre)



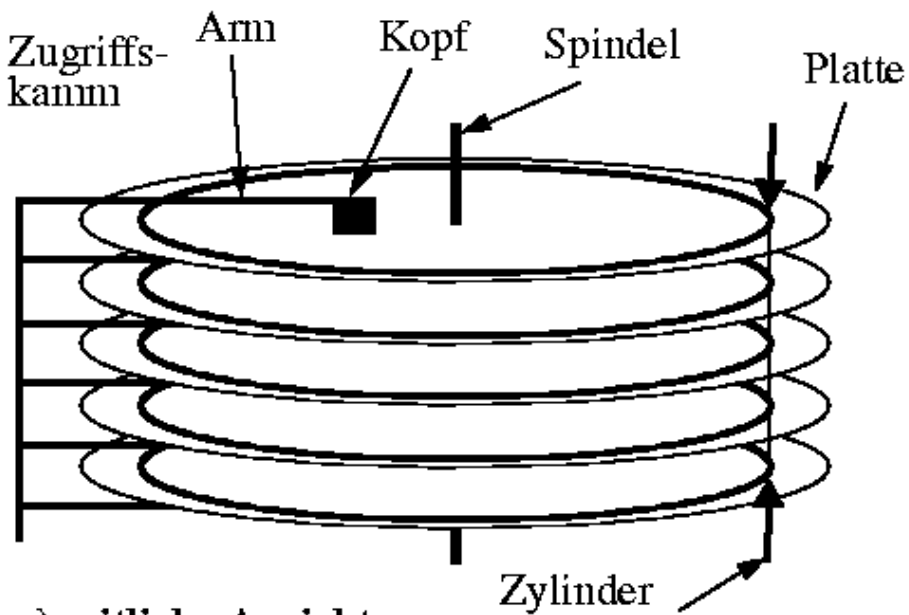
Magnetplattenspeicher

Aufbau

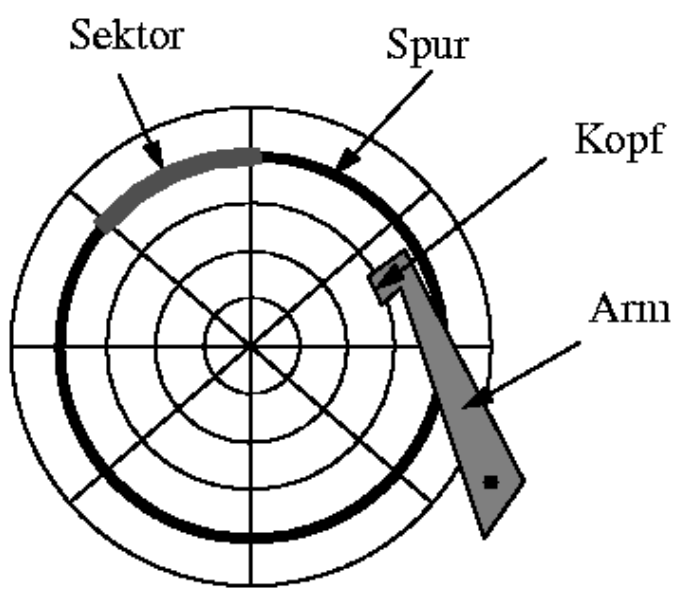
- mehrere gleichförmig rotierende Platten, für jede Plattenoberfläche ein Schreib-/Lesekopf
- jede Plattenoberfläche ist eingeteilt in Spuren
- die Spuren sind formatiert als Sektoren fester Größe (Slots)
- Sektoren (typischerweise 1 - 8 KB) sind die kleinste Schreib-/Leseeinheit auf einer Platte

Adressierung

- Zylindernummer, Spurnummer, Sektornummer
- jeder Sektor speichert selbstkorrigierende Fehlercodes; bei nicht behebbaren Fehlern erfolgt automatische Abbildung auf Ersatzsektoren



a) seitliche Ansicht



b) Draufsicht

Magnetplatten: Technische Merkmale

Merkmal / Magnetplattentyp	typische Werte 1998	IBM 3390 (1990)	IBM 3380 (1985)	IBM 3330 (1970)
t_{smin} = Zugr.bewegung(Min)	1 ms	k. A.	2 ms	10 ms
t_{sav} = " (Mittel)	8 ms	12.5 ms	16 ms	30 ms
t_{smax} = " (Max.)	16 ms	k. A.	29 ms	55 ms
t_r = Umdrehungszeit	6 ms	14.1.ms	16.7 ms	16.7 ms
T_{cap} = Spurkapazität	100 KB	56 KB	47 KB	13 KB
T_{cyl} = #Spuren pro Zyl.	20	15	15	19
N_{dev} = #Zylinder	5000	2226	2655	411
u = Transferrate	15 MB/s	4.2 MB/s	3 MB/s	0. 8 MB/s
Nettokapazität	10 GB	1.89 GB	1.89 GB	0.094 GB

Typische Werte in 2000:

- 5 - 100 GB Kapazität, 10 - 30 MB/s
- 2 - 6 ms Umdrehungszeit, 5 - 10 ms seek
- 20 $ / GB (SCSI-Platten) bzw. 7 $ / GB (IDE-Platten)

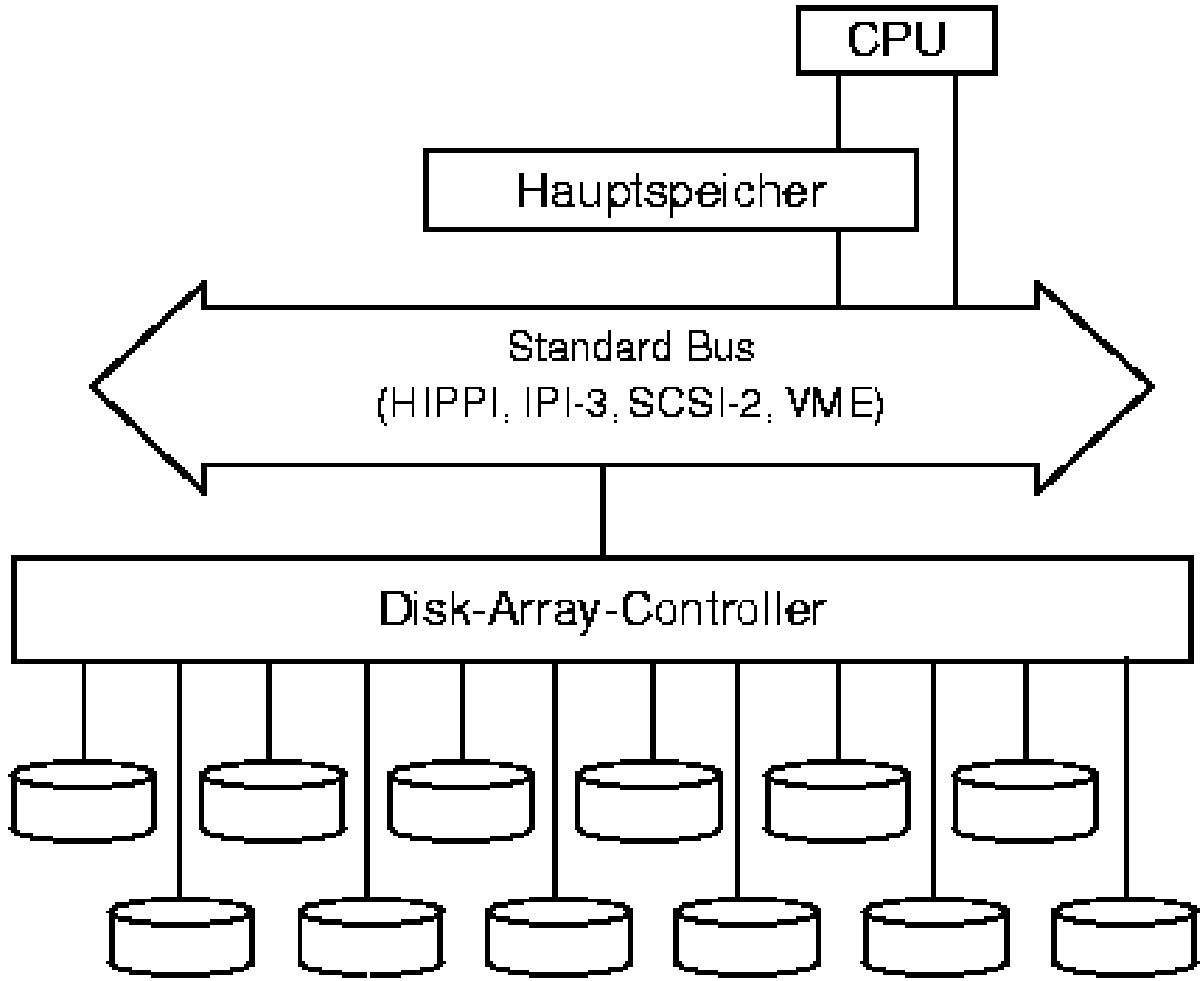
Lesen von Daten von der Platte

- Seek Time: Arm positionieren
  - 5ms
- Latenzzeit: ½ Plattenumdrehung (im Durchschnitt)
  - 10000 Umdrehungen / Minute
  - ➔ Ca 3ms
- Transfer von der Platte zum Hauptspeicher
  - 100 Mb /s ➔ 15 MB/s

Random versus Chained I/O

- 1000 Blöcke à 4KB sind zu lesen
- Random I/O
  - Jedesmal Arm positionieren
  - Jedesmal Latenzzeit
  - ➔ 1000 * (5 ms + 3 ms) + Transferzeit von 4 MB
  - ➔ > 8000 ms + 300ms → 8.3 s
- Chained I/O
  - Einmal positionieren, dann „von der Platte kratzen“
  - ➔ 5 ms + 3ms + Transferzeit von 4 MB
  - ➔ 8ms + 300 ms → 0.38 s
- Also ist chained I/O **ein bis zwei Größenordnungen schneller** als random I/O.
- Ist bei Datenbank-Algorithmen unbedingt zu beachten !

Disk Arrays ➔ RAID-Systeme



Fehlertoleranz

"The Problem with Many Small Disks: Many Small Faults"

Disk-Array mit N Platten: ohne Fehlertoleranzmechanismen N-fach erhöhte Ausfallwahrscheinlichkeit
=> System ist unbrauchbar

Begriffe

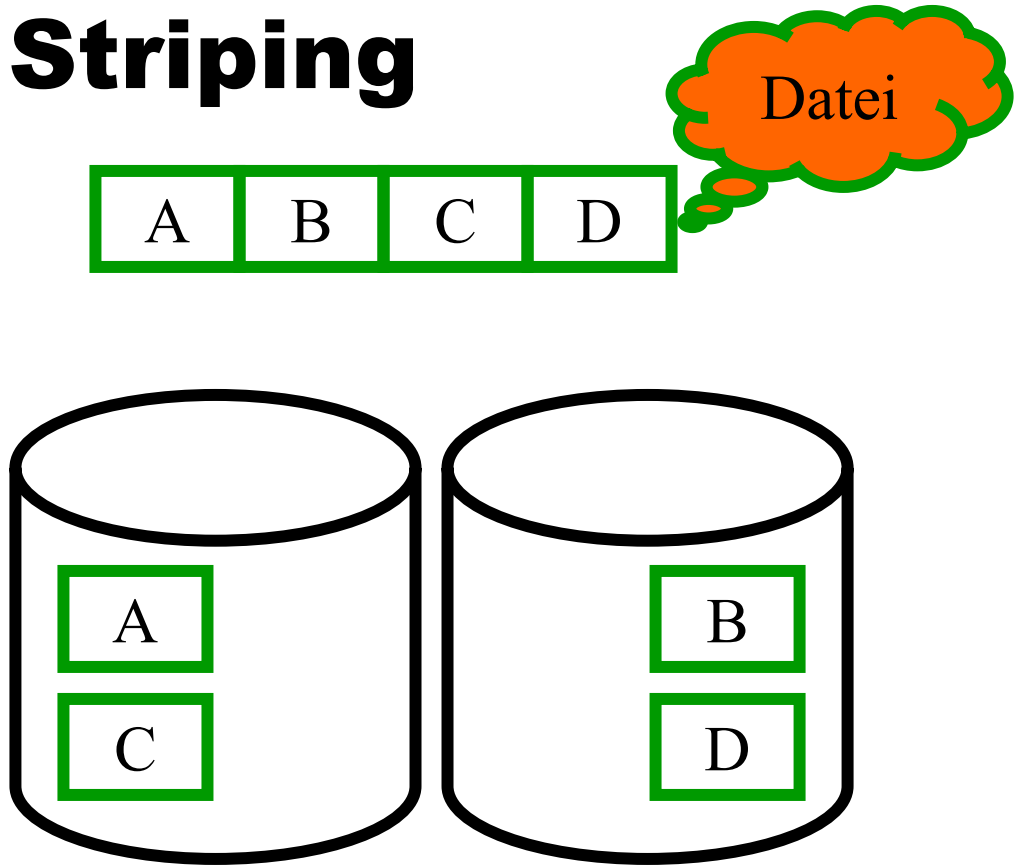
- Mean Time To Failure (MTTF): Erwartungswert für die Zeit (von der Inbetriebnahme) bis zum Ausfall einer Platte
- Mean Time To Repair (MTTR): Erwartungswert für die Zeit zur Ersetzung der Platte und der Rekonstruktion der Daten
- Mean Time To Data Loss (MTTDL): Erwartungswert für die Zeit bis zu einem nicht-maskierbaren Fehler

Disk-Array mit N Platten ohne Fehlertoleranzmechanismen: MTTDL = MTTF / N

Der Schlüssel zur Fehlertoleranz ist Redundanz => Redundant Arrays of Independent Disks (RAID)

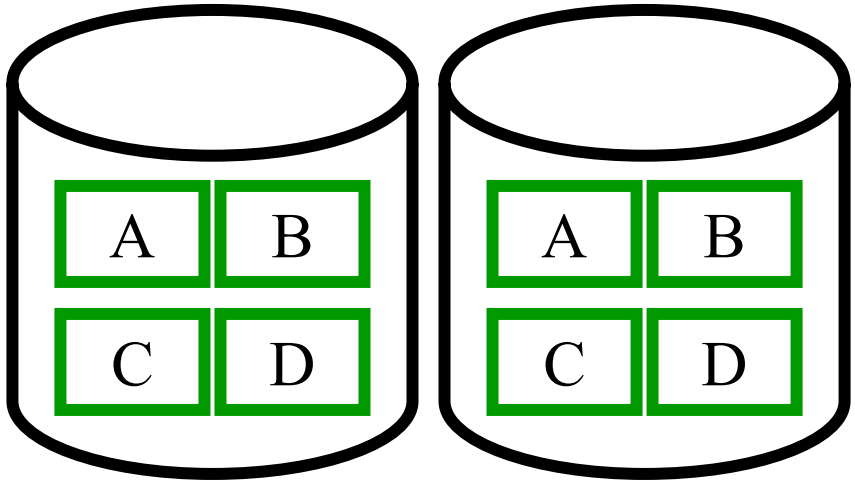
- durch Replikation der Daten (z. B. Spiegelplatten) - RAID1
- durch zusätzlich zu den Daten gespeicherte Error-Correcting-Codes (ECCs), z.B. Paritätsbits (RAID-4, RAID-5)

RAID 0: Striping



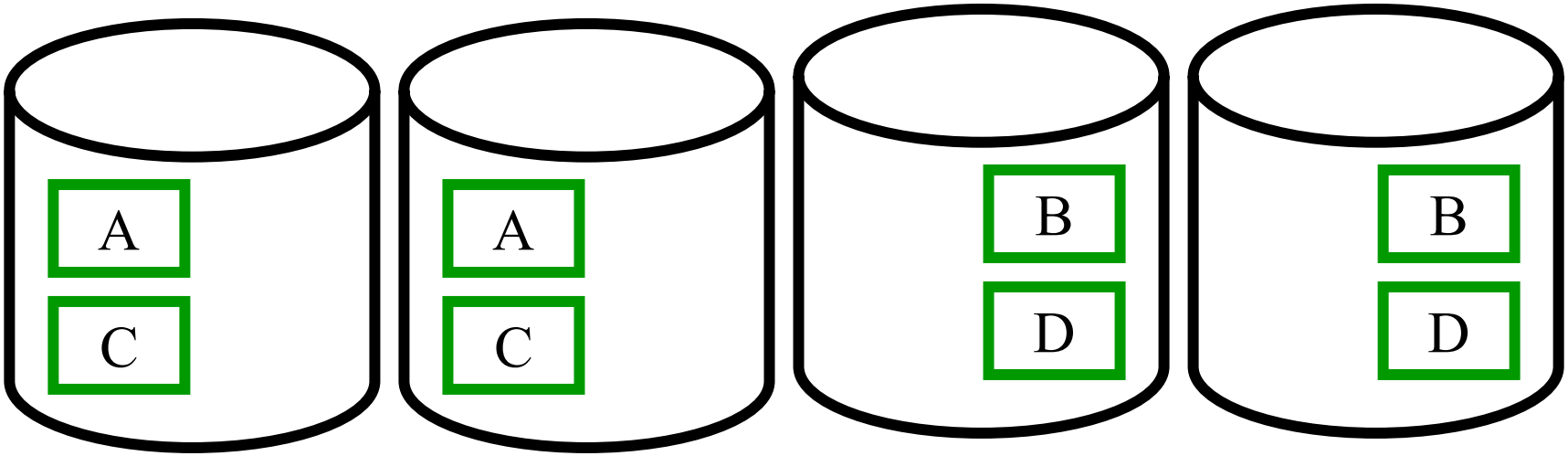
- Lastbalancierung, wenn alle Blöcke mit gleicher Häufigkeit gelesen/geschrieben werden
- Doppelte Bandbreite beim sequentiellen Lesen der Datei bestehend aus den Blöcken ABCD...
- Aber: Datenverlust wird immer wahrscheinlicher, je mehr Platten man verwendet (Stripingbreite = Anzahl der Platten, hier 2)

