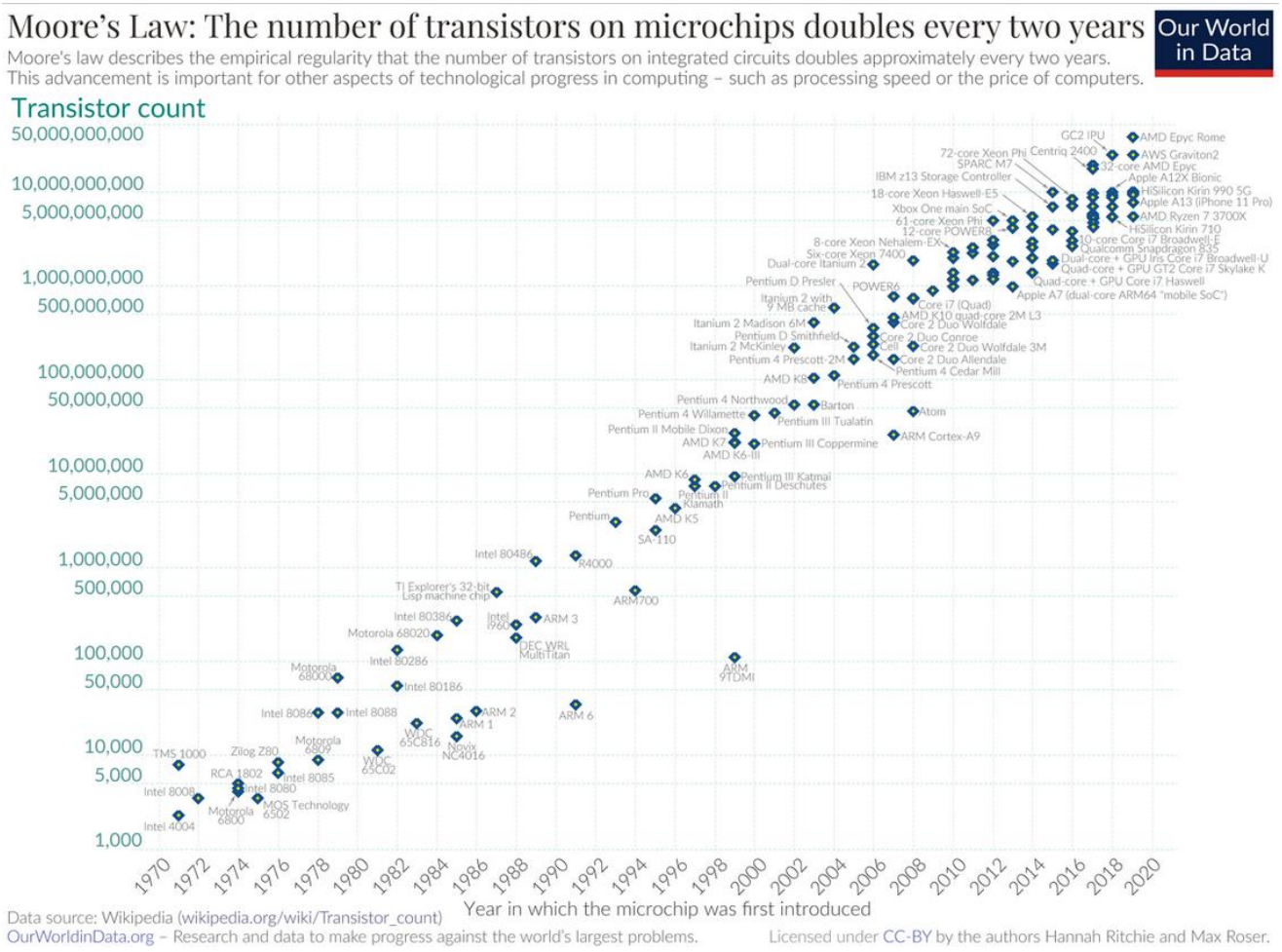
Betriebssysteme

Kapitel 8 Multiprozessorsysteme

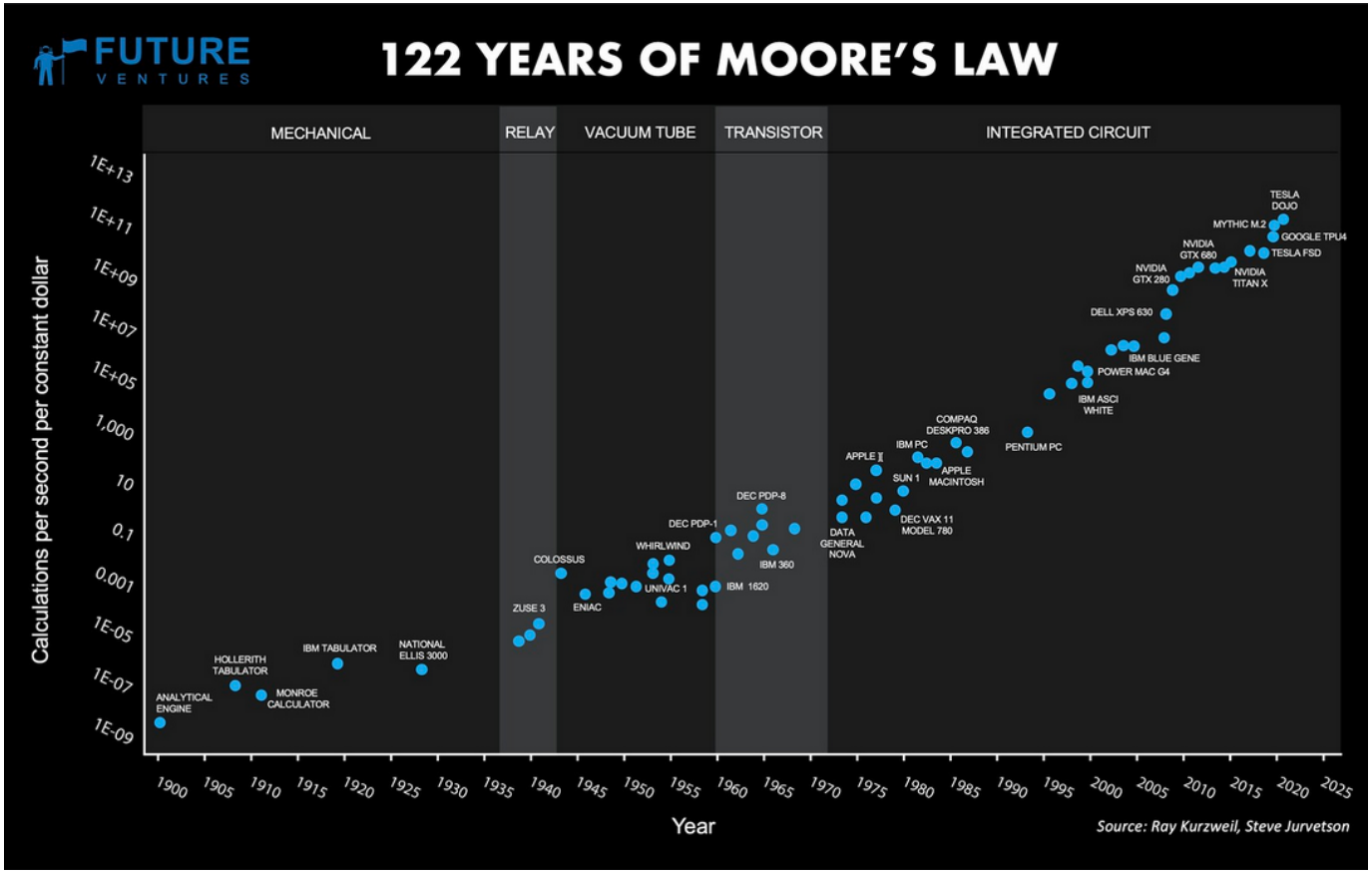
Moore’s Law: Rechenleistung verdoppelt sich alle 12-24 Monate (technisch: im betrachteten Zeitraum verdoppelt sich die Anzahl der Transistoren etwa alle zwei Jahre)



- Single-Core Systeme erreichen Technologie-grenzen
- Probleme: Verlustleistung, physikalische Grenzen
- 2010: Koomey’s Law misst die Anzahl der Berechnungen pro Joule dissipierter* Energie

Dissipation (lat.) =Zerstreuung; (hier verlorene Wärme)

Steve Jurvetson: Tesla Has Done Something No Other Automaker Has: Assumed The Mantle Of Moore's Law (08/2021)



Tesla introduced its D1 chip for the DOJO Supercomputer.

Steve Jurvetson: Tesla sorgt mit KI-Chip für Weiterbestehen von Moore's Gesetz

Tesla AI Day: https://www.youtube.com/watch?v=j0z4FweCy4M&t=6340s

https://www.flickr.com/photos/jurvetson/51391518506/

Rechenleistung

- Wunsch:
  - immer mehr CPU-Zyklen bzw.
  - immer höhere Prozessorgeschwindigkeiten

- Vergangenheit: Erhöhung Prozessortakt
- Einstein spez. Relativitätstheorie: kein elektrisches Signal kann sich schneller fortbewegen als das Licht, das bedeutet
  - Lichtgeschwindigkeit in Vakkum **30cm/ns** und in Kupfer 20 cm/nsec
  - Beispiel (Signal innerhalb eines Zyklus von einem zum anderen Ende):
    10 GHz-CPU → ein Takt entspricht 2 cm Laufweg
    100 Ghz-CPU → 2 mm Laufweg
    1000 Ghz-CPU → 0,2 mm Laufweg
  - Entwicklung eines solch kleinen Computers ist möglich, aber das Problem ist die **Hitze**.
  - Je schneller der Computer ist, desto größer ist die Hitzeabstrahlung; je kleiner er ist, desto schwieriger ist die Hitzeableitung (siehe auch Kühler der x86-Systeme)

Lösungsansatz für mehr Rechenleistung

- Nicht die Geschwindigkeit erhöhen, sondern die **Parallelisierung**
  - Verwendung von sehr vielen Prozessoren gleichzeitig (jede einzelne hat eigene, ‚**normale**‘ Gechwindigkeit)
  - Zusammen haben sie viel mehr Rechenleistung als eine einzelne CPU
  - Beispiel:
    - HPC Parallelrechner (HPC=High Performance Computer)
    - Clusterrechner (Server-Cluster, Storage-Cluster, Datenbank-Cluster, usw.
    - Grid-Systeme (viele lose über Netzwerke miteinander verbundene Rechner stellen ungenutzte Kapazität für Anwendungen mit hohem Ressourcenbedarf zur Verfügung)
    - Mehr-Kern-Prozessoren (Multi-Cores)
  - Anwendungsbeispiele (lange Rechenzeiten mit vielen unabhängigen CPUs):
    - Wettervorhersage
    - Modellierung des Luftstrom um Flugzeugflügel
    - Simulation der Weltwirtschaft
    - Hochrechnung für Lebensversicherungen
- Schnelle Entwicklung des Internets ermöglichte das Verbinden von Tausenden von Rechnern weltweit

Lösungsansatz für mehr Rechenleistung: Parallele Architekturen

- **Cluster:**
  Mehrere unabhängige Rechner, nur durch Netzwerk verbunden
- **Multi-Prozessor:**
  Mehrere CPUs auf einer Hauptplatine
  - Weniger effiziente (ältere) Variante von Multi-Core-Systemen
  - Betriebssystem sieht mehrere echte CPUs
- **Hyper-Threading:**
  mehrere logische CPUs in einem CPU-Chip (auch kombinierbar mit Multi-Core)
  - Multi-Threaded-CPUs sind mehrfädige (engl. multithreading) Prozessor- kerne mit mehreren Programmzählern und Registersätzen. Sie melden sich gegenüber dem System als mehrere Kerne.
  - Intel nennt sie in einigen Prozessorlinien Hyper-Threading, AMD Simultanous Multithreading (SMT)
  - IBM SMT nennt sie Symmetrisches Multi-Threading (SMT) z.B. IBM Power5-Prozessor ist z. B. ein Doppelkernprozessor mit zwei Threads pro Kern
- **Multi-Core:**
  Mehrere Cores in einem CPU-Chip

Lösungsansatz für mehr Rechenleistung: Parallele Architekturen

- **Multi-Core-Prozessor**
  - Mehrkernprozessoren (auch Multicore-Prozessoren oder Multikernprozessoren) bezeichnen einen Mikroprozessor/Prozessor mit mehr als einem vollständigen Hauptprozessor auf einem einzigen Chip. Sämtliche Ressourcen mit Ausnahme des Bus und eventuell einiger Caches sind repliziert. Es handelt sich also um mehrere vollständige.
  - Mikroprozessoren mit **einem Hauptprozessor** sind als Einzelkernprozessor (auch Single-Core-Prozessor) bekannt.
  - Dual-Core-Prozessor ist ein Mehrkernprozessor mit zwei Hauptprozessoren.
  - Vierkernprozessor ist Quad-Core-Prozessor…
  - Grund der Entwicklung von Mehrkernprozessoren: Kosten
    Mit der gleichen Anzahl an Chip-Sockeln und Chips kann theoretisch eine vervielfachte Rechenleistung erzielt werden (das n-fache bei n Kernen)
    → abhängig von Parallelisierbarkeit der Software

Aufgabe/Frage

Annahme1: Es stehen 1000 Computer in einem Raum
Annahme2: Es stehen 1000 Computer auf der ganzen Welt verteilt

Es gibt sicherlich technische Unterschiede zwischen den Installationen, welche zwei Unterschiede sind sofort erkennbar?

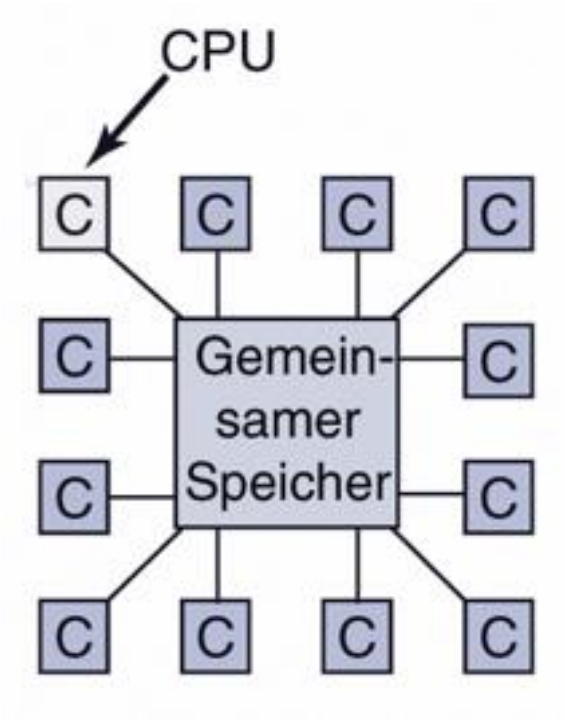
Bedingung:
→ ad hoc

Kommunikation zwischen elektronischen Komponenten

- Kommunikation beruht auf Senden von Nachrichten (wohldefinierte Bit-Zeichenketten)
- Unterschiede liegen:
  - Im Zeitmaßstab
  - Im Entfernungsmaßstab
  - In der verwendeten logischen Organisation

Kategorisierung der Multiprozesssorsysteme

- Implementierung von Multiprozessorsystem mit gemeinsamem Speicher
  - Mit physisch getrennten CPUs
  - Mit mehreren Kernen auf einer CPU oder
  - durch Kombination der beiden
- Multiprozessorsystem mit gemeinsamem Speicher

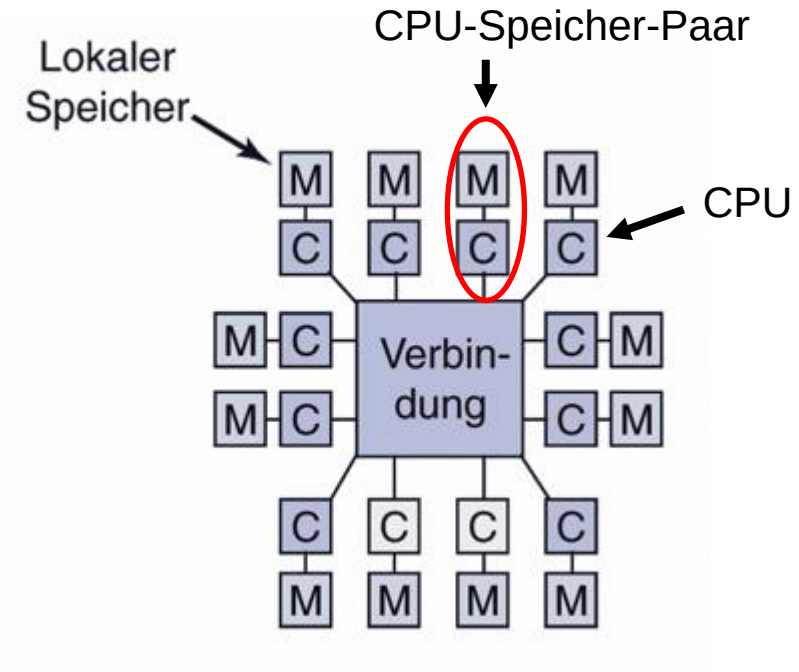


- Jede CPU hat gleichermaßen Zugriff auf den gesamten physischen Speicher
- Einzelne Wörter werden mit LOAD und STORE-Befehlen gelesen bzw. geschrieben
- Zugriff auf Speicherwort benötigt i.d.R. zwischen 1 und 10 nano-sec
- Intensiver Nachrichtenaustausch notwendig

- **Heute: Mehr als ein Prozessorkern auf einem CPU-Chip**
  - **Kerne haben gemeinsamen Zugriff aufs RAM**
  - **manchmal Teilung des Caches**

Kategorisierung der Multiprozesssorsysteme

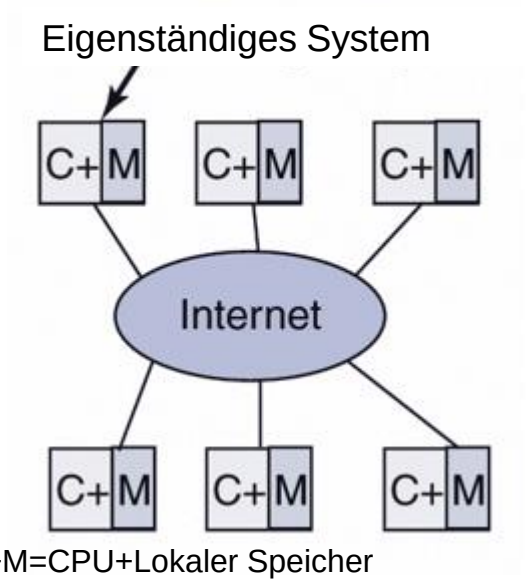
- Multicomputersystem mit Nachrichtenaustausch



- CPU-Speicher-Paare sind durch ein Hochgeschwindigkeitsverbindung untereinander verknüpft
- Jede Speichereinheit ist lokal mit einer CPU verbunden
- Zugriff nur über diese CPU
- CPUs kommunizieren mit Mehrwort-Nachrichten über die Hochgeschwindig-keitsverbindung
- Kurze Nachricht kann in 10 bis 50 micro-sec verschickt werden
- Kein gemeinsamer Speicher
- Leichter zu realisieren als Multiprozessorsystem mit gemein-samem Speicher, Aber schwerer zu programmieren
- **Enge Kopplung (tightly coupled)**

Kategorisierung der Multiprozesssorsysteme

- Großräumig verteiltes System



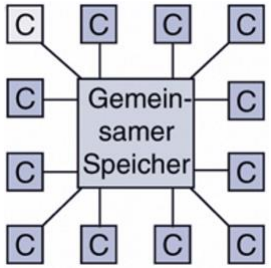
- Verbindung ganzer Computeranlagen über ein WAN, z.B. Internet
- **Verteiltes System**
- Jeder Computer hat seinen eigenen Speicher
- **Kommunikation der Systeme durch Nachrichtenaustausch**
- Senden der Nachrichten von 10 bis 100 micro-sec
- Verzögerung bedingt eine lose Kopplung **(loosely coupled)**

Unterschied in der Länge der Zeitverzögerung bei den drei Modellen um Faktor 1000

Kategorisierung der Multiprozesssorsysteme

- Multiprozessorsysteme mit gemeinsamem Speicher
  - Multiprozessor-Betriebssysteme sind ‚normale' Betriebssysteme
    - Behandlung von Systemaufrufen
    - Verwaltung von Speicher,
    - Verfügbarmachen eines Dateisystems
    - Steuerung von Ein-/Ausgabegeräten
  - Sie haben auch einzigartige Eigenschaften
    - Synchronisation von Prozessen
    - Die Ressourcenverwaltung
    - das Scheduling

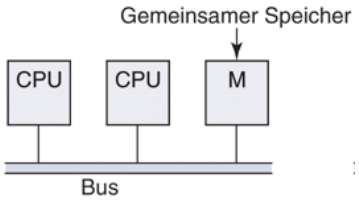
Kategorisierung der Multiprozesssorsysteme



- Multiprozessorsysteme mit gemeinsamem Speicher
  - Shared Memory System
  - Zwei oder mehr CPUs teilen sich ein gemeinsames RAM
  - Kommunikation erfolgt über diesen Adressraum
  - Jede CPU sieht den gesamten virtuellen Adressraum
    - Eine CPU schreibt Daten in den Speicher und eine andere liest die Daten aus → **Grundlage für Interprozesskommunikation**
    - **UMA-Multiprozessoren**: Speicherzugriff auf alle Speicherbereiche gleich schnell (UMA=**Uniform** Memory Access)
    - **NUMA-Multiprozessoren** (sprich non-UMA, **Non-Uniform** Memory Access)

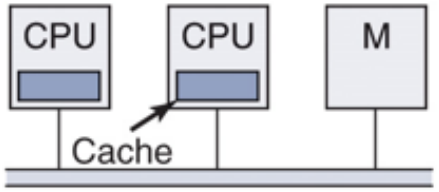
Kategorisierung der Multiprozesssorsysteme

- Multiprozessorsysteme mit gemeinsamem Speicher
  - UMA und Bus-basierte Architekturen
    - Zwei oder mehrere CPUs benutzen gemeinsam denselben Bus zur Kommunikation
    - Prüfen vor Speicherzugriff, ob der Bus belegt ist
      - Falls frei, legt CPU die Speicheradresse auf den Bus, fügt ein paar Steuersignale hinzu und wartet auf den Inhalt
      - Falls Bus belegt, wartet CPU bis Bus frei ist
    - **Wettstreit um den Bus**
      - Bandbreite des Buses beschränkt das System
      - CPUs meist im Leerlauf (→ Beschränkung Anzahl CPU < 32)



Kategorisierung der Multiprozesssorsysteme

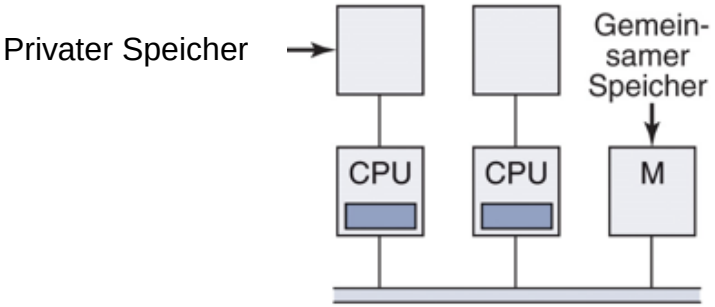
- Multiprozessorsysteme mit gemeinsamem Speicher
  - **UMA und Bus-basierte Architekturen**
    - **Lösung: Reduzierung Zugriff auf Bus**
      - **I. Cache für jeden Prozessor**



        - Ort:
          - Innerhalb des CPU-Chips
          - Neben dem CPU-Chip
          - Auf Hauptplatine
        - Viele Lesezugriffe aus dem lokalen Cache → wenige Buszugriffe
        - Wenn Wort referenziert, dann laden eines ganzen Blocks **(Cache-Line)** in den Cache der betreffenden CPU
        - Management vom Cache notwendig
          - Sicherstellung, daß bei Mehrprozessorsystemen mit mehreren CPU-Caches verhindert wird, dass die einzelnen Caches für dieselbe Speicheradresse unterschiedliche (inkonsistente) Daten zurückliefern **(Cache-Kohärenz)**
            - Markieren Cache-Block = nur zum Lesen (in mehreren Caches gleichzeitig) ODER
            - Markieren Cache-Block = zum Lesen und Schreiben (ausschließlich in einem Cache)
            - **Beispiel:**
            - **Schreibversuch** auf Wort, das in einem oder mehreren fremden Caches liegt
              - Bushardware erkennt dies **und informiert alle Caches über den Schreibvorgang**
              - Gibt es unveränderten Cache-Block, → ok,
              - Gibt es veränderte Cache-Blöcke → zurückschreiben in Speicher oder direkter Transfer zum Schreiber bevor der Schreibvorgang fortgesetzt wird
                → **Regeln heißen: Cache-Kohärenz-Protokoll**

Kategorisierung der Multiprozesssorsysteme

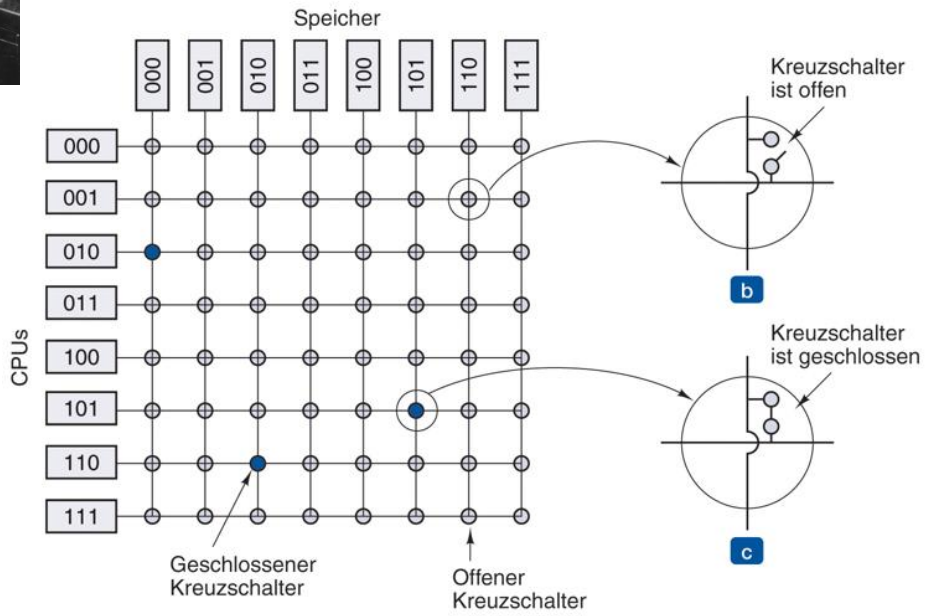
- Multiprozessorsysteme mit gemeinsamem Speicher
  - **UMA und Bus-basierte Architekturen**
    - **Lösung: Reduzierung Zugriff auf Bus**
      - **II. Privater Speicher für jeden Prozessor**
        - Jede CPU hat nicht nur einen Cache sondern auch lokalen privaten Speicher
        - Dedizierter Zugriff über einen dedizierten (privaten) Bus
        - Privater Speicher nur für lokale Variablen, Programmtext, Zeichenketten, Konstanten, die nur gelesen werden
        - Unterstützung durch Kompiler notwendig



Kategorisierung der Multiprozesssorsysteme

- Multiprozessorsysteme mit gemeinsamem Speicher
  - **UMA unter Verwendung von Kopplungsfeldern (crossbar switch)**

  - siehe auch Netze in Telefonvermittlungszentralen zum Verbinden von Mengen von eingehenden mit einer Menge von ausgehenden Leitungen
  - Einfache Art n CPUs mit k Speichermodulen zu verbinden → *Koppelfelder*
  - spezielle Hardware auf Chip

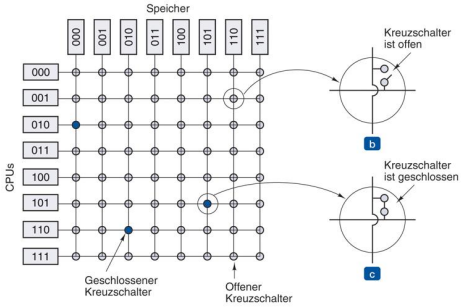


- An jeder Kreuzung von einer horizontalen (eingehenden) und einer vertikalen (ausgehenden) Leitung befindet sich ein Kreuzschalter.
- Kreuzschalter=elekr. Schalter
- Der Schalter wird elektronisch geöffnet oder geschlossen, je nachdem eine horizontale oder vertikale Leitung verbunden werden soll

https://de.rbth.com/geschichte/84341-telefone-russland-geschichte

Kategorisierung der Multiprozesssorsysteme

- Multiprozessorsysteme mit gemeinsamem Speicher
  - **UMA unter Verwendung von Kopplungsfeldern (crossbar switch)**
    - spezielle Hardware auf Chip
      - Vorteile:
        - Mehr als 32 CPUs
        - Nicht blockierende Zugriffe
          - Keiner CPU wird jemals eine benötigte Verbindung zu einem Speichermodul verwehrt, weil ein Kreuzschalter oder eine Leitung bereits besetzt ist
          - Jede CPU kann auf den zugeteilten Speicher zugreifen
          - Wettstreit ist möglich, wenn zwei CPUs gleichzeitig auf dasselbe Speichermodul zugreifen wollen
            → Aufteilung des Speichers in n Einheiten reduziert die Konkurrenz-Situation um Faktor n im Vergleich zu den busbasierten Architekturen
      - Nachteil:
        - Anzahl der Kreuzschalter wächst mit n hoch 2 (für 1000 CPUs und 1000 Speichermodule → 1 Mill. Kreuzschalter
        - **Nicht praktikabel**
        - nur für mittelgroße Systeme realisierbar (<100)



Kategorisierung der Multiprozesssorsysteme

- Multiprozessorsysteme mit gemeinsamem Speicher
  - **UMA mit Mehrstufigen Schaltnetzwerken (multistage switching network)**
    - spezielle Hardware auf Chip
    - Schalter hat 2 Ein- und 2 Ausgänge
    - Eingehende Nachrichten können jeder ausgehenden Leitung zugeteilt werden

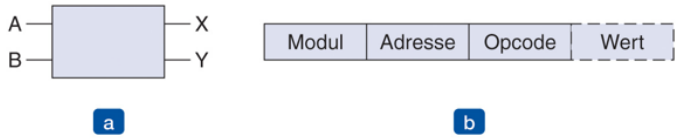


Abbildung 8.4: (a) 2 × 2-Schalter mit zwei Eingangsleitungen, *A* und *B*, und zwei Ausgangsleitungen, *X* und *Y* (b) Nachrichtenformat

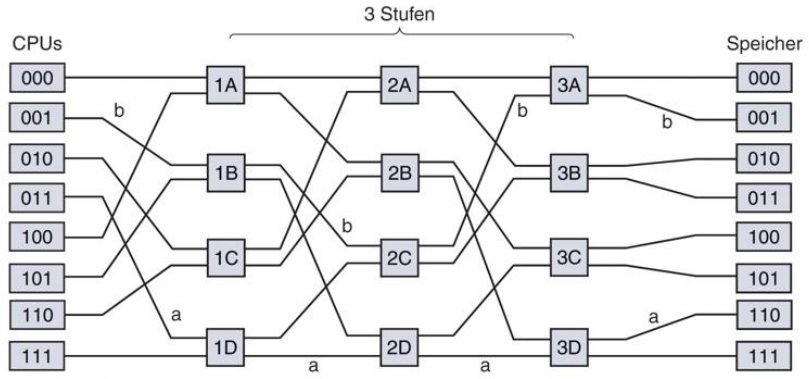


Abbildung 8.5: Omega-Schaltnetzwerk

- **Beispiel: Mit zwölf Schaltern werden acht CPUs mit 8 Speichermodulen verbunden**
- CPU 011 will ein Speicherwort von 110 lesen
- CPU sendet einen READ-Befehl an den Schalter 1D, der den Wert 110 im Modulfeld enthält
- Schalter verarbeitet das erste Bit (links) von 110 und verzweigt nach 2D **(0 verzweigt nach oben, 1 nach unten)**
- Schalter 2D verarbeitet das zweite Bit und zweigt aufgrund der 1 nach unten → 3D
- Schalter 3D verarbeitet das dritte Bit und zweigt aufgrund der 0 nach oben ab → 110

- Nach Verarbeitung durch das Netzwerk wird das am weitesten links stehende Bit nicht mehr benötigt.
- Speicherstelle wird benutzt, um die ankommende Leitung aufzuzeichnen, so dass die Antwort den Weg zurückfindet.
- Ergebnis: Pfad a = 0, 1, 1

Aufgabe/Frage

Wir haben den Pfad a gerade durchgespielt.
Zusätzlich will CPU 001 ein Wort in das Speichermodul 001 schreiben.

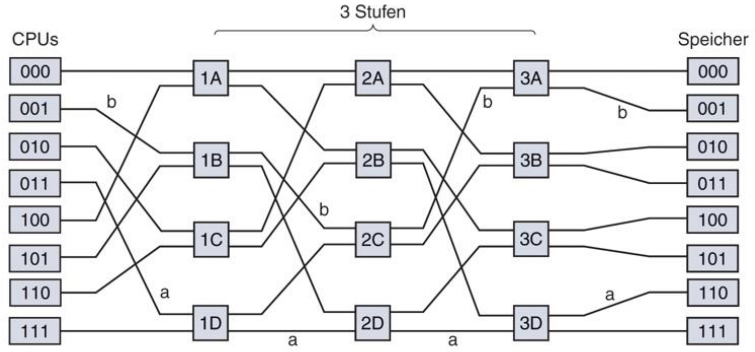


Abbildung 8.5: Omega-Schaltnetzwerk

Frage 1:Wie sieht der Pfad nach 001 aus?

Frage 2: Was passiert, wenn gleichzeitig CPU000 auf den Speicher 000 zugreifen will?

Bedingung:
→ 5 min

5 min

Kategorisierung der Multiprozesssorsysteme

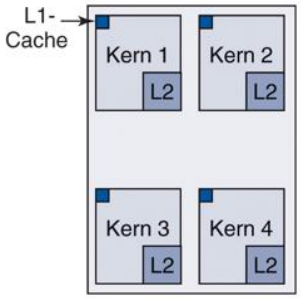
- **Multiprozessorsysteme mit gemeinsamem Speicher**
  - **NUMA (NON Uniform Memory Access)**
    - Nicht alle Speichermodule haben die gleiche Zugriffszeiten
    - Drei Schlüsseleigenschaften von NUMA Systemen
      1. Ein einziger Adressraum, der für alle CPUs sichtbar ist
      2. Zugriff auf entfernten Speicher mittels der Befehle LOAD und STORE
      3. Zugriff auf entfernten Speicher ist langsamer als auf lokalen Speicher
    - Zwei Varianten
      - NC-NUMA = No-Cache-Coherent NUMA, Zugriffszeit auf entfernten Speicher nicht durch Caches verborgen (z.B. kein Cache vorhanden)
      - CC-NUMA == Cache-Coherent-NUMA
    - Ansatz für große CC-NUMA-Systeme
      - Grundgedanke: Verwaltung einer Datenbank als Verzeichnis, in dem jede Cache-Line mit ihrem status geführt ist.
        - **Verzeichnisbasierte Multiprozessorsysteme = directory based multiprocessor**
      - Nachschauen in der Datenbank bei jedem Zugriff auf den Cache, wo sich Cache-Line befindet und ob unverändert (clean) oder modifiziert (dirty)
      - Extrem schnelle Hardware notwendig, die innerhalb eines Bruchteils eines Buszyklus antworten kann

Kategorisierung der Multiprozesssorsysteme

- Multicore-Systeme (Mehrkernsysteme)
  - Immer mehr Transistoren auf einem einzigen Chip (Gordon Moore, Mitbegründer von Intel)
  - Erinnerung:
    - Multiprozessoren: Rechner mit mehreren Prozessoren
    - Multikernprozessoren: Prozessoren mit mehreren Kernen (cores)
    - Multicomputer: Zusammenschluss mehrerer Rechner über eine Netzwerk
  - Mehrkernchips sind **Kleine Multiprozessor-Systeme**
    - Jeder physische Kern hat L1-Befehlscache und einen L1-Datencache (>= 32 KB)
    - Jeder physische Kern hat L2-Cache (>=256 KB)
    - Jeder physische Kern hat einen L3-Cache (>= 30MB)
    - Arbeitsspeicher wird geteilt
      - Falls ein Wort in zwei oder mehreren Cache-Speichern liegt und eine der CPUs dieses Wort ändert, wird es durch spezielle Hardwareschaltkreise automatisch aus all den anderen Caches gelöscht, um konsistent zu sein (Snooping).

Kategorisierung der Multiprozesssorsysteme

- Multicore-Systeme (Mehrkernsysteme)
  - CMP (Chip-Level-Multiprozessor)
    - Jede CPU eines busbasierten Multiprozessors hat ihren eigenen Cache (z.B. AMD)



  - Problem:
    - Entwickeln von Software
      - Parallele Programmierung
      - Aufteilung von Tasks auf mehrere Pakete
    - Race Conditions und Deadlocks
      - werden zu Alpträumen
      - Semaphore sind keine Lösungen
    - Es gibt neue Anforderungen an Betriebsysteme
      → Betriebssysteme für Multiprozessoren

Betriebssystemarten für Multiprozessoren

- Jede CPU hat eigenes Betriebssystem

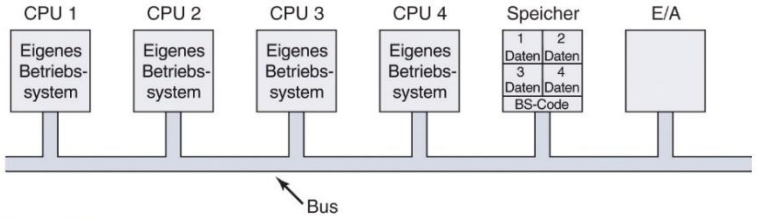


Abbildung 8.7: Aufteilung des Speichers auf vier CPUs unter Verwendung nur einer einzigen Kopie des Betriebssystemcodes. Kästen, die mit Daten beschriftet sind, stellen die privaten Daten des Betriebssystems für jede CPU dar.

- Unterteilung des Speichers in so viele Partitionen, wie es CPUs gibt
- Jede CPU erhält Ihren eigenen Speicher und ihre eigene Kopie des Betriebssystems
- N CPUs arbeiten wie n unabhängige Computer
- Nachteile:
  - Löst ein Prozess einen Systemaufruf aus, wird dieser von der eigenen CPU abgefangen und unter Verwendung der Datenstrukturen in den eigenen Tabellen des BS behandelt
  - Jedes BS hat seine eigenen Tabellen, damit hat es auch seine eigene Prozessmenge (Prozesse können die CPUs nicht verlassen)
  - Physische Speicherseiten werden nicht gemeinsam genutzt (z.B. CPU1 hat freie Speicherseiten, CPU2 lagert ständig aus)
    → **Heute: keine Verwendung mehr**

Betriebssystemarten für Multiprozessoren

- Master-Slave-Multiprozessoren

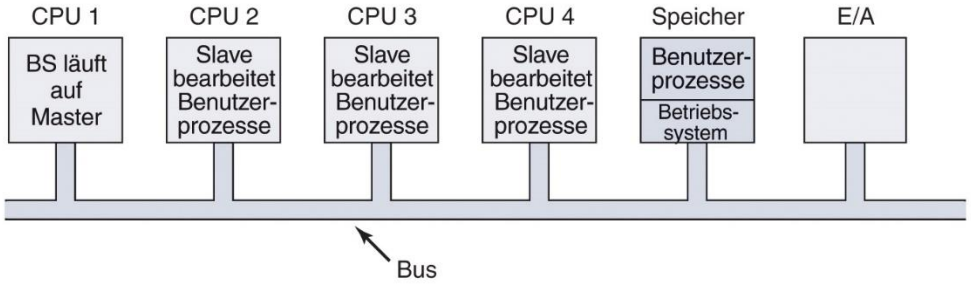


Abbildung 8.8: Master-Slave-Multiprozessormodell

- Kopie des Betriebssystems und seiner Tabellen befindet sich nur auf CPU 1
- Systemaufrufe werden auf CPU 1 umgeleitet und dort bearbeitet
- Vorteile:
  - Eine einzige Datenstruktur (Liste oder Menge von Listen) für Verwaltung von Prozessen
    - → Die Einschränkungen von ‚Jede CPU hat eigenes BS' verschwinden
- Nachteile:
  - Master (CPU 1) wird zum Engpass
    - Annahme: 10% der Rechenzeit zur Behandlung von Systemaufrufen
    - 10 CPUs lasten die CPU 1 aus, 20 CPUs überlasten die CPU 1
      - → **Kleine Multiprozessorsysteme, für Große nicht**

Betriebssystemarten für Multiprozessoren

- **SMP Multiprozessoren (Symmetric Multiprocessor)**

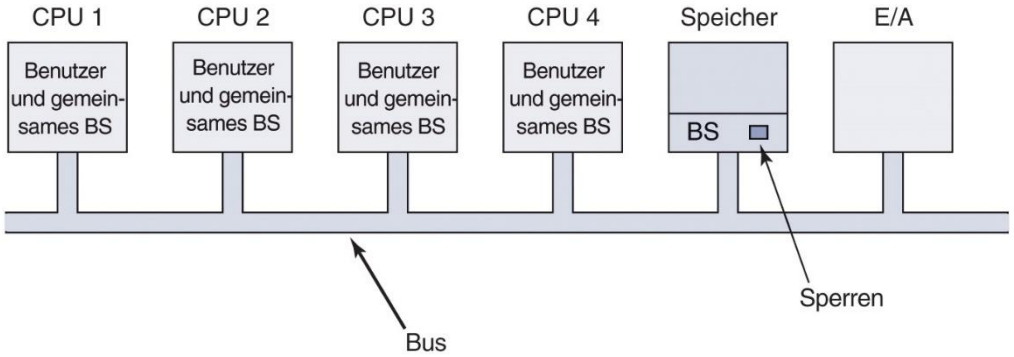


Abbildung 8.9: Das SMP-Multiprozessormodell

- Es gibt nur eine Kopie des Betriebssystems im Speicher; Jede CPU kann sie ausführen
- Jede CPU kann bei Systemaufruf in Kernmodus wechseln und auf Speicher zugreifen
- Es gibt nur ein Satz an Betriebssystemtabellen
- Umgehung des Master-Engpass erzeugt neue Probleme
  - Es gibt jetzt einen Wettstreit um die Ressource Betriebssystem
    - Mutex sperrt das ganze BS als eine kritische Region → Jede CPU kann wohl das Betriebssystem ausführen, jedoch immer nur eine zu einer Zeit
  - **Verbesserung:** Unterteilung des BS in verschiedene unabhängige kritische Regionen (z.B. Scheduler, Systemaufruf des Dateisystems, Verwaltung Seitenfehler)
    → **Moderne Multiprozessorsystem benutzen diese Aufteilung**

Betriebssystemarten für Multiprozessoren

- **SMP Multiprozessoren (Symmetric Multiprocessor)**
  - Die meisten modernen Multiprozessorsysteme benutzen Aufteilung in unabhängige verschiedene kritische Regionen
  - Schwierig:
    - Aufteilen des Betriebssystems in die kritischen Regionen, die parallel von verschiedenen CPUs ausgeführt werden können ohne sich gegenseitig zu stören
    - Jede Tabelle, die von mehreren kritischen Regionen benutzt wird, muss durch einen Mutex geschützt werden
    - Jeder Code, der diese Tabelle benutzt, muss den Mutex richtig verwenden
    - Deadlocks müssen vermieden werden
  - Weitere Funktionen, die notwendig sind:
    - Synchronisation der CPUs in einem Multiprozessor
    - Multiprozessor-Scheduling
    -
    -

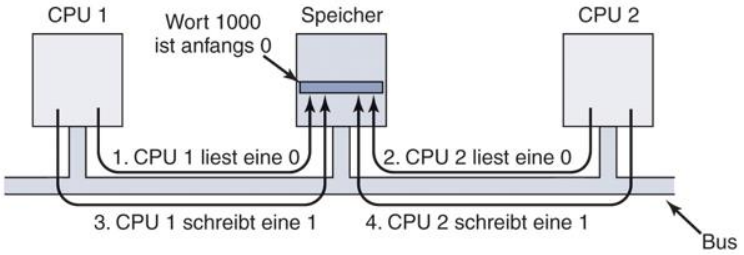
Synchronisation in Multiprozessor-Systemen

- Erinnerung: Auf Einprozessor-Machinen kann man einfach die Interrupts sperren, um ausschließlichen Zugriff auf kritische Kerntabellen bei Systemaufrufen zu ermöglichen.
- Bei Multiprozessor-Systemen betrifft diese Sperrung nur die CPU, von der aus der Aufruf kommt. Andere CPUs können weiterhin auf kritische Tabellen zugreifen
- **Lösung:** Verwendung geeignetes Mutex-Protokoll, das von allen CPUs respektiert wird, um wechselseitigen Ausschluss zu gewährleisten.
  - Spezieller Befehl für Lesen und Schreiben in einer einzigen atomaren Operation ist TSL (Test and Set Lock)
  - TSL liest ein Speicherwort aus und speichert seinen Inhalt in einem Register. Gleichzeitig schreibt er eine 1 (oder was anderes ungleich 0) ins Speicherwort
  - Es werden 2 Buszyklen fürs Lesen und Schreiben benötigt

Synchronisation in Multiprozessor-Systemen

- Beispiel:
  - Speicherwort 1000 (wird als Sperre verwendet) hat anfangs den Wert 0
  - Schritt 1: CPU 1 liest das Wort und erhält eine 0
  - Schritt 2: CPU 2 liest auch das Wort zu 0 aus, noch bevor CPU 1 das Wort zurückgeschrieben und auf 1 gesetzt hat
  - Schritt 3: CPU 1 schreibt nun 1 in das Wort.
  - Schritt 4: CPU 2 schreibt auch eine 1 ins Wort
  - Beide CPUs bekamen vom TSL-Befehl eine 0 zurückgeliefert, beide haben nun Zugang zur kritischen Region

    → Wechselseitiger Ausschluss ist fehlgeschlagen



  - Lösung: Der TSL (Test and Set Lock) Befehl wird verändert.
    - TSL sperrt zuerst den Bus, um anderen den Zugriff zu verwehren
    - Beide Speicherzugriffe werden ausgeführt
    - Bus wird wieder freigegeben

Scheduling in Multiprozessor-Systemen

- Problem:
  - **Erinnerung Einprozessor-Systeme:** Welcher Thread/Prozess soll als nächstes ausgeführt werden?
  - **Multiprozessor-Systeme:** Welcher Thread/Prozess soll auf welcher CPU als nächstes ausgeführt werden?
    → Zwei-dimensionales Problem!
    → Zusätzlich könnten die Threads noch voneinander abhängen

- Time-Sharing
  - Scheduler behandelt unabhängige Threads (später abhängige Threads)
    - Nutzung einer systemweiten Datenstruktur für rechenbereite Threads
    - Die Datenstruktur ist vielleicht eine einfache Liste, wahrscheinlich eine Menge von Listen von Threads
    - Wie bei Einprozessor-System, aber….

Scheduling in Multiprozessor-Systemen

- Time-Sharing
  - Beispiel:
    - Die Datenstruktur ist eine Menge von Listen für Threads mit verschiedenen Prioritäten
    - 16 CPUs sind aktuell beschäftigt
    - Eine priorisierte Menge von 14 Threads (A-N) wartet auf die Ausführung
    - CPU 4 beendet ihre momentane Arbeit
    - CPU 4 sperrt die Scheduling Warteschlangen und wählt den Thread mit höchster Prio aus **(A)**
    - Als nächstes wird CPU 12 frei und wählt B aus
    - Wenn Threads unabhängig voneinander sind → OK

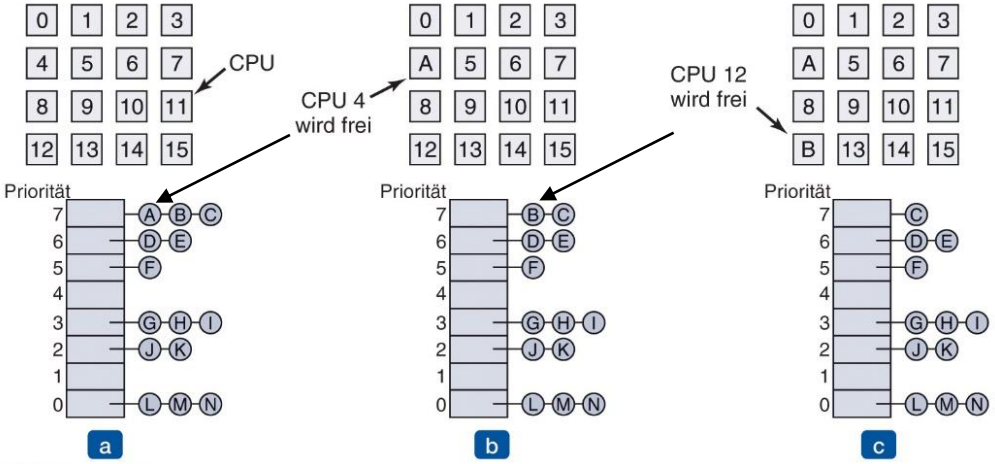


Abbildung 8.12: Verwendung einer einzigen Datenstruktur für das Scheduling auf Multiprozessorsystemen

Scheduling in Multiprozessor-Systemen

- Time-Sharing
  - Unabhängige Threads
    - Nachteile
      - Wettstreit um die Datenstruktur bei wachsender Anzahl von CPUs
      - üblicher Aufwand für Kontextwechsel, wenn ein Thread einer Ein-/Ausgabeoperation blockiert wird
        - Hier eventuell über CPU Grenzen hinweg
      - Kontextwechsel kann auch auftreten, wenn ein Thread sein Zeitquantum überschreitet
    - Beispiel 1:
      - Ein Thread auf einem Multiprozessor hält eine Sperre, wenn sein Quantum abläuft
      - Andere Threads warten auf Freigabe dieser Sperre
        - Diese Threads verschwenden ihre Zeit mit unnötigem Warten bis der Scheduler den Thread erneut auswählt und Sperre freigibt
      - **Lösung: Smart Scheduling**
        - Einem Thread, der aktuell eine Sperre hat, wird nicht angehalten, sondern es wird ihm mehr Zeit gegeben, um die kritische Region zu verlassen

Scheduling in Multiprozessor-Systemen

- Time-Sharing
  - Unabhängige Threads
    - Beispiel 2:
      - Ein Thread läuft einige Zeit auf CPU x
        → viele Blöcke für diesen Prozess sind im Cache von CPU x
      - Kontext-Wechsel auf die CPU y ist teuer, da der Cache von CPU x auf CPU y transferiert werden muss

      - **Lösung: Affinity-Scheduling**
        - Affinity Scheduling versucht den Prozess auf der gleichen CPU laufen zu lassen, auf der er vorher schon lief

Scheduling in Multiprozessor-Systemen

- Space-Sharing
  - Scheduling mehrerer Threads zur gleichen Zeit und über mehrere CPUs hinweg nennt man Space-Sharing
  - Wenn Threads voneinander abhängen (z.B. viel miteinander kommunizieren), ist es sinnvoll diese gleichzeitig laufen zu lassen
  - Es kommt häufig vor, daß zu einem einzelnen Prozess mehrere Threads gehören, die zusammenarbeiten.
    - Beispiel:
      Eine Gruppe von n Threads soll gleichzeitig gestartet werden
    - Lösung:
      - ➢ Nur wenn n CPUs frei sind, wird die Gruppe gestartet
      - ➢ Nach dem Start bleiben die einzelnen Threads mit ihrer CPU verbunden
        → Partitionierung der CPU

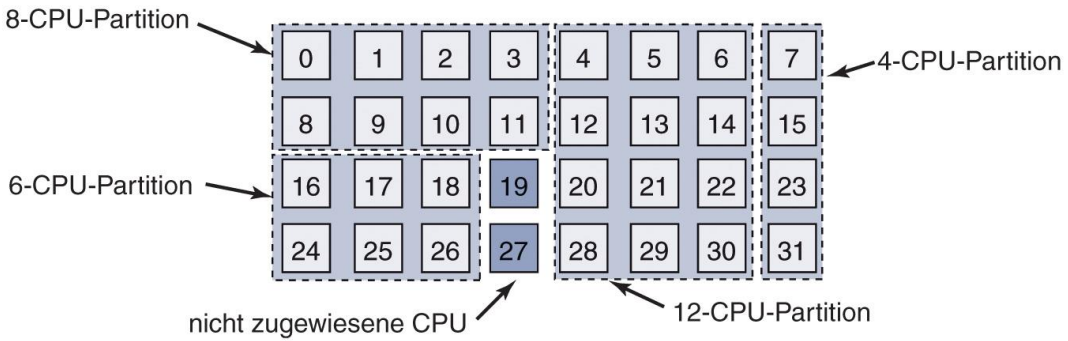


Abbildung 8.13: 32 CPUs, in vier Partitionen aufgeteilt, und zwei verfügbare CPUs

- Die Menge der CPUs (32) ist statisch in eine Anzahl von Partitionen aufgeteilt, von denen jeweils eine Partition mit 4, 6, 8 und 12 CPUs darstellt.
- Zwei der CPUs sind nicht zugewiesen.
- Im Laufe der Zeit werden sich Anzahl und Größe der Partitionen ändern, wenn neue Threads erzeugt und alte beendet werden.

Heute: Verteilte Systeme

- Verwandlung einer lose verbundenen Ansammlung von Computern in ein kohärentes System
- Vereinheitlichung des Systems
  - Beispiel: Unix
    - Alle Ein-/Ausgabegeräte sehen wie Dateien aus, d.h. der Umgang mit Tastaturen, Drucker und seriellen Schnittstellen ist viel einfacher als wenn sie alle konzeptionell unterschiedlich wären
- Einführung einer Softwareschicht oberhalb des Betriebssystems, um eine Einheitlichkeit der unterschiedlichen zugrunde liegenden Hardware und Betriebssysteme zu erreichen
  → Softwareschicht Middleware (eine Art Betriebssystem für verteilte Systeme)
- Die Schicht stellt bestimmte Datenstrukturen und Operationen zur Verfügung, die es Prozessen und Benutzern auf weit verstreuten Maschine erlaubt, konsistent zu interagieren
  - Entfernung (zeitlich, räumlich) der Prozessoren wird groß
  - Verwendung von Netzwerkdiensten und -protokollen

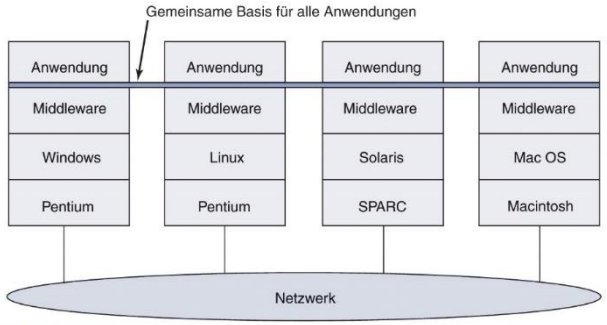


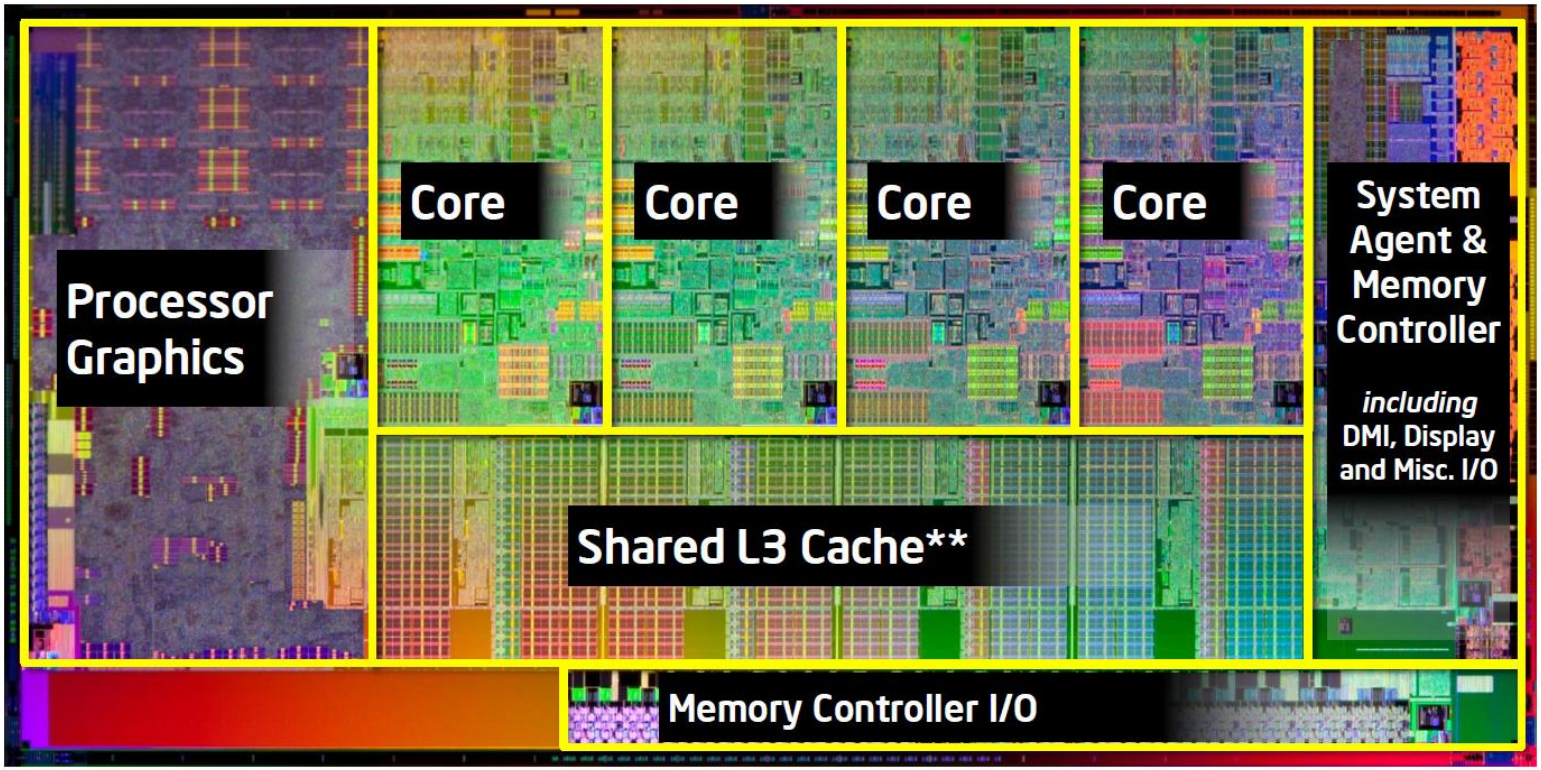
Abbildung 8.30: Position der Middleware in einem verteilten System

Anhang

Semiconductor & Computer Engineering:
https://en.wikichip.org/wiki/WikiChip

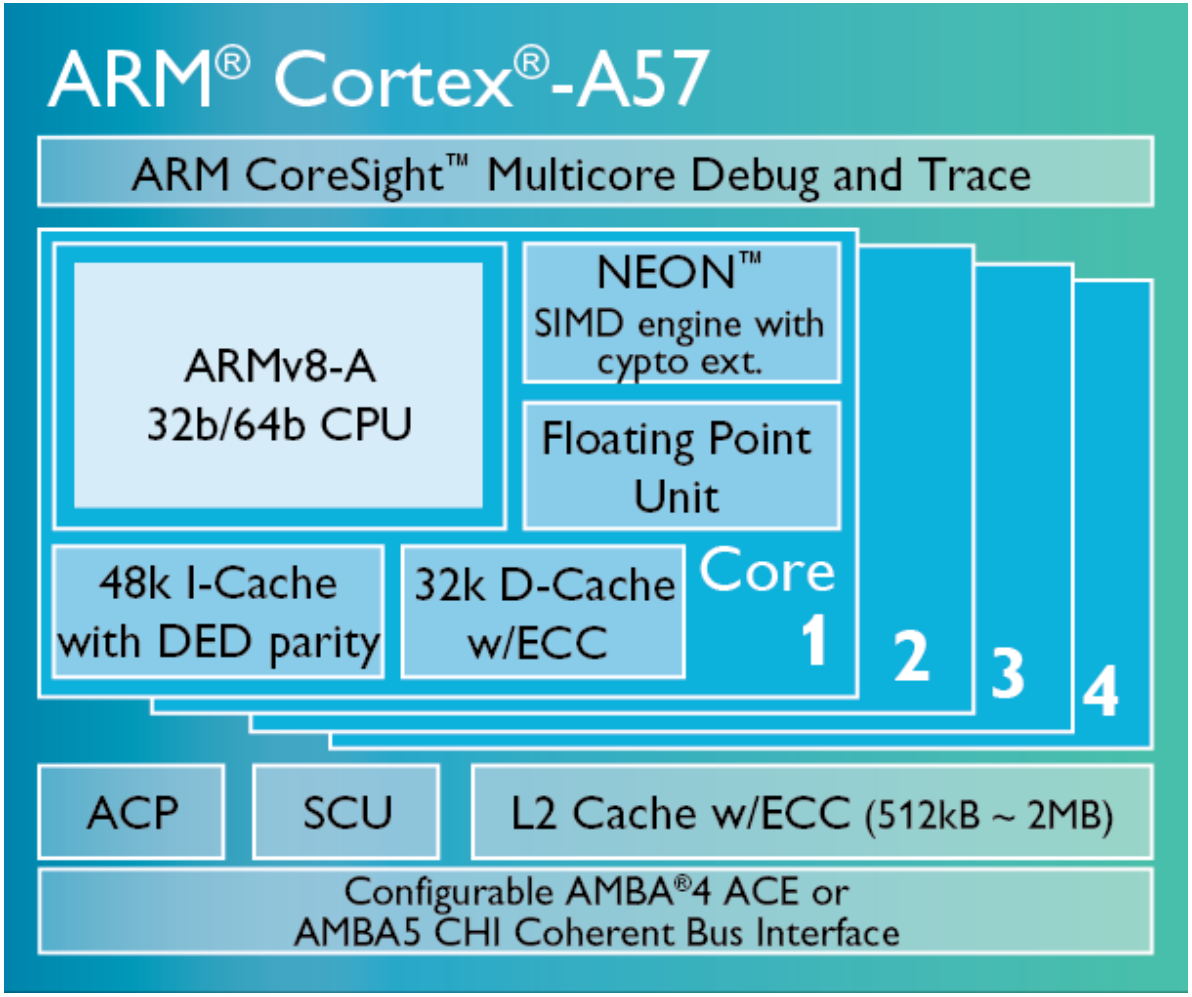
Operating Systems | News, how-tos, features, reviews, and videos:
https://www.computerworld.com/category/operating-systems/

DIE einer Sandy Bridge CPU (2011)



Ein Die ist in der Halbleiter- und Mikrosystemtechnik die Bezeichnung eines einzelnen, ungehäusten Stücks eines Halbleiter-Wafers. Ein solches Die wird üblicherweise durch Sägen oder Brechen des fertig bearbeiteten Wafers in rechteckige Teile gewonnen. In der Regel befindet sich auf einem Die ein Bauteil,

Mehrkernprozessoren ARM Cortex-A57



Diskussion

Tschüß

Frohe Weihnachten

Viel Erfolg bei den anstehenden Klausuren.

Bleiben Sie gesund und zuversichtlich.

Wir sehen uns im Frühling 2023 wieder.

Bitte melden Sie sich wegen
des Refresh-Termins.