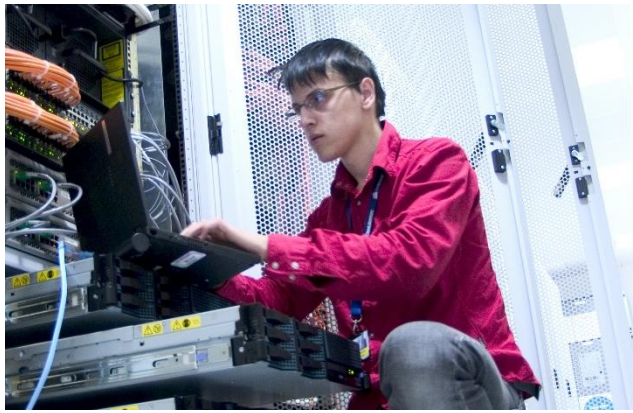
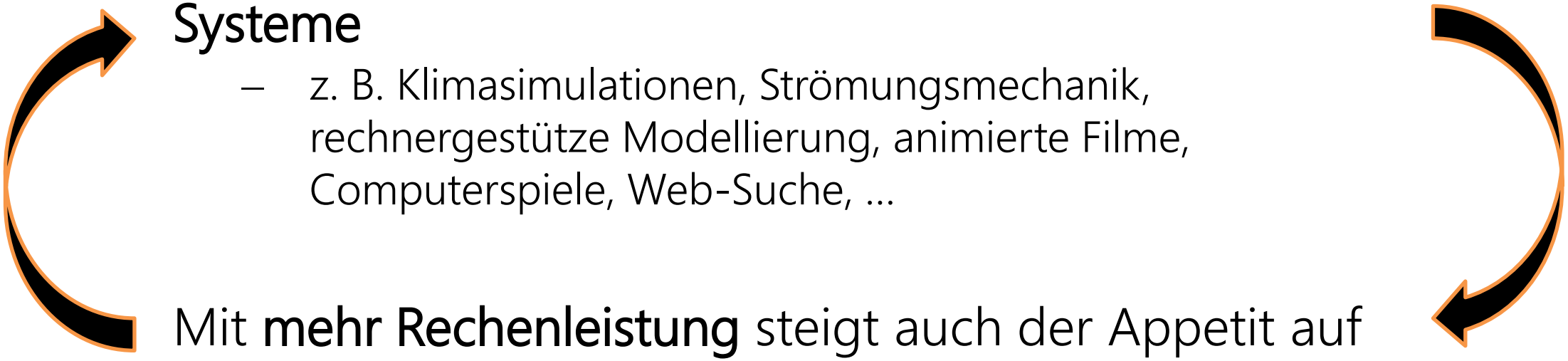
EMIL-VON-BEHRING GYMNASIUM

Einführung Nebenläufige Prozesse



Aktuelle Situation



Immer **größere Probleme** erfordern immer **schnellere Systeme**

- z. B. Klimasimulationen, Strömungsmechanik, rechnergestütze Modellierung, animierte Filme, Computerspiele, Web-Suche, …

Mit **mehr Rechenleistung** steigt auch der Appetit auf anspruchsvollere Anwendungen

Beispiel: Computerspiele

- Simulationen, Darstellungen werden immer realistischer → erfordern mehr Rechenkapazität
- Beispiel: *Assassin's Creed*-Reihe

Beispiel: Computeranimation

- Pixar: *Brave*, *Toy Story*, *WALL-E*, …
- Dreamworks: *Drachenzähmen leicht gemacht*, …
- **Beispiel „Drachenzähmen leicht gemacht 2“**
- 90*60*24 Frames = 129.600 Frames
- 398 TB Animationsdaten
- 90 Mio. Rechenstunden für Final Cut (berechnet in unter einer Woche)
- Figuren-Design mit Animationsbearbeitung in Echtzeit, verwendet bis zu 24 Cores (Kerne)

Weitere Beispiele

- Google durchsucht und indiziert einen signifikanten Anteil der Seiten im Web
- **„Deep Blue“** (IBM) besiegt G. Kasparov (Mai 1997)
- **„Watson“** (IBM) besiegt 2 Menschen in Jeopardy (Februar 2011)
  - 2880 parallele Threads, 90 Octcores
- **AlphaGo** (Google) besiegt L. Sedol (März 2016)
  - 1202 CPUs, 176 GPUs