RAID 1: Spiegelung (mirroring)



- Datensicherheit: durch Redundanz aller Daten (Engl. mirror)
- Doppelter Speicherbedarf
- Lastbalancierung beim Lesen: z.B. kann Block A von der linken oder der rechten Platte gelesen werden
- Aber beim Schreiben müssen beide Kopien geschrieben werden
  - Kann aber parallel geschehen
  - Dauert also nicht doppelt so lange wie das Schreiben nur eines Blocks

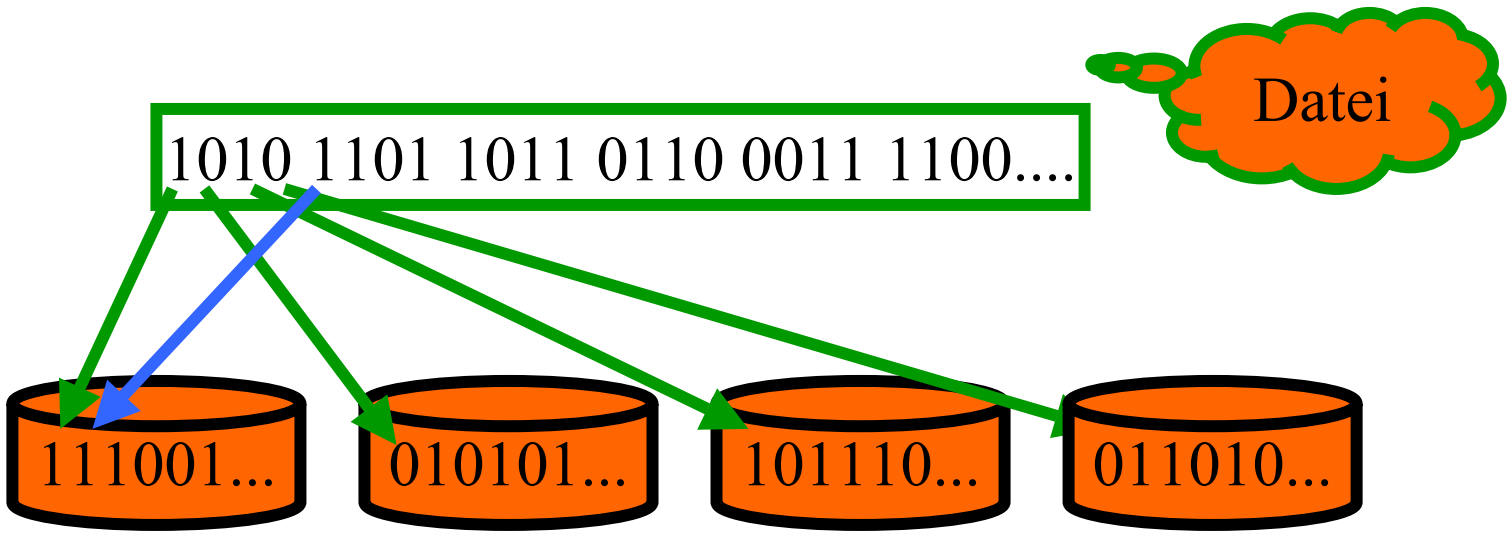
RAID 0+1: Striping und Spiegelung



- Kombiniert RAID 0 und RAID 1
- Immer noch doppelter Speicherbedarf
- Zusätzlich zu RAID 1 erzielt man hierbei auch eine höhere Bandbreite beim Lesen der gesamten Datei ABCD....
- Wird manchmal auch als RAID 10 bezeichnet.

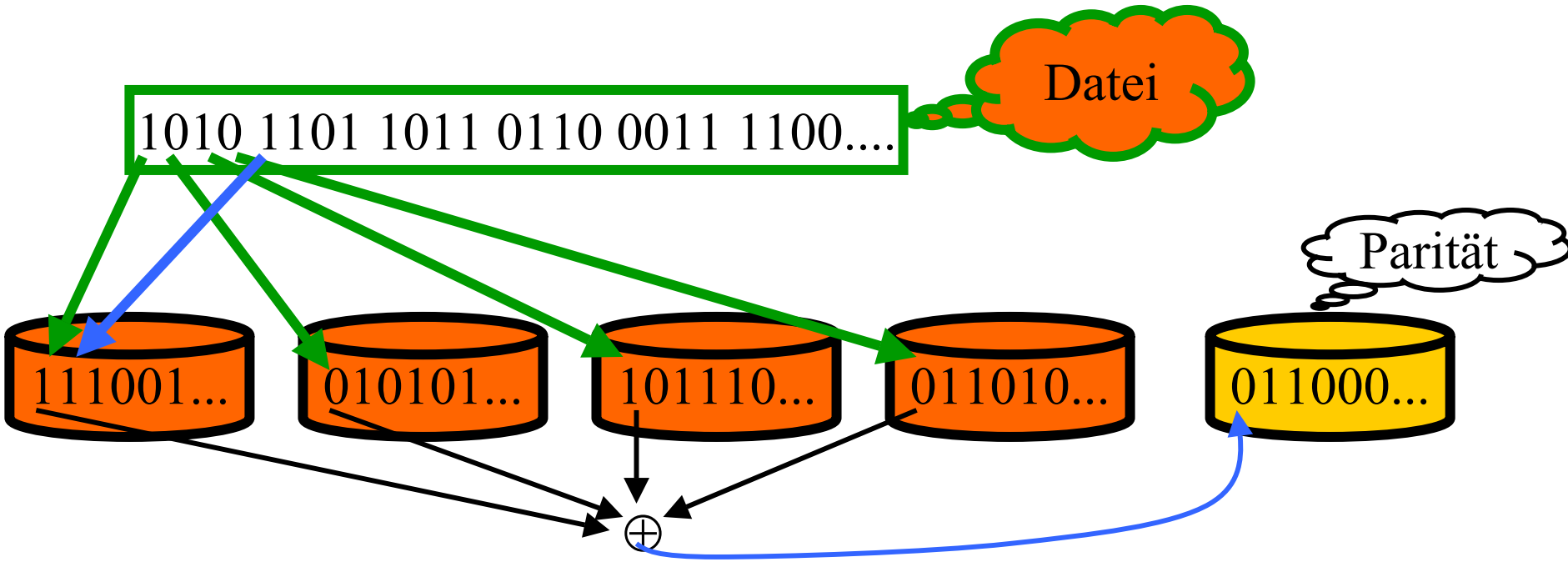
RAID 2: Striping auf Bit-Ebene

- Anstatt ganzer Blöcke, wie bei RAID 0 und RAID 0+1, wird das Striping auf Bit- (oder Byte-) Ebene durchgeführt



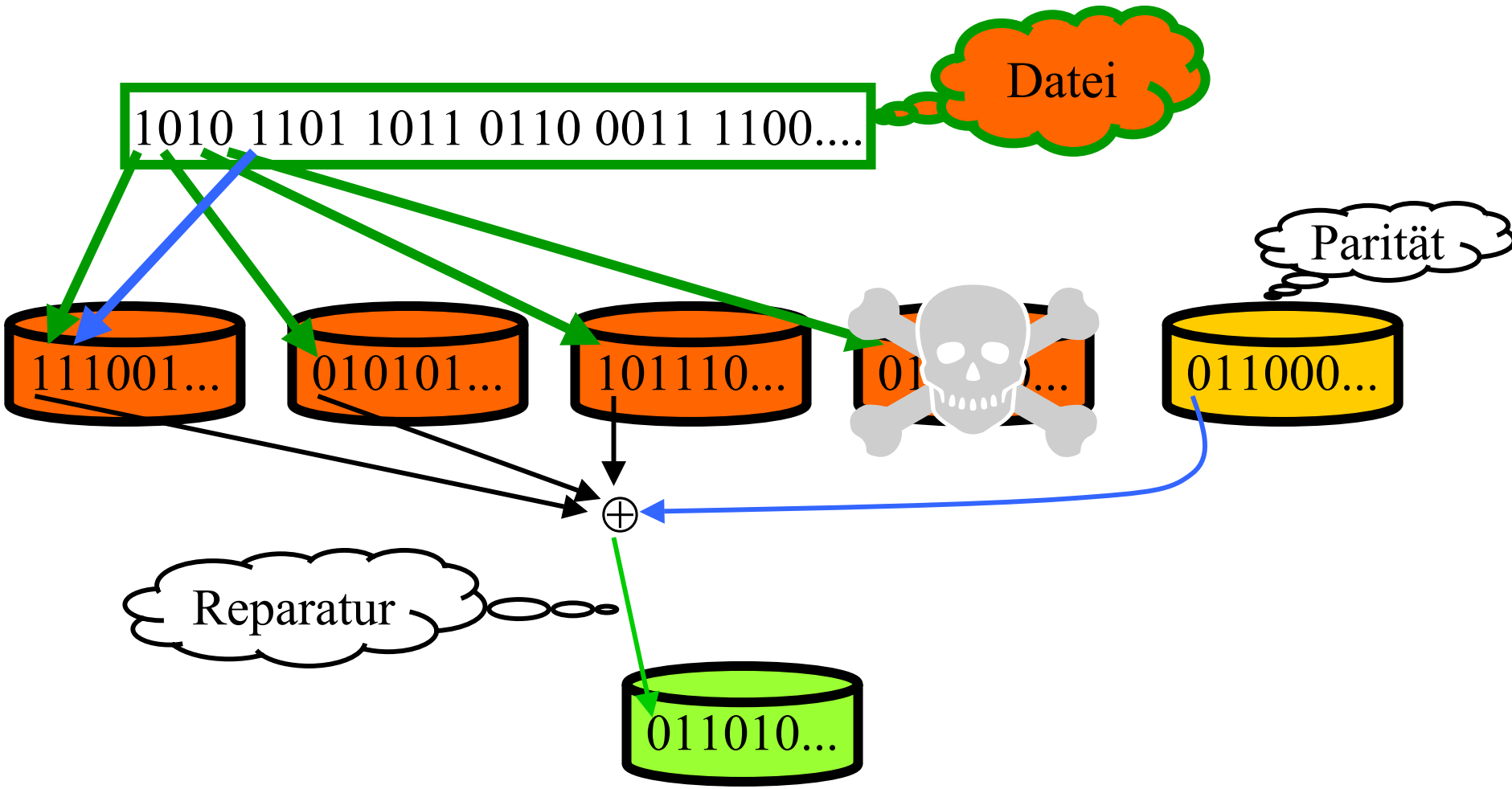
- Es werden zusätzlich auf einer Platte noch Fehlererkennungs- und Korrekturcodes gespeichert
- In der Praxis nicht eingesetzt, da Platten sowieso schon Fehlererkennungscodes verwalten

RAID 3: Striping auf Bit-Ebene, zusätzliche Platte für Paritätsinfo

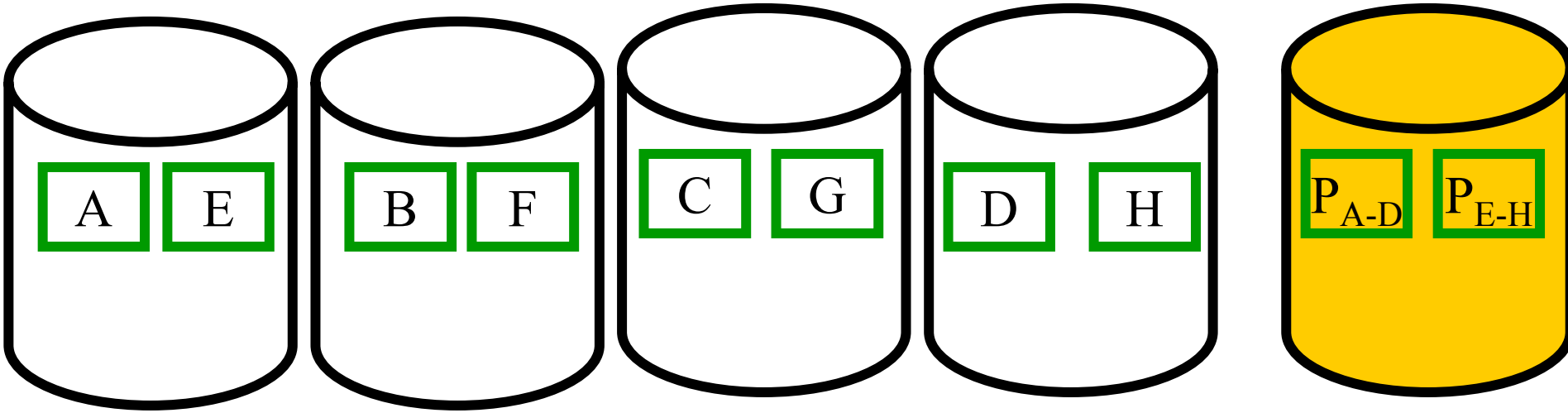


- Das Striping wird auf Bit- (oder Byte-) Ebene durchgeführt.
- Es wird auf einer Platte noch die Parität der anderen Platten gespeichert.
  Parität = bitweise xor (⊕)
- Dadurch ist der Ausfall einer Platte zu kompensieren.
- Das Lesen eines Blocks erfordert den Zugriff auf alle Platten:
  - Verschwendung von Schreib/Leseköpfen
  - Alle marschieren synchron

