Fachhochschule Nordwestschweiz
Hochschule für Technik

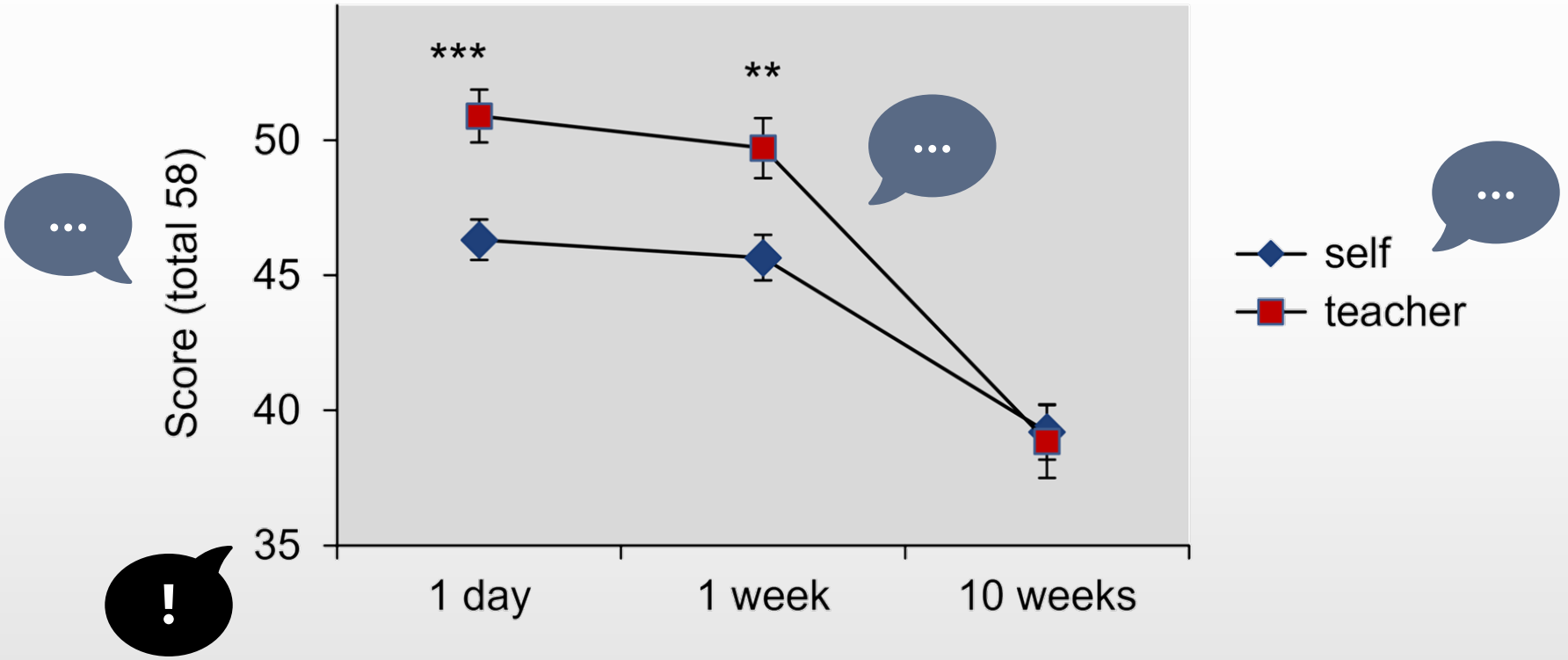
Application Performance Management

I/O & Buffering

Michael Faes

Nachtrag: Diagramme

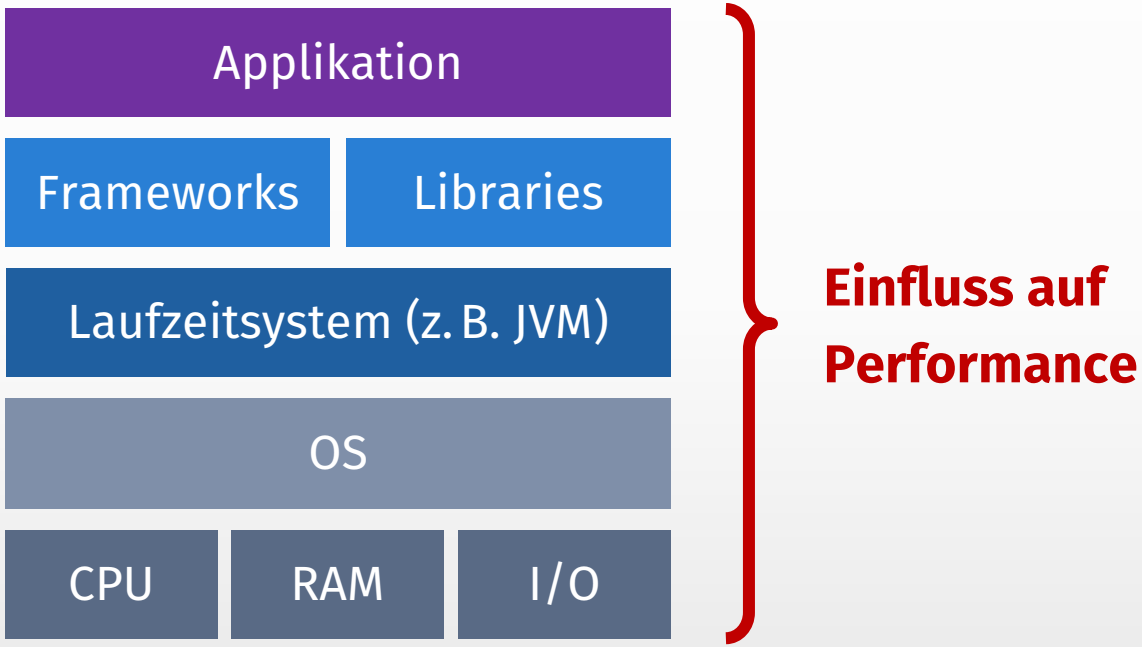
Kürzlich an einer Weiterbildung gesehen:



(Im Paper deutlich besser.)

Rückblick: Schichten & Performance

Sämtliche Schichten eines Systems haben *Einfluss auf Performance*:



Und: Abstraktionsschichten haben üblicherweise auch *Kosten*!