CPUs im Wandel

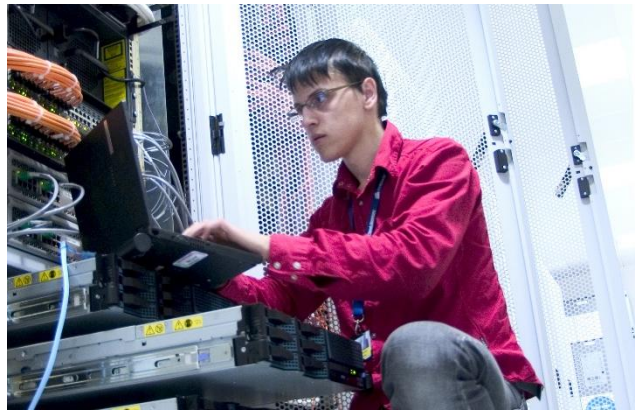
- **There ain‘t no such thing as a free lunch.**

„The free lunch is over.“ – Herb Sutter

- **Früher:** schnellere CPUs → kürzere Laufzeiten
- **Heute:** mehr CPUs →
  - Nicht automatisch!
  - Parallele Prozesse bzw. Programmierung
  - Warum? → siehe PDF zu schnelleren Prozessoren

EMIL-VON-BEHRING GYMNASIUM

Einführung Nebenläufige Prozesse
Wichtige Begriffe



Warum Parallelität?

- Beschleunigung von Arbeitsvorgängen durch
  - Aufteilung in mehrere Prozesse
  - die teilweise gleichzeitig ablaufen
- Wartezeiten mit anderen Aktivitäten nutzen

- **Parallele Programmierung als Schlüssel zur Leistungsfähigkeit moderner Softwarearchitekturen!**

Anwendungsbereiche

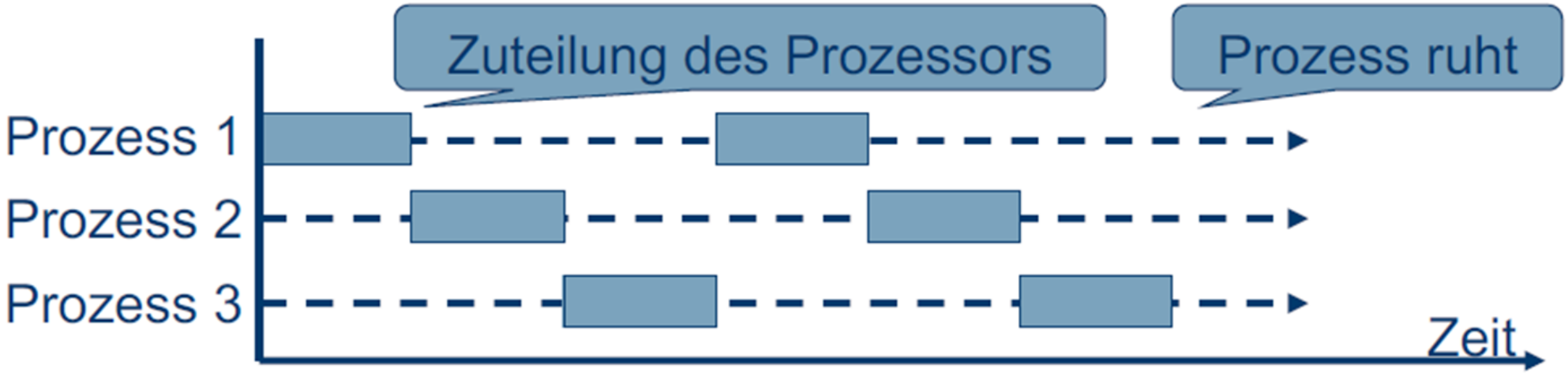
- Parallelität benötigt man immer, wenn bestimmte Ereignisse gleichzeitig ablaufen sollen.
  Beispiel: Musik während eines Spiels abspielen. Damit die Musik flüssig und durchgehend abgespielt werden kann und während des Abspielens das Spiel nicht pausieren soll, muss dies über Parallelisierung programmiert werden.
- Parallelität ist immer vorhanden, wenn mehrere Rechner involviert sind.
- Parallelität und Kommunikation sind die Zukunft.
  Beispiel: Jede Handy-App kann mittlerweile mit anderen Geräten/Internet kommunizieren und viele Apps nutzen damit auch bereits die Mehrkernprozessoren voll aus, die auch in jedem Handy vorhanden sind.