RAID 3: Plattenausfall

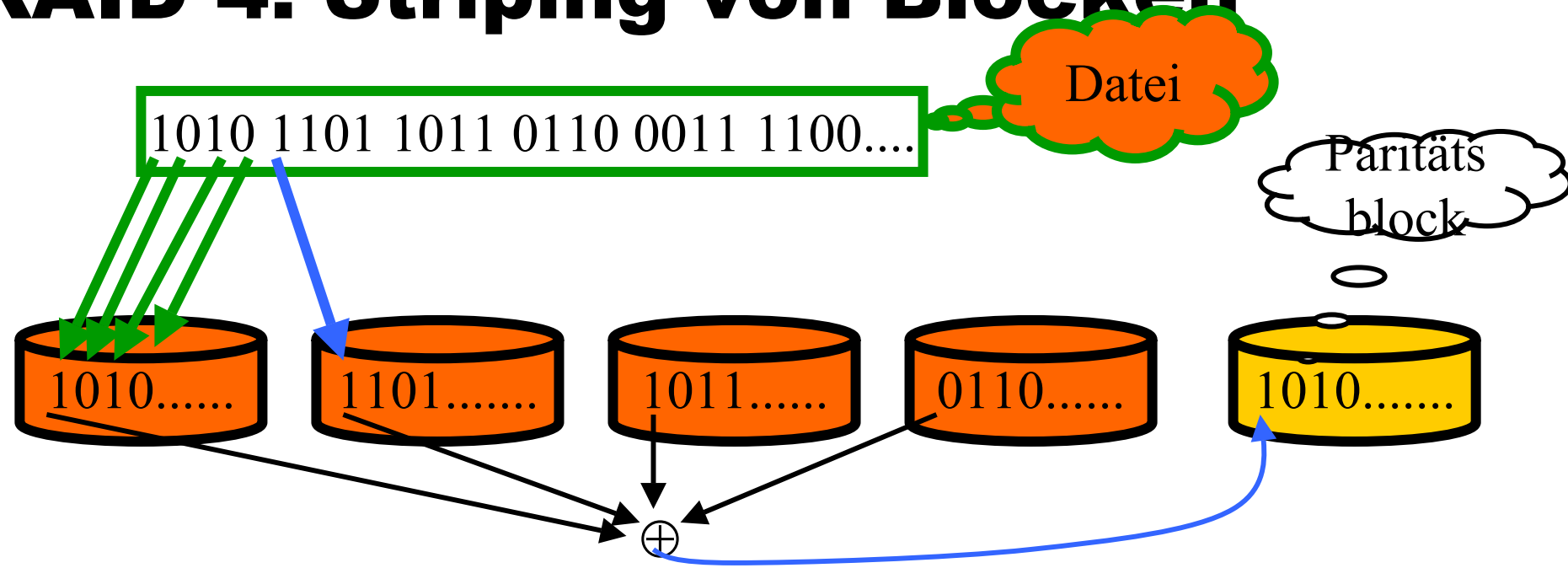


RAID 4: Striping von Blöcken



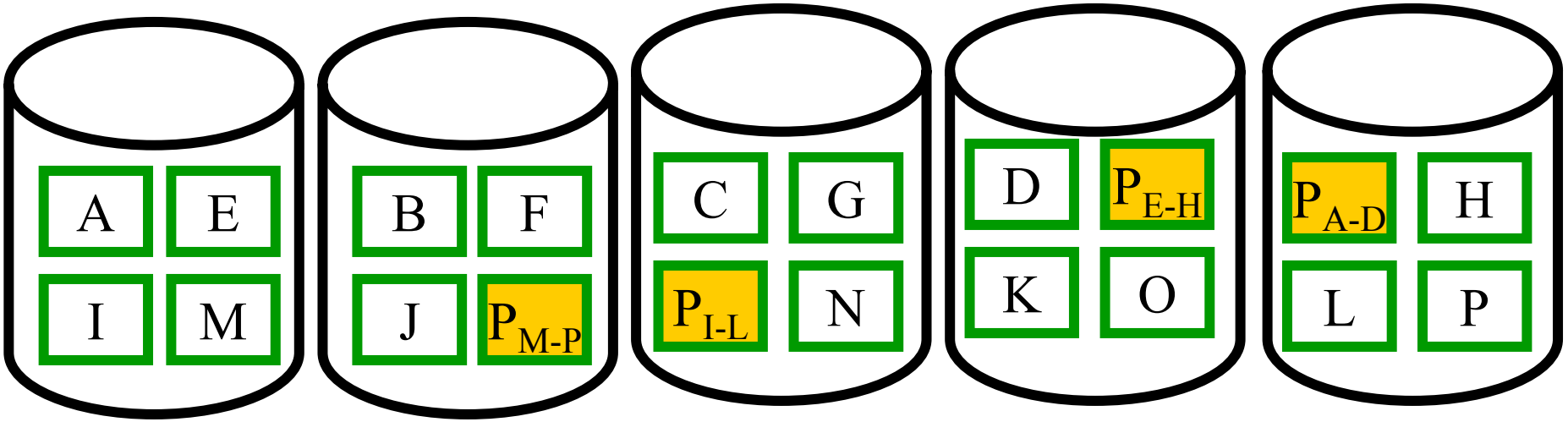
- Bessere Lastbalancierung als bei RAID 3
- Flaschenhals bildet die Paritätsplatte
- Bei jedem Schreiben muss darauf zugegriffen werden
  - Bei Modifikation von Block A zu A‘ wird die Parität P_{A-D} wie folgt neu berechnet:
  - $P'_{A-D} := P_{A-D} \oplus A \oplus A'$
- D.h. bei einer Änderung von Block A muss der alte Zustand von A und der alte Paritätsblock gelesen werden und der neue Paritätsblock und der neue Block A‘ geschrieben werden

RAID 4: Striping von Blöcken



- Flaschenhals bildet die Paritätsplatte
- Bei jedem Schreiben muss darauf zugegriffen werden
  - Bei Modifikation von Block A zu A‘ wird die Parität $P_{A\text{-}D}$ wie folgt neu berechnet:
  - $P'_{A\text{-}D} := P_{A\text{-}D} \oplus A \oplus A'$
- D.h. bei einer Änderung von Block A muss der alte Zustand von A und der alte Paritätsblock gelesen werden und der neue Paritätsblock und der neue Block A‘ geschrieben werden

RAID 5: Striping von Blöcken, Verteilung der Paritätsblöcke



- Bessere Lastbalancierung als bei RAID 4
- die Paritätsplatte bildet jetzt keinen Flaschenhals mehr
- Wird in der Praxis häufig eingesetzt
- Guter Ausgleich zwischen Platzbedarf und Leistungsfähigkeit

Lastbalancierung bei der Blockabbildung auf die Platten

Vergleich von Greedy-Verfahren und Round-Robin-Allokation

Datei1

1.1	1.2	1.3	1.4	1.5	1.6
Hitze: 10	4	4	3	2	1

Datei 2

2.1	2.2	2.3	2.4
Hitze: 8	5	5	1

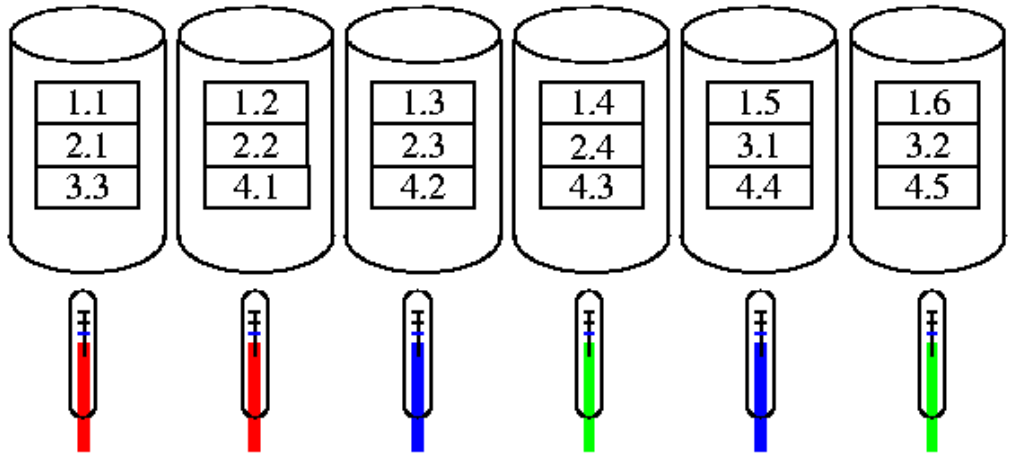
Datei 3

3.1	3.2	3.3
Hitze: 5	5	5

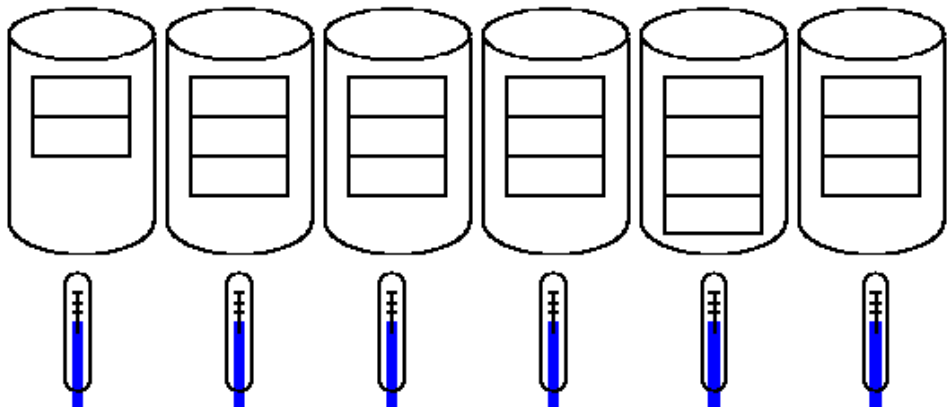
Datei 4

4.1	4.2	4.3	4.4	4.5
Hitze: 7	4	3	2	1

Round-Robin-Allokation



Greedy-Methode

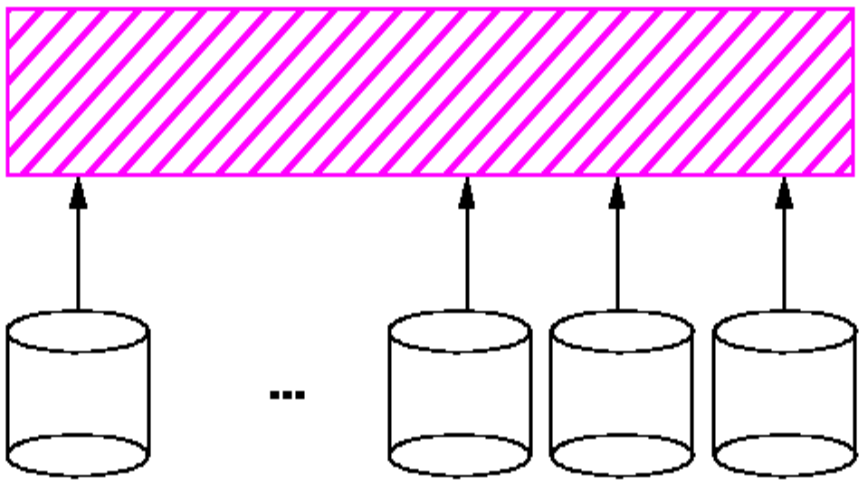


Parallelität bei Lese/Schreib-Aufträgen

Voraussetzung: ***Declustering*** von Dateien über mehrere Platten

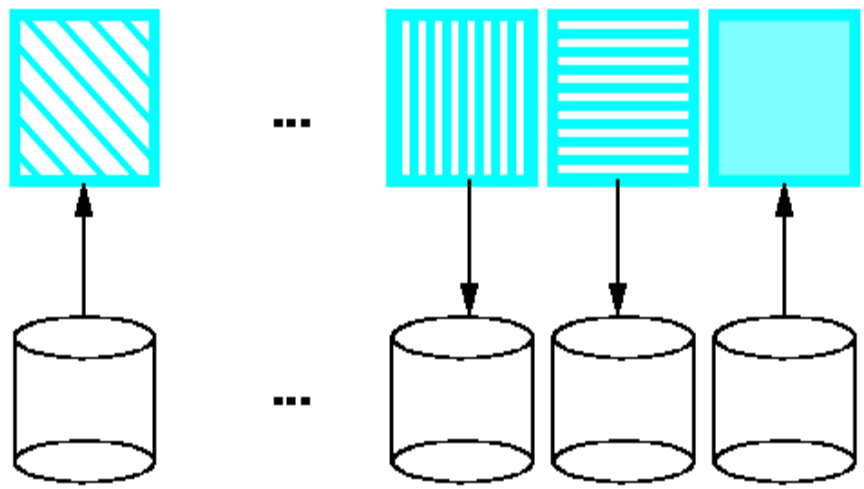
2 generelle Arten von E/A-Parallelität

Intra-E/A-Parallelität (Zugriffsparallelität)



1 E/A-Auftrag wird in mehrere, parallel ausführbare Plattenzugriffe umgesetzt

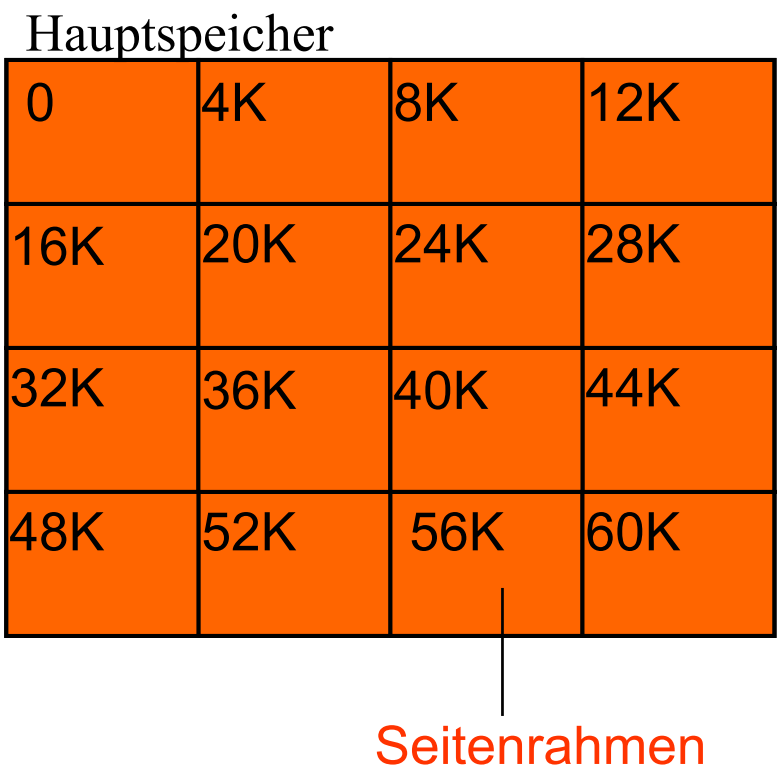
Inter-E/A-Parallelität (Auftragsparallelität)

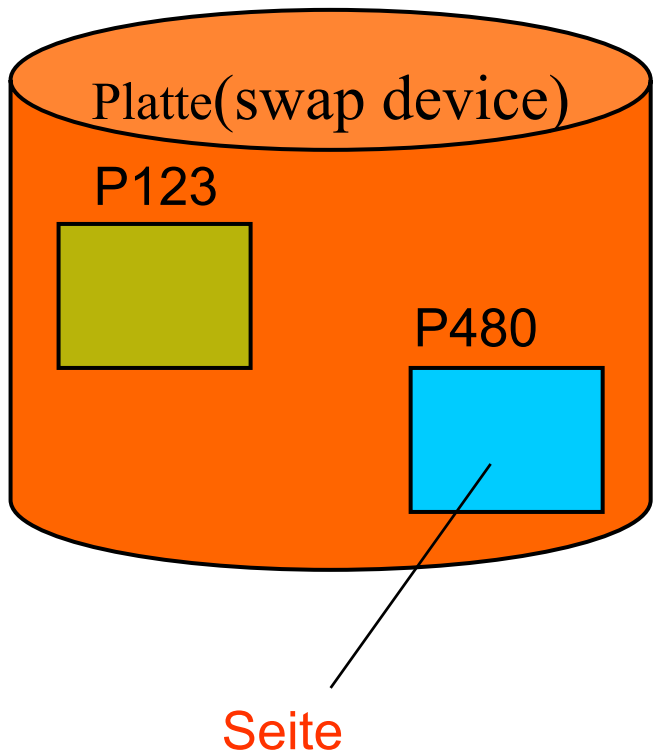


Mehrere unabhängige E/A-Aufträge können parallel ausgeführt werden, sofern die betreffenden Daten über verschiedene Platten verteilt sind

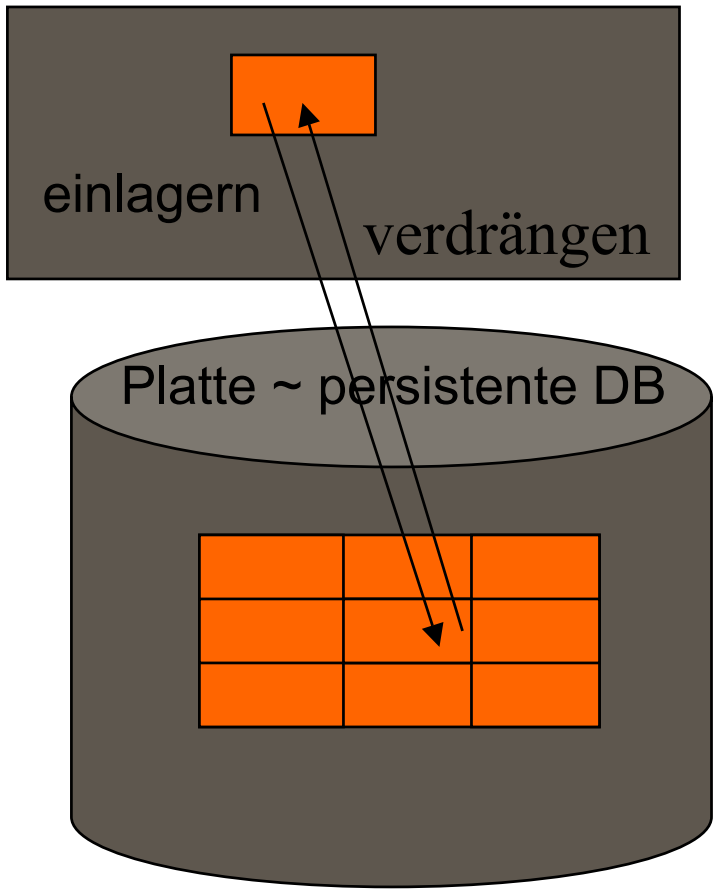
Ein- und Auslagern von Seiten

- Systempuffer ist in Seitenrahmen gleicher Größe aufgeteilt
- Ein Rahmen kann eine Seite aufnehmen
- „Überzählige“ Seiten werden auf die Platte ausgelagert





Systempuffer-Verwaltung



Hauptspeicher

Verdrängungsstrategien:

- Least-Recently-Used (LRU)
- First-in-first out (FIFO)
- Second Chance
- Zähler (simulierte Uhr)

(Ziel: Annäherung der Belady-Strategie: Seite, die am längsten nicht mehr benötigt wird, verdrängen)

Adressierung von Tupeln auf dem Hintergrundspeicher

Naiver Ansatz:

- jedes Tupel hat gleiche Länge l
- i. Tupel hat Position (i -1) x l

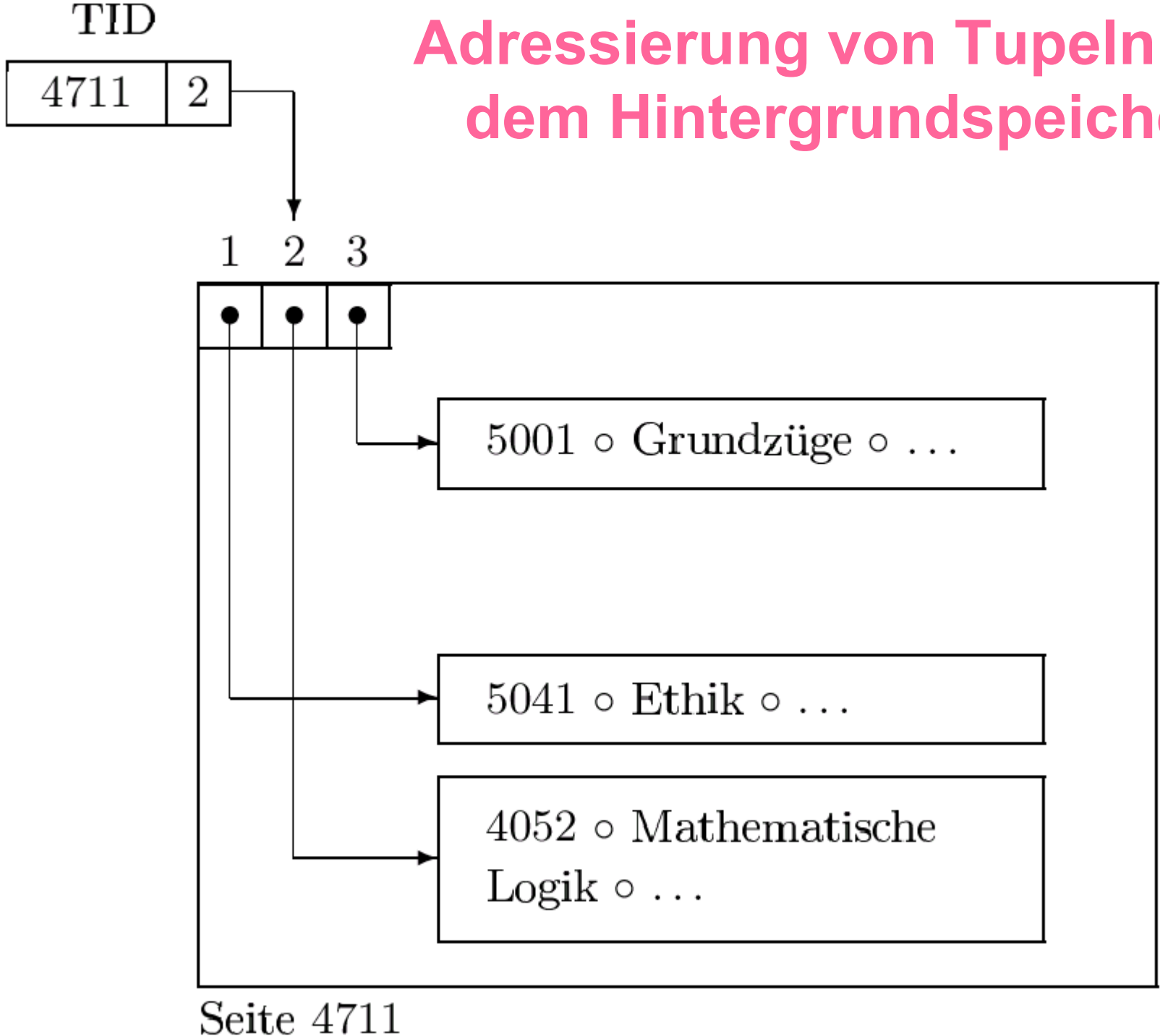
Komplexe Anforderungen verlangen flexiblere Adressierung:

- variable Feldlängen
- Logdateien
- mehrfacher Bezug auf Werte

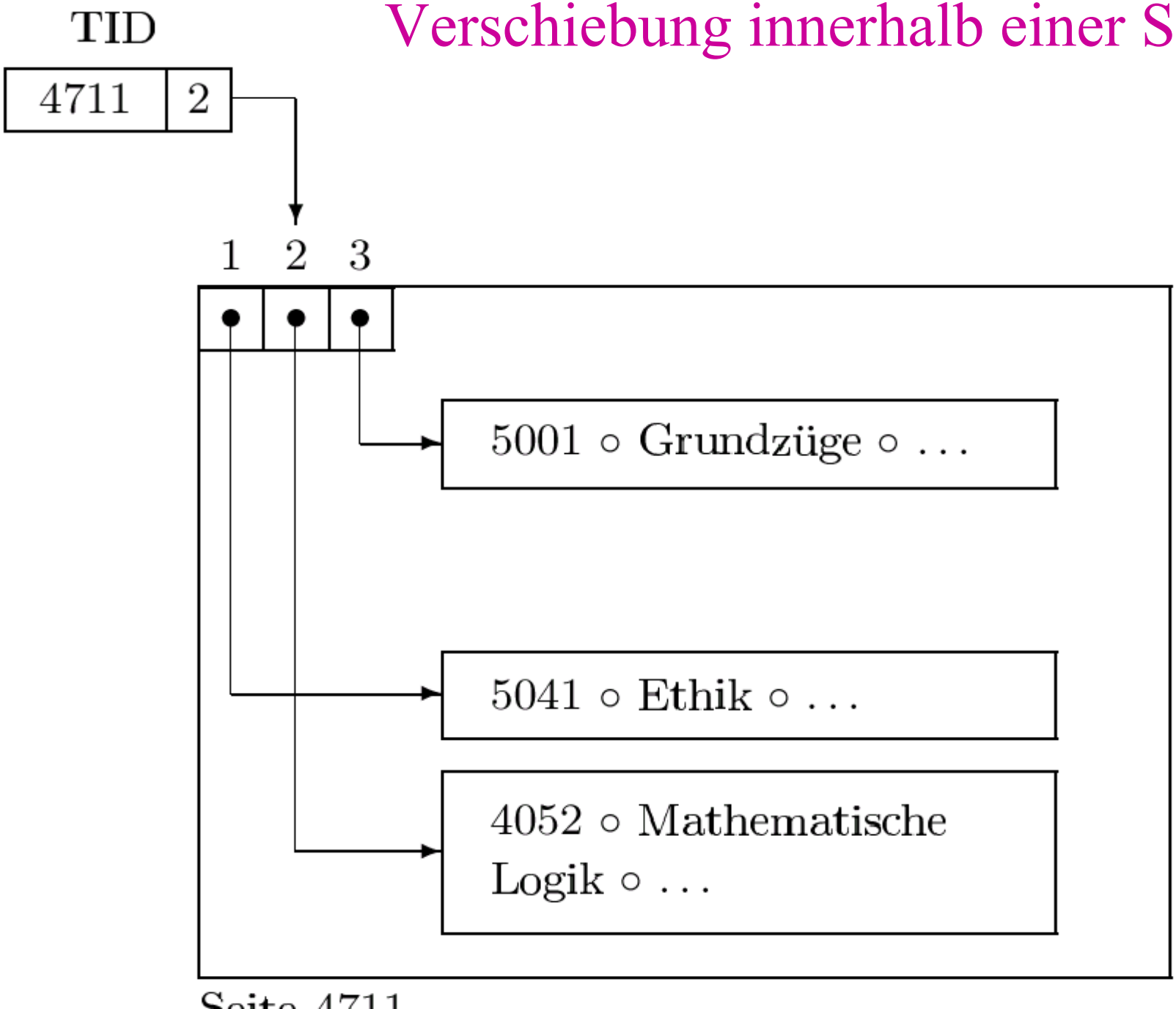
Lösung:

- nur die Seiten werden nach obigem Schema addressiert.
- innerhalb einer Seite werden Zeiger verwendet.