> “We can solve any problem in computer science by introducing an extra level of indirection.”
>
> — David J. Wheeler

> “… except the problem of *too many levels of indirection*.”
>
> — unbekannt

Indirektion: Möglichkeit, ein «Ding» durch Namen/Referenz zu verwenden, statt direkt

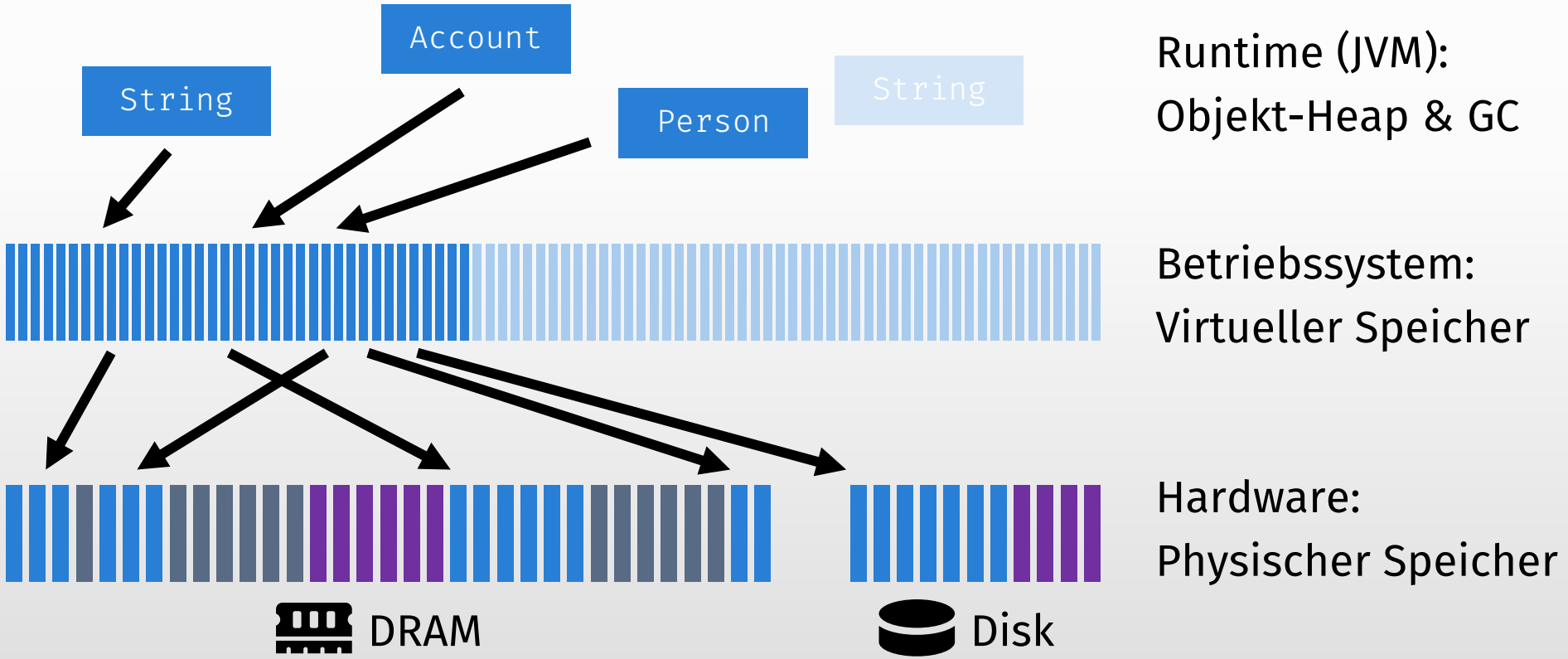
Beispiele

- Domainname statt IP-Adresse (und IP-Adresse statt MAC-Adresse)
- Programmierung: Referenz/Pointer statt Wert (kopieren)
- Virtuelle statt physische Speicheraddresse

Kosten von Abstraktionen

Indirektion ist *eine Art von* Abstraktion. Oft nicht gratis.

Beispiel: Speicher



Runtime (JVM):
Objekt-Heap & GC

Betriebssystem:
Virtueller Speicher

Hardware:
Physischer Speicher

Heute: Input/Output

Input/Output (I/O) wird ebenfalls durch Abstraktionen vereinfacht.
Und performance-mässig beeinflusst!

Beispiel: Datei lesen in Java

Setup:
Core i9-10885H
64 GB RAM
SSD
Windows 10
Java 17

```java
@Benchmark
@BenchmarkMode(Mode.SampleTime)
public int read() throws IOException {
    try (var in = Files.newInputStream(pathTo5MBFile)) {
        int zeroCount = 0;
        int b;
        while ((b = in.read()) >= 0) {
            if (b == 0) { zeroCount++; }
        }
        return zeroCount;
    }
}
```

Benchmark	Mode	Cnt	Score	Units
BytewiseCountZero.read	sample	5	**10.328**	**s**/op

Übersicht Woche 7

1. Übungsbesprechung
2. Einführung I/O
3. I/O-Grundlagen
4. Buffering
5. I/O-Performance-Tuning in Java
6. Übung