Wichtige Begriffe

- Prozess = ein im Speicher(zugriff) befindliches Programm mit seinen dafür nötigen Datenstrukturen mit Anwendungsdaten, Verwaltungsdaten (wie Methodenstapel, Programmzähler, …), Systemzustand (geöffnete Dateien, …)
- Ablaufplaner (scheduler) ist ein Programm/Bestandteil des Betriebssystems und teilt die Prozesse dem Prozessor bzw. Prozessoren zu, damit diese (abwechselnd) vorankommen.
- Thread/Aktivitätsfaden(-objekte) sind die logischen Einheiten/Objekte, die Code ausführen können. Jeder Prozess besitzt mindestens einen – oder auch mehrere – Threads.

Echt parallel vs. Quasi-parallel

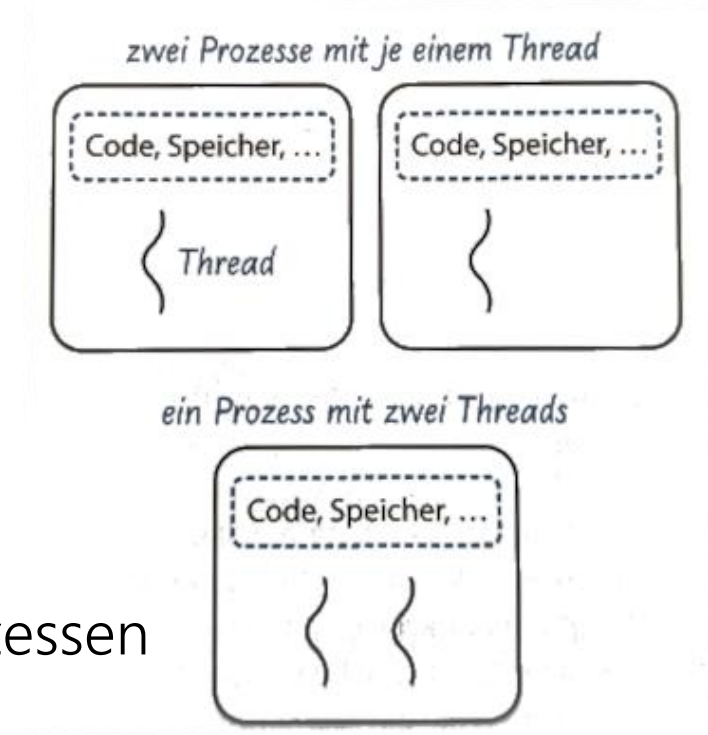
- Auf einem Rechner mit einem 1-Kern-Prozessor können auch mehrere Prozesse scheinbar gleichzeitig ausgeführt werden. Hier wird die Rechenleistung des Prozessors durch den Scheduler des Betriebssystems auf die Prozesse „verteilt“ (quasi-parallel).



Alle Prozesse scheinen gleichzeitig voranzukommen (wenn auch langsam).

Parallele Prozesse

- Schwergewichtiger Prozess (normaler Prozess im Betriebssystem)
  - Erhält eigenständiger Adressraum vom Betriebssystem
  - Dadurch werden Benutzer und Dienste getrennt
  - Prozesswechsel zeitaufwändig
  - Hat immer mindestens einen Thread
- Leichtgewichtiger Prozess (Thread)
  - Läuft im Adressraum eines schwergewichtigen Prozesses
  - Gemeinsame Nutzung von Variablen möglich
  - Schnellerer Wechsel als zwischen schwergewichtigen Prozessen



- In Java werden Threads durch die Klasse Thread realisiert. Eigene Unterklassen von Thread werden meist als Hilfsklassen deklariert.

Java: mehrere Klassen in einer Datei („Hilfsklassen“)

Aufgabe 1:

- Sieh dir die Beispielcodes zu den Varianten der Implementierung der Hilfsklassen an.

- Eine Datei kann mehrere Klassen enthalten, aber nach "außen" darf nur eine Public-Klasse sichtbar sein.
- In einer Public-Klasse dürfen allerdings weitere Public-Klassen implementiert sein.
- Anonyme Klassen werden beim Initialisieren auch definiert. Sie erweitern existierende Klassen ohne extends zu verwenden. Sie können aber nicht alleine existieren, sondern nur bestehende Klassen erweitern.