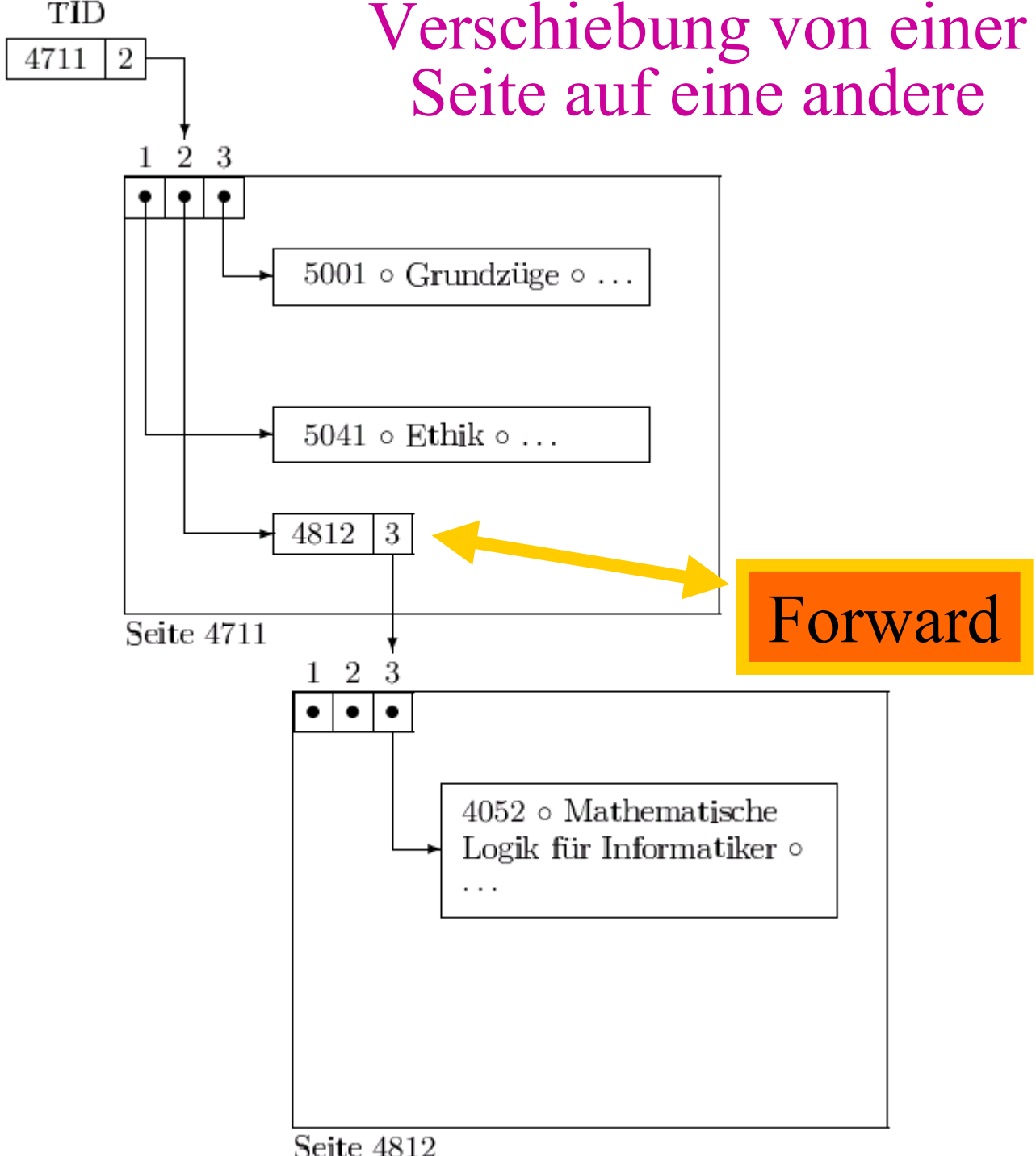
Adressierung von Tupeln auf dem Hintergrundspeicher



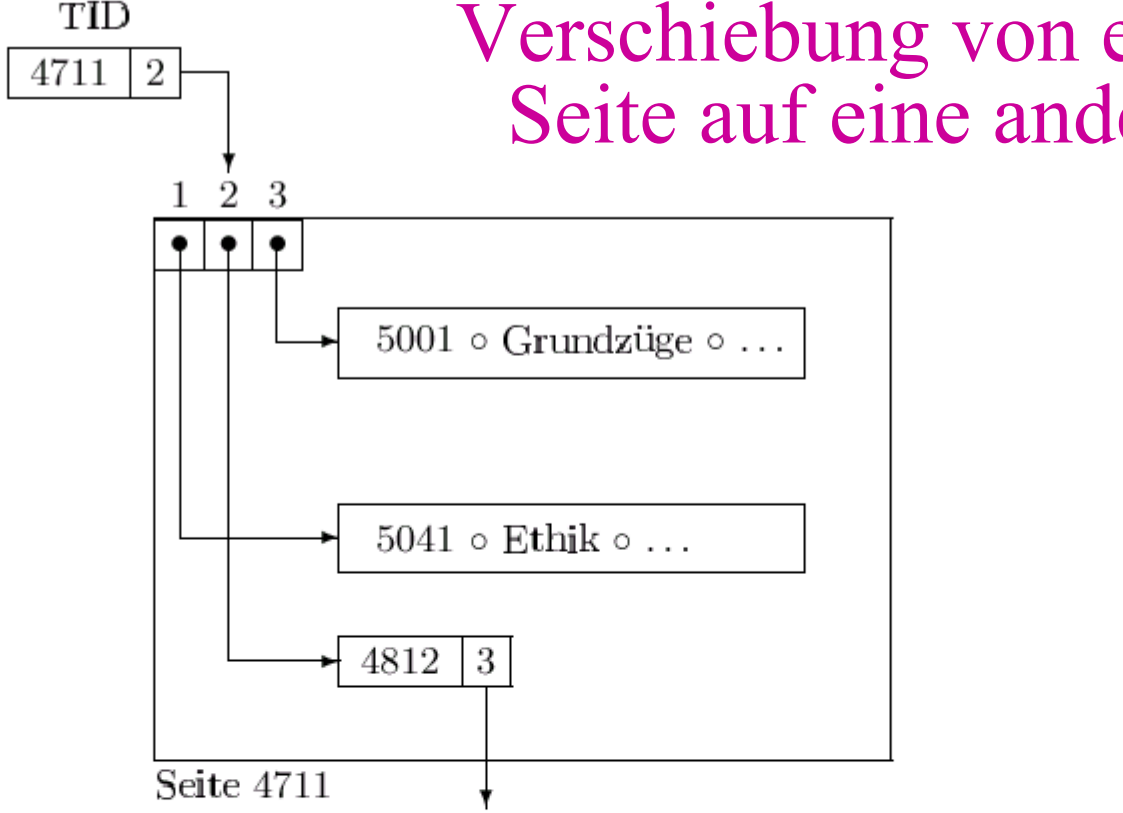
Verschiebung innerhalb einer Seite

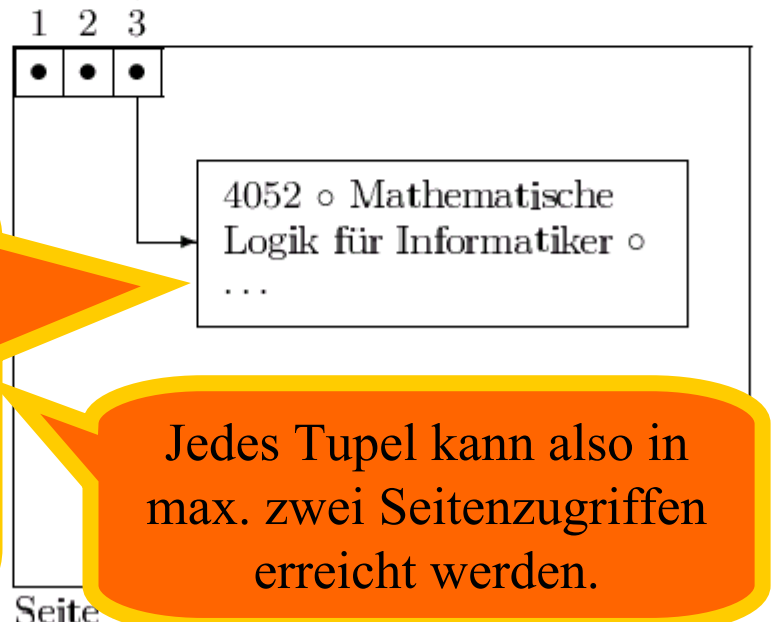


Verschiebung von einer Seite auf eine andere



Verschiebung von einer Seite auf eine andere





Bei der nächsten Verschiebung wird der „Forward“ auf Seite 4711 geändert (kein weiterer Forward auf Seite 4812)

Jedes Tupel kann also in max. zwei Seitenzugriffen erreicht werden.