I/O-Grundlagen

I/O & Computer-Architektur

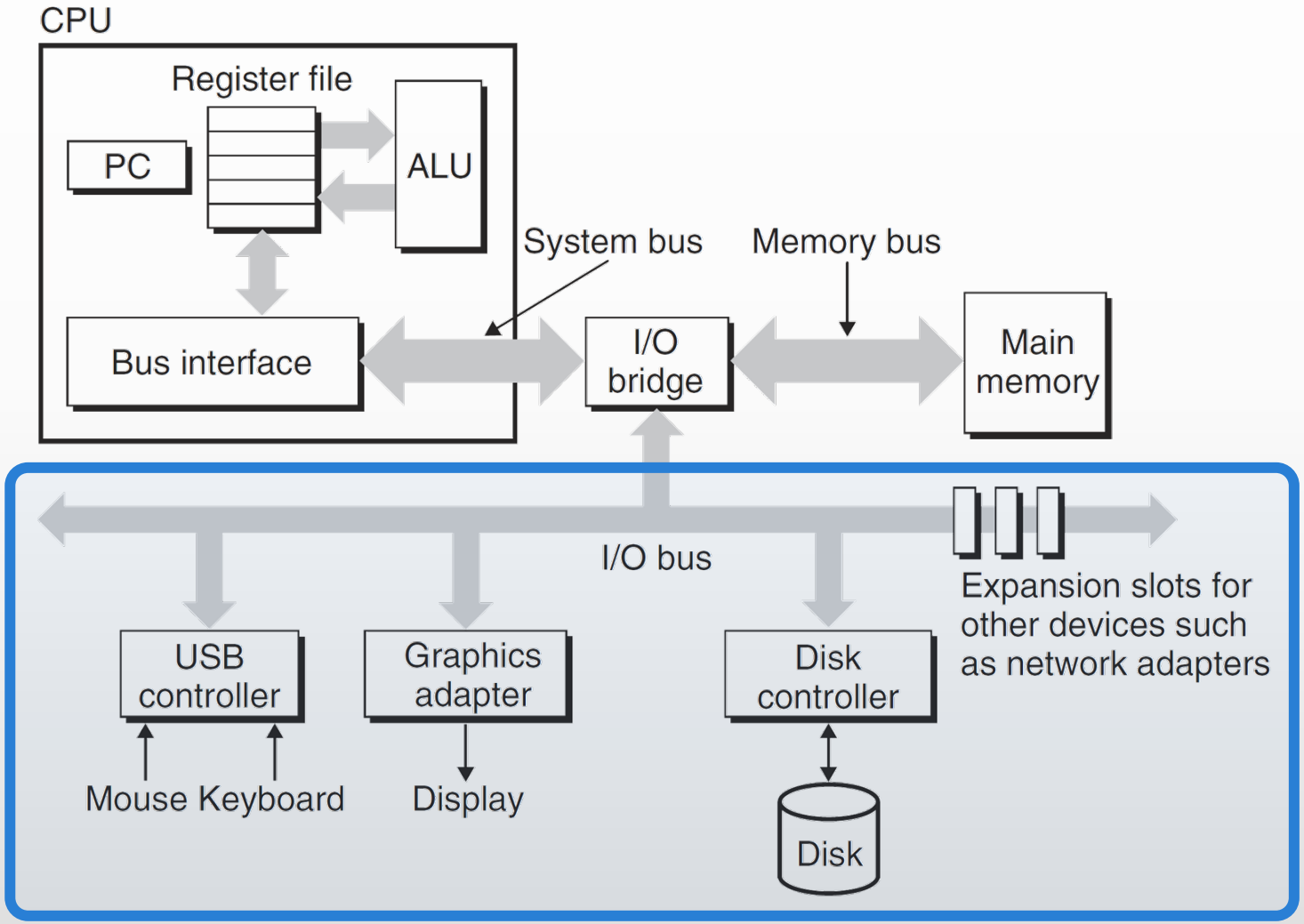


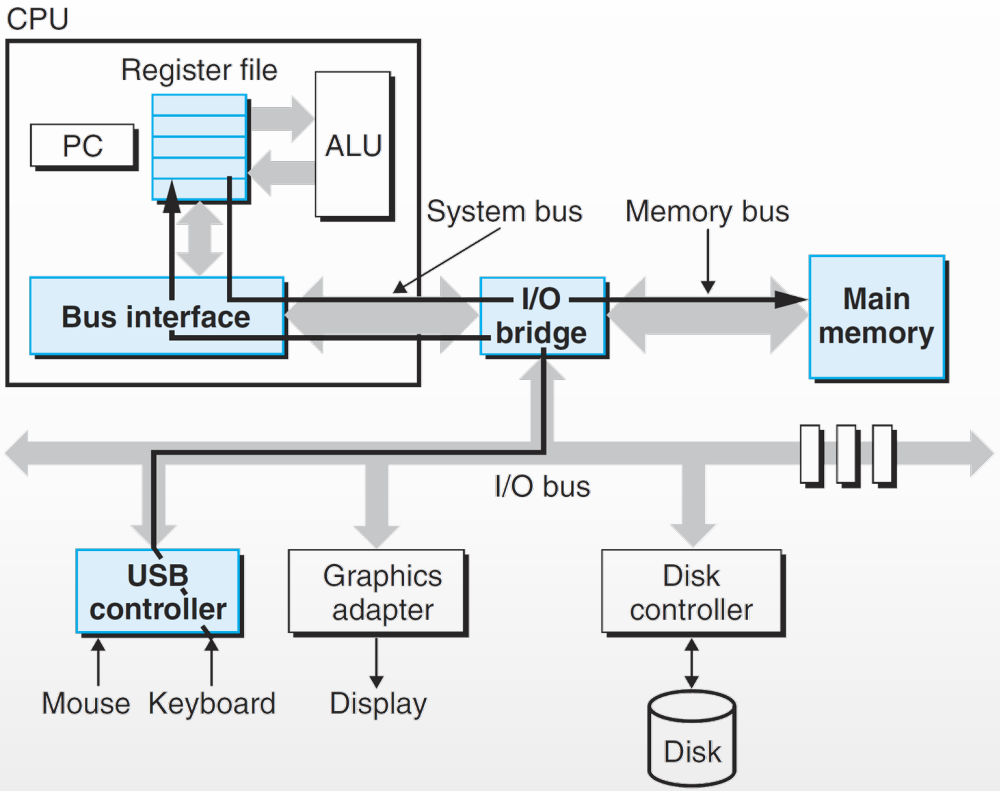
Bild: Bryant & O'Hallaron (2015)

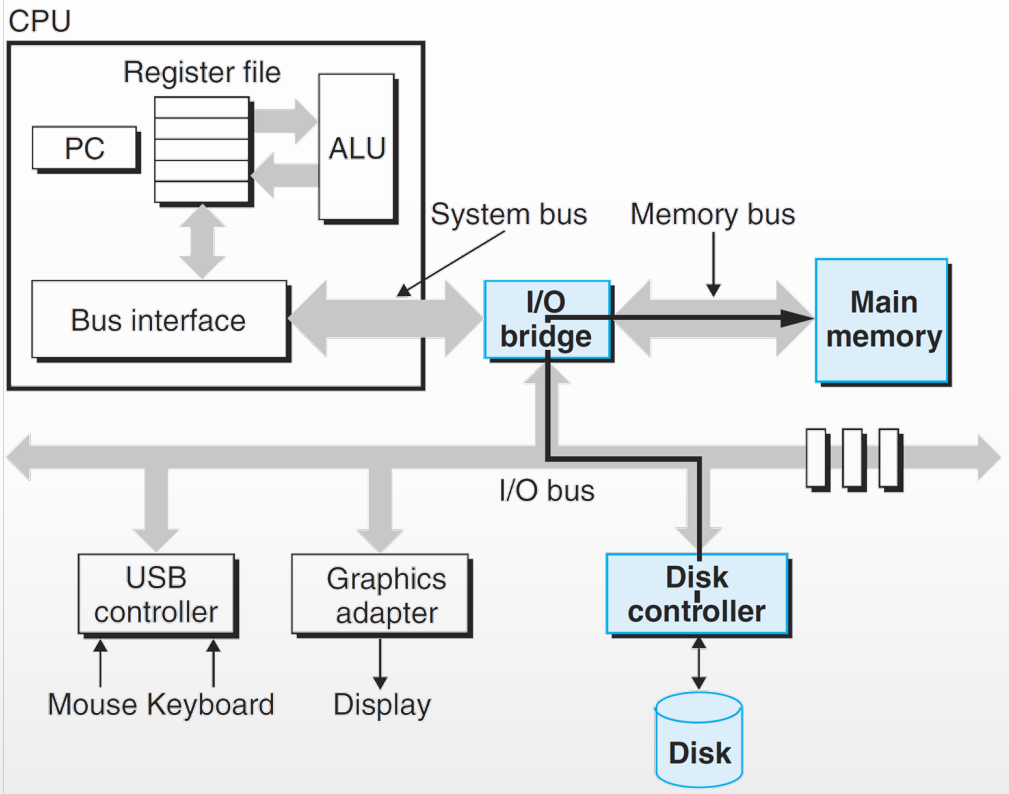
I/O:

Alle Datentransfers, welche *nicht* zwischen CPU(s) und Speicher stattfinden.

- Eingabegeräte
- Disks/SSD
- Netzwerk
- Grafik & Sound
- ...

Arten von I/O





Programmed I/O (PIO)

Transfer via CPU

Direct Memory Access (DMA)

Transfer von Gerät direkt in RAM

Bilder: Bryant & O'Hallaron (2015)

Rückblick: Zeitmassstäbe in Computersystemen:

Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	3 ns	10 s
Level 3 cache access	10 ns	33 s
Main memory access (DRAM, from CPU)	100 ns	6 min
Solid-state disk I/O (flash memory)	10–100 µs	9–90 hours
Rotational disk I/O	1–10 ms	1–12 months
Internet: San Francisco to New York	40 ms	4 years
Internet: San Francisco to United Kingdom	81 ms	8 years
Lightweight hardware virtualization boot	100 ms	11 years
Internet: San Francisco to Australia	183 ms	19 years
OS virtualization system boot	< 1 s	105 years
TCP timer-based retransmit	1–3 s	105–317 years
SCSI command time-out	30 s	3 millennia
Hardware (HW) virtualization system boot	40 s	4 millennia
Physical system reboot	5 m	32 millennia

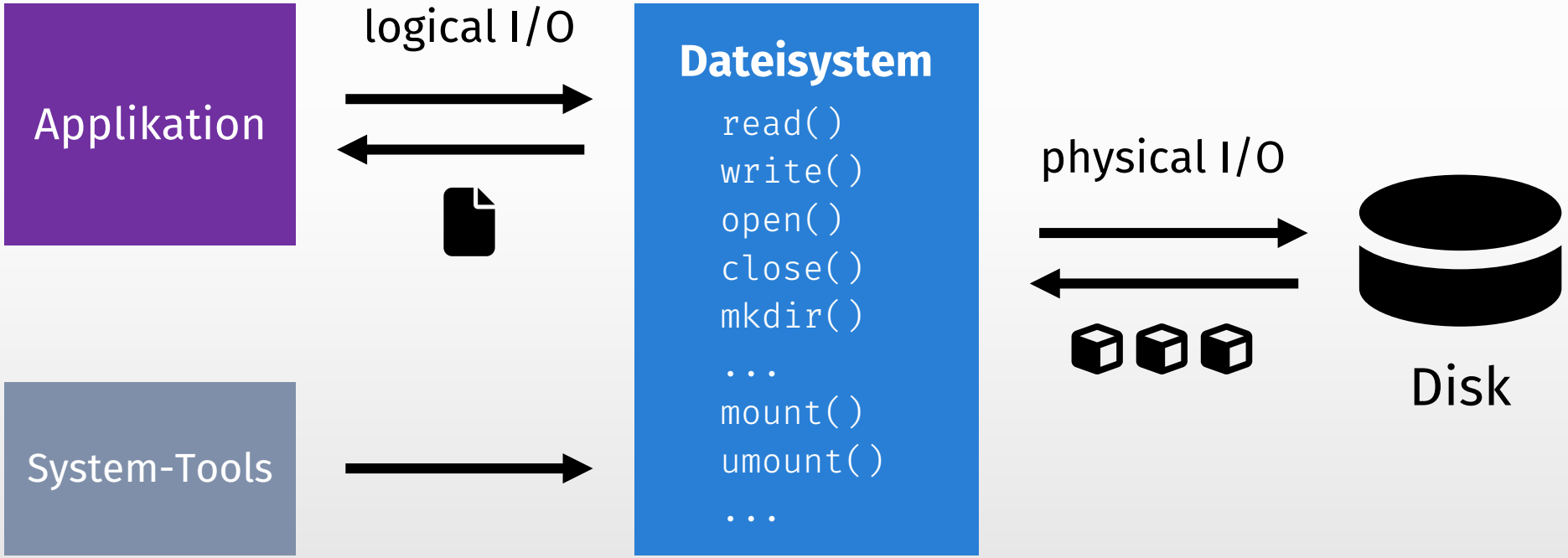
Quelle: Gregg (2020)

Vorteil von DMA:
CPU kann während Zugriff auf Disk etwa 10’000’000 Befehle ausführen!

Datei-I/O: logisch vs. physisch

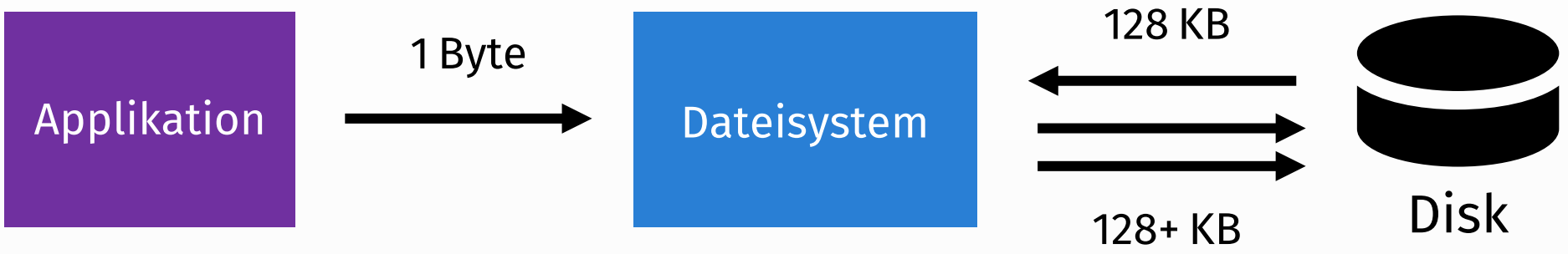
Häufige Art von I/O: Lesen und Schreiben von *Dateien*

Abstraktion durch *Dateisystem* (Teil des Betriebssystems):



Dateisysteme sind oft *Block-basiert*, d. h. schreiben nicht einzelne Bytes, sondern ganze Blöcke, z. B. 4 KB gross.

Logical und physical I/O können sich deutlich unterscheiden!



Mechanismen, die Einfluss auf Unterschied haben:

- Zusammenfassen in Blöcke
- Buffering
- Caching
- Prefetching
- Kompression
- Journaling
- RAID
- ...

Beispiel, +/- aus Gregg (2020):

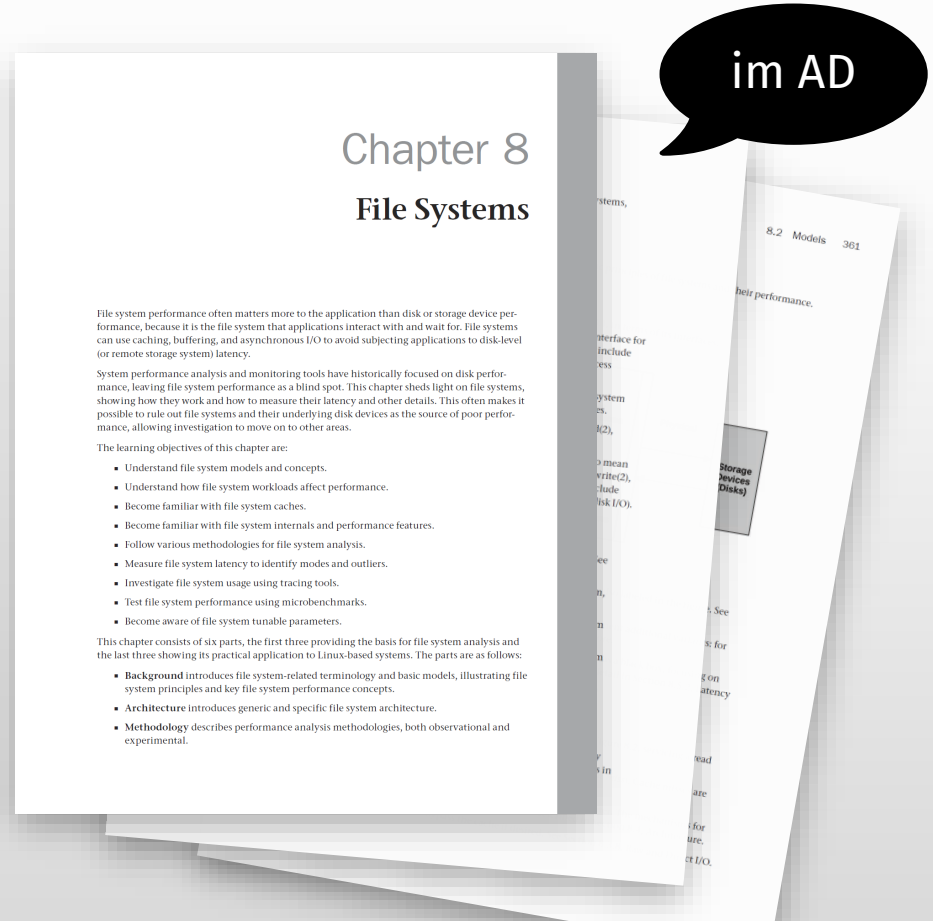
1. Applikation macht 1-Byte-Änderung an existierender Datei
2. Dateisystem identifiziert Ort als Teil eines 128-KB-Records (der nicht bereits im RAM gecachet ist)
3. Dateisystem lädt Record von Disk in Hauptspeicher
4. Dateisystem ersetzt das Byte im Record mit neuem Byte
5. Irgendwann später verlangt OS, dass der «Dirty»-Record mit Grösse 128 KB zurück auf Disk geschrieben wird
6. Dateisystem schreibt zusätzlich ein paar Metadaten, z. B. bezüglich Zugriffszeit

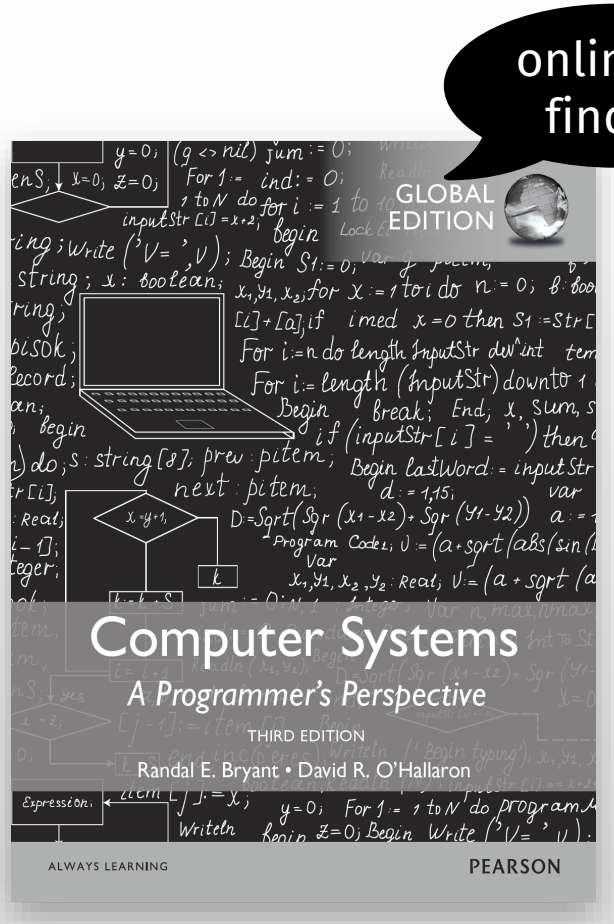
Bedeutet: Von logical I/O kann nur schwer (oder gar nicht) auf physical I/O geschlossen werden…

Dateisysteme & physical I/O...

Take-Home-Message: Dateisysteme und physical I/O sind komplex...
Zu komplex, um hier im Detail zu untersuchen.

Weitere Informationen:



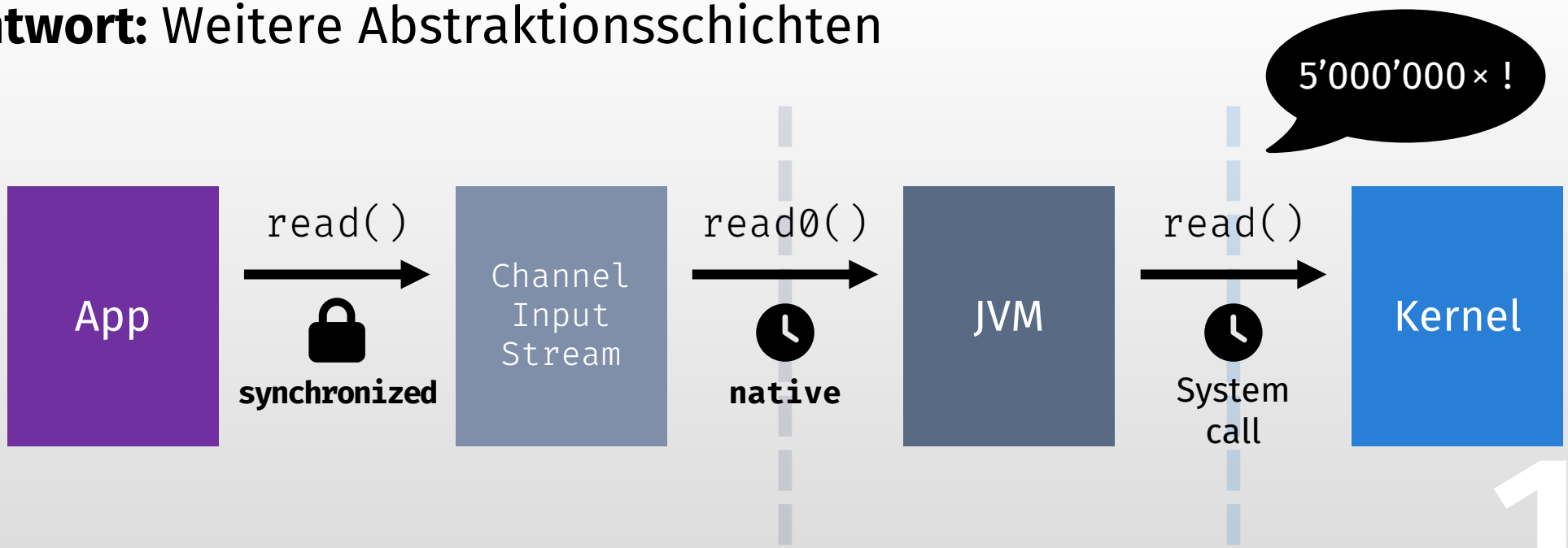


I/O in der JVM

Wenn Betriebssystem I/O bereits durch Caching, Buffering, DMA, … optimiert, wieso ist Performance so schlecht?

```
var in = newInputStream(pathTo5MBFile);
int b;
while ((b = in.read()) >= 0) {
    ...
}
```

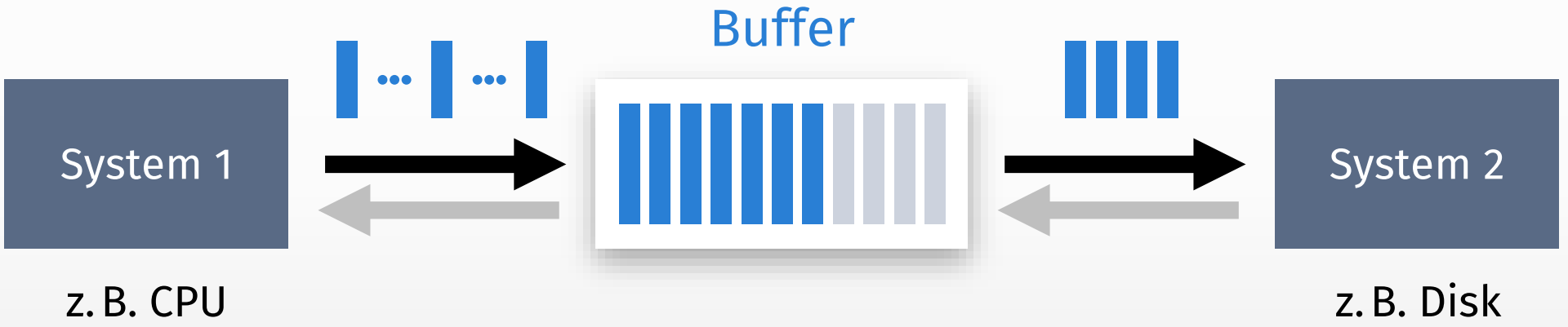
Antwort: Weitere Abstraktionsschichten



Buffering

Buffering

Buffer: Zwischenspeicher für Datenübertragung, wenn Verarbeitungs-geschwindigkeiten *unterschiedlich* oder ***variabel*** sind.

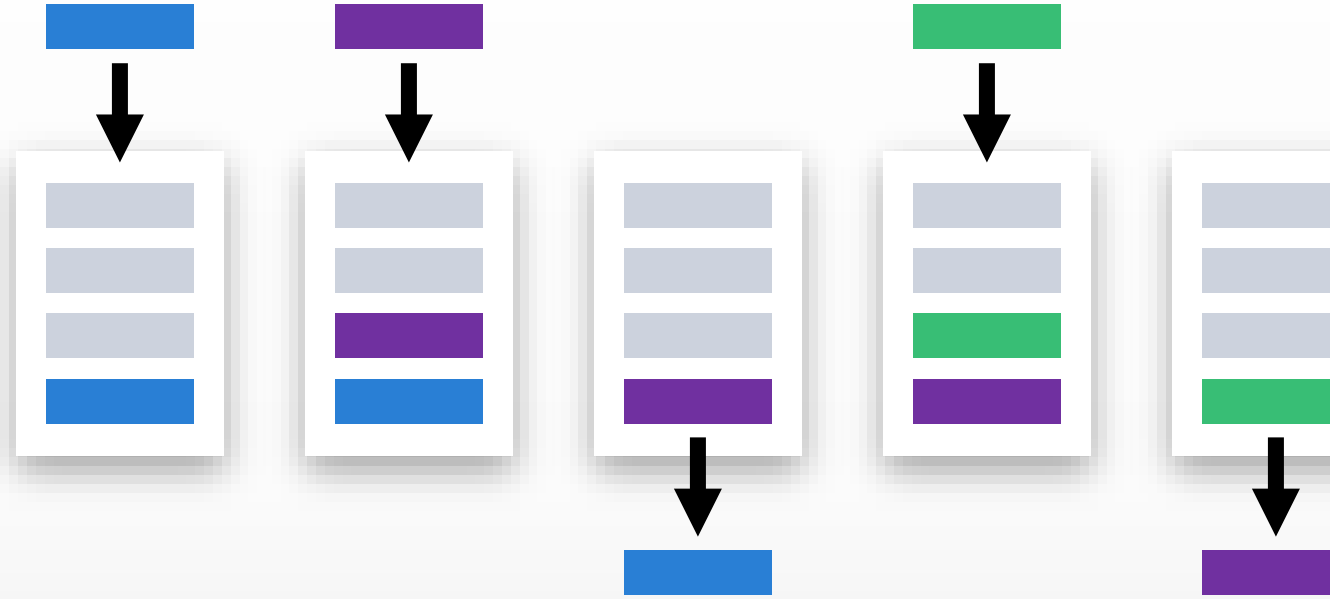


Analogie: Warteschlange vor Achterbahn

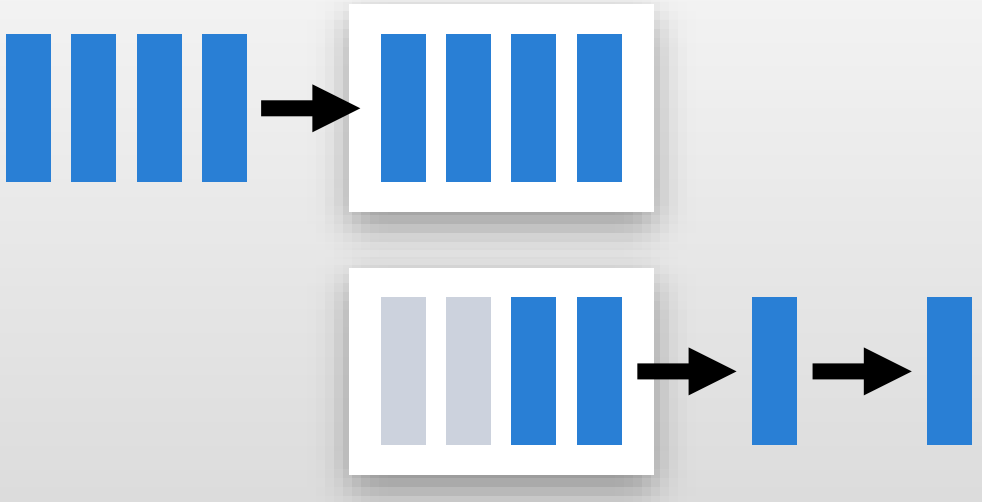
- Leute stossen in hoher Rate dazu, aber einzeln / in kleinen Gruppen
- Wagen nehmen mit niedriger Rate Leute auf, aber viele aufs Mal

Buffering & I/O

Im Allgemeinen können Lesen und Schreiben von Buffer «gleichzeitig» geschehen, z. B. bei FIFO-Buffer:



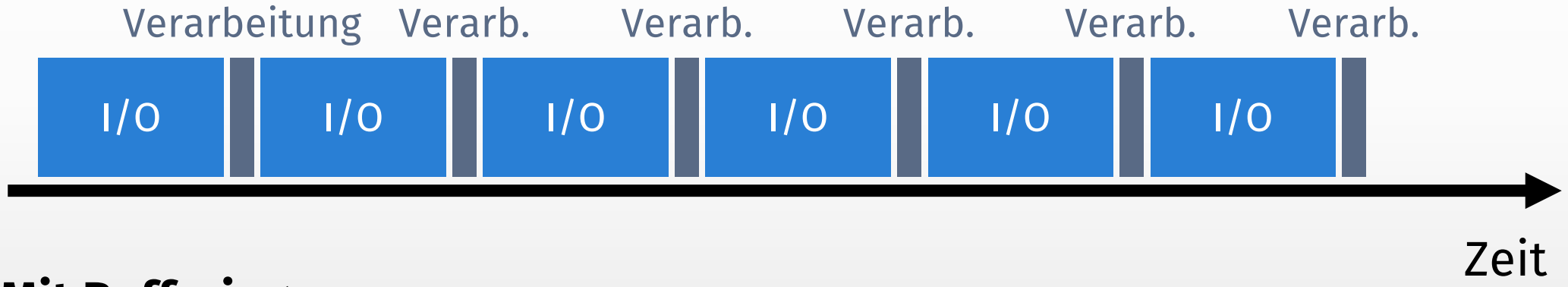
Bei Datei-I/O: Buffer wird durch I/O **vollständig** gefüllt (bzw. geleert).
App verarbeitet nachher (bzw. vorher) Byte für Byte.



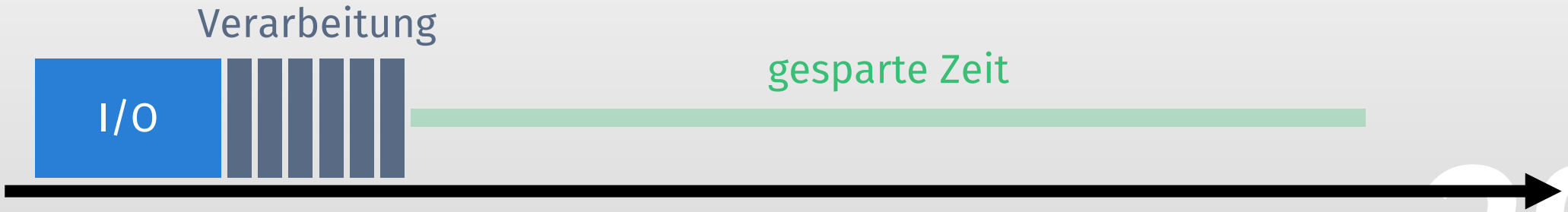
Bündeln von Operationen

Andere Perspektive: Durch Buffering werden *viele teure* Operationen zu *einer einzigen* (immer noch teuren) «gebündelt».

Ohne Buffering:

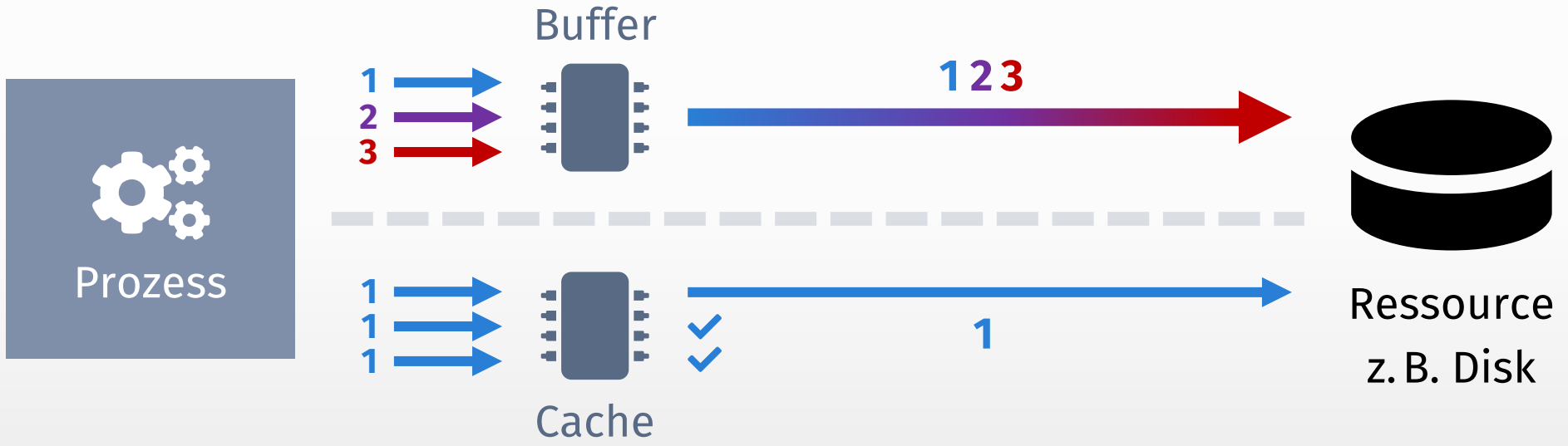


Mit Buffering:



Buffering vs. Caching

Abgrenzung zu *Caching*: Beschleunigen von **identischen** wiederholten Zugriffen auf langsame Ressource.

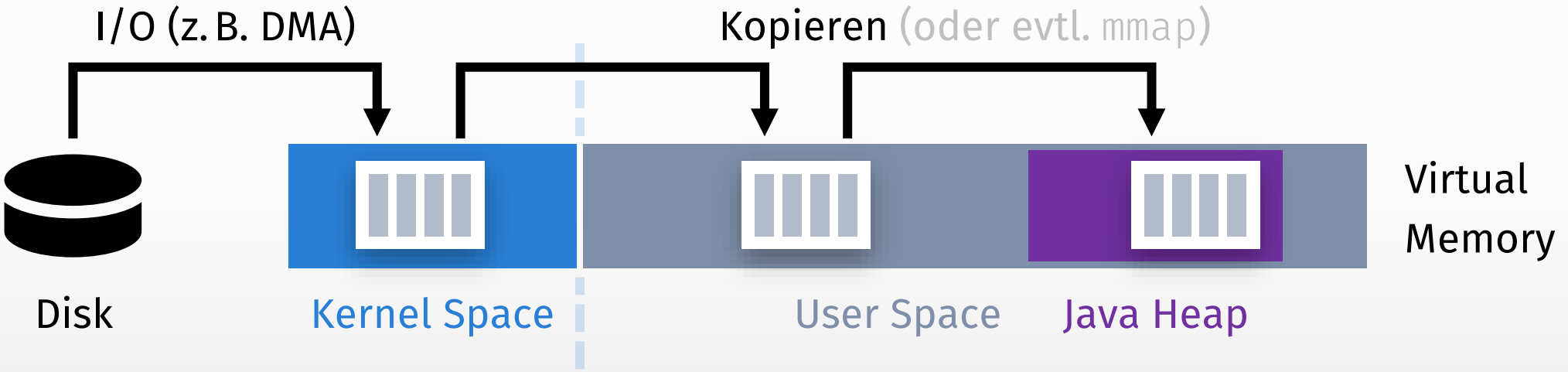


Beispiele

- Datei wird mehrmals eingelesen
- Stylesheet für Website wird mehrmals gebraucht
- IP-Adresse für Domain wird mehrmals verwendet

Buffering in Java

Betriebssystem macht bereits Buffering, aber in Speicher in *Kernel Space*. Zugriff nur durch teure `native`-Methoden & System Calls.



Wieso I/O nicht direkt in ein Java-`byte[]`?

Kosten von Abstraktionen! `byte[]` ist ein normales Java-Objekt, das von GC jederzeit verschoben werden kann…

Stattdessen: Brauchen weitere Buffer, in *User Space* und Java-Heap…

I/O-Performance-Tuning in Java

Eine Art Fallstudie

Fragen?