Algorithms (effizientere)

indem sie ↓ Speicher benötigen
↓ Rechenoperationen ausführen
besser Parallelisierung ausnutzen

"A major goal is to solve a problem with less computational work."

much of the progress in algorithms will come from three sources: (i) attacking new problem domains, (ii) addressing scalability concerns, and (iii) tailoring algorithms to take advantage of modern hardware.

1) New problem domains

ML, Cybersecurity, Robotik etc.

Wettbewerbsvorteile durch Algorithmen
- Google PageRank
- Google AdWords

Computational Biology: DNA Sequenzierungen durch dynamische Algorithmen

Bei Skalierungsproblemen: sublineare Algorithmen

2) New machine models

serial random-access machine model (24) originally developed in the 1960s and 1970s, which assumes that a processor can do only one operation at a time and that the cost to access any part of the memory is the same.

↳ viele Algorithmen sind nicht darauf optimiert Parallelität, Caching und Vektorisierung zu benutzen.

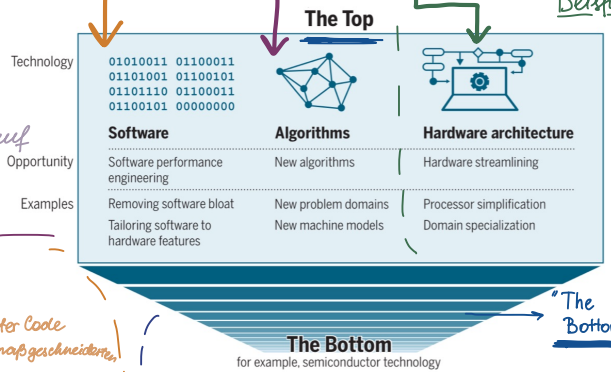
Hardware Architektur

"hardware streamlining" ≈ unnötige Teile entfernen

post-Moore era, architects will need to adopt the opposite strategy and focus on hardware streamlining: implementing hardware functions using fewer transistors and less silicon area.

"weniger Transistoren" pro CPU-Kern ① auf dem gleichen Chip können mehr Kerne platziert werden → mehr Kerne können parallel arbeiten → Leistungssteigerung
② ↓ Transistoren → ↓ Stromverbrauch → ↓ Kühlbedarf → spart Energie

neue Ansätze zur Leistungssteigerung



Performance gains after Moore's law ends. In the post-Moore era, improvements in computing power will increasingly come from technologies at the "Top" of the computing stack, not from those at the "Bottom", reversing the historical trend.

Beispiel: Domain Spezialisation → GPU

Domain: grafische Berechnungen

GPU's haben viele kleine processing units → massive parallele Verarbeitung

Was fehlt? Wieso streamlining?

Im Vergleich zu CPUs:

Komplexe Kontrolllogik:
- Keine **Branch Prediction** (Vorhersage von Verzweigungen im Code).
- Keine **Out-of-Order Execution** (flexibles Abarbeiten von Befehlen).
- Keine **Spekulative Ausführung** (Berechnung von möglichen Ergebnissen vorab).

Große Caches:
- GPUs haben kleinere Caches, da sie Daten oft direkt aus dem VRAM oder RAM streamen.

Multitasking und Kontextwechsel:
- GPUs können nicht schnell zwischen verschiedenen Aufgaben wechseln.
- CPUs haben **fortschrittliche Mechanismen** für Multitasking (z. B. Task Scheduling).

Allzweck-Befehlssatz:
- GPUs fehlen viele Befehle für allgemeine Aufgaben wie Dateioperationen, Betriebssystemverwaltung oder Interrupts.

Komplexe Speicherverwaltung:
- Keine oder eingeschränkte **virtuelle Speicherverwaltung**.
- CPUs haben umfangreiche Mechanismen wie Paging und Memory Protection.

Eingebaute Sicherheit:
- GPUs haben keine fortschrittlichen Sicherheitsfunktionen wie CPUs, z. B. für isolierte Prozesse.

Single-Thread-Leistung:
- GPUs sind schwächer bei der Bearbeitung einzelner Threads, da sie auf parallele Verarbeitung spezialisiert sind.

Software

"Software bloat" – ineffizienter schnell entwickelter Code
Entwickler verwenden existierende Lösungen statt maßgeschneiderten, Code zu schreiben – "reduction"

Even if each reduction achieves an impressive 80% efficiency, a sequence of two independent reductions achieves just 80% × 80% = 64%.

"Tailoring software to hardware features" Software an die Architektur der Hardware anpassen

Performance engineering:

Simply choosing a more efficient programming language speeds up this calculation dramatically.

The price for this performance gain is programmer productivity: Coding in C is more onerous than coding in Python, and Java lies somewhere in between.

"The Bottom" = Verbesserung auf Hardware Ebene
↓
stößt auf ihre Grenzen!

Wieso endet Moore'sche Gesetz?

Why is miniaturization stalling? It's stalling because of fundamental physical limits—the physics of materials changes at atomic levels—and because of the economics of chip manufacturing. Although semiconductor technology may be able to produce transistors as small as 2 nm (20 Å), as a practical matter, miniaturization may end around 5 nm because of diminishing returns (10). And even if semiconductor technologists can push things a little further, the cost of doing so rises precipitously as we approach atomic scales (11, 12).

je kleiner Transistoren desto höher die Kosten!
gleichzeitig nimmt Leistungsgewinn ab
↓
mehr Aufwand bringt weniger Vorteile!

* theoretisch könnte die Miniaturisierung bis zu 2 nm gehen