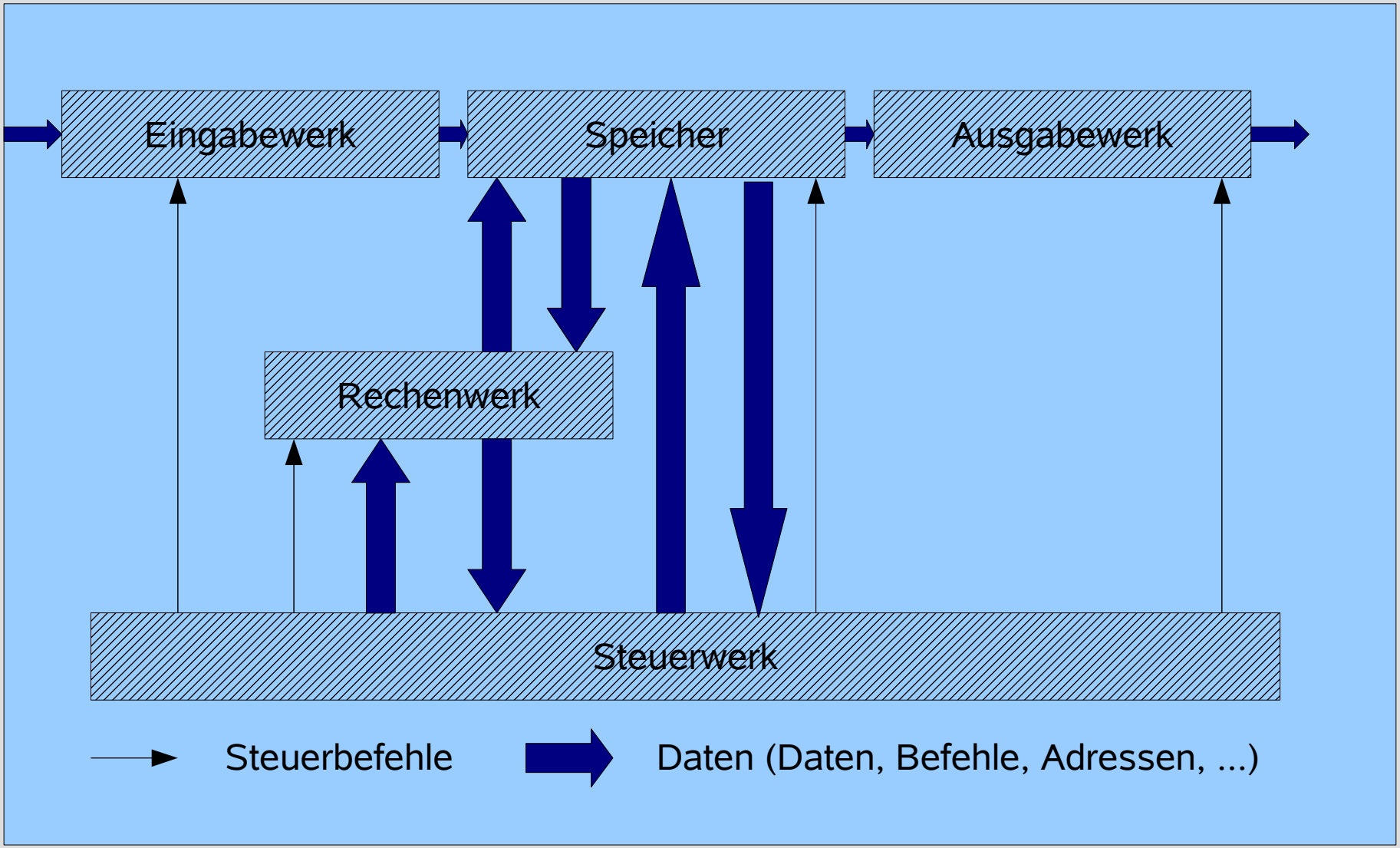
Parallele Rechnerarchitekturen

- Bisher behandelte von Neumann-Architektur:

v.-Neumann-Architektur



Parallele Rechnerarchitekturen

- Bisher behandelte von Neumann-Architektur
  - entscheidender Nachteil:
  - zu einem Zeitpunkt kann nur ein Maschinenbefehl geholt und verarbeitet werden.
  - diese Sequentialität ist aber nicht zwingend.

Parallele Rechnerarchitekturen

- Bisher behandelte von Neumann-Architektur
  - entscheidender Nachteil:
    - zu einem Zeitpunkt kann nur ein Maschinenbefehl geholt und verarbeitet werden.
    - diese Sequentialität ist aber nicht zwingend.
  - Ausweg:
    - statt *sequentieller* Ausführung **parallele** Ausführung.

Parallele Rechnerarchitekturen

- Bisher behandelte von Neumann-Architektur
  - entscheidender Nachteil:
    - zu einem Zeitpunkt kann nur ein Maschinenbefehl geholt und verarbeitet werden.
    - diese Sequentialität ist aber nicht zwingend.
  - Ausweg:
    - statt *sequentieller* Ausführung **parallele** Ausführung.
  - 2 Ausprägungen der Parallelität:
    - zeitliche Parallelität
    - räumliche Parallelität

Parallele Rechnerarchitekturen

- Ziele:
  - Hohe Leistung
    Leistung > schnellster Uniprozessor
  - konkurrierendes Preis-Leistungsverhältnis
    konkurrierend mit Workstations
  - Skalierbare Leistung und Kosteneffektivität für kleine und große Prozessorzahlen
  - Allgemein kosteneffektiv für einen großen Bereich von Anwendungen.

Räumliche Parallelität

- Idee:
  - Leistungsfähige Computer durch das Zusammenschalten mehrerer kleinerer Computer zu realisieren

  - Multiprozessoren:
    - Parallele Prozessoren mit einem einzigen, gemeinsam genutzten Adressraum
  - Cluster

Räumliche Parallelität (2)

- Fragestellungen:
  - Wie nutzen parallele Prozessoren gemeinsame Daten?
  - Wie werden parallele Prozessoren koordiniert?
  - Wieviele Prozessoren sollen verwendet werden?

  - Unterschiedliche Sichtweisen bedingen unterschiedliche Architekturen

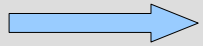
Räumliche Parallelität (3)

- Einschub: Nutzen des parallelen Aufwandes
  - Amdahls Gesetz

  Ausführungszeit nach Verbesserung =

  (von Verbesserung betroffene Ausführungszeit / Verbesserungsfaktor)
  + von der Verbesserung nicht betroffene Ausführungszeit

Räumliche Parallelität (4)

- Einschub: Nutzen des parallelen Aufwandes
  - Amdahls Gesetz
  - Ziel lineare Verbesserung um Faktor 100 durch den Einsatz von 100 Prozessoren 

  Ausführungszeit vor der Verbesserung / 100 = (Von der Verbesserung beeinflusste Ausführungszeit / 100) + nicht beeinflusste Ausführungszeit

Räumliche Parallelität (5)

- Einschub: Nutzen des parallelen Aufwandes
  - weil gilt:
  - **Ausführungszeit vor der Verbesserung = Von der Verbesserung beeinflusste Ausführungszeit + Nicht beeinflusste Ausführungszeit**
  - und dies in die obige Gleichung eingesetzt:

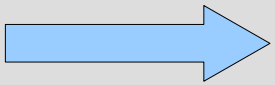
Räumliche Parallelität (6)

- Einschub: Nutzen des parallelen Aufwandes
  (von der Verbesserung beeinflusste
  Ausführungszeit + nicht beeinflusste
  Ausführungszeit) / 100
  =
  Von der Verbesserung beeinflusste
  Ausführungszeit / 100 +
  Nicht beeinflusste Ausführungszeit

  oder aber

Räumliche Parallelität (7)

- Einschub: Nutzen des parallelen Aufwandes
  Nicht beeinflusste Ausführungszeit / 100
  =
  nicht beeinflusste Ausführungszeit

  gilt nur wenn nicht beeinflusste Ausführungszeit (linearer Anteil) gleich Null

Räumliche Parallelität (7)

- S - Speedup
- a - sequentieller Anteil
- P - Prozessorzahl

$$S = \frac{1}{a + \frac{(1-a)}{p}}$$

Räumliche Parallelität (8)

- Frage:
  es sollen 2 Additionen ausgeführt werden:
  1. Addition zweier Variablen
  2. Addition zweier Matrizen 1000 * 1000

  Welche Beschleunigung erhalten Sie mit 1000 Prozessoren?

Räumliche Parallelität (9)

- Weitere Ausprägungen des Amdahlschen Gesetzes:
- Berücksichtigung der Kommunikation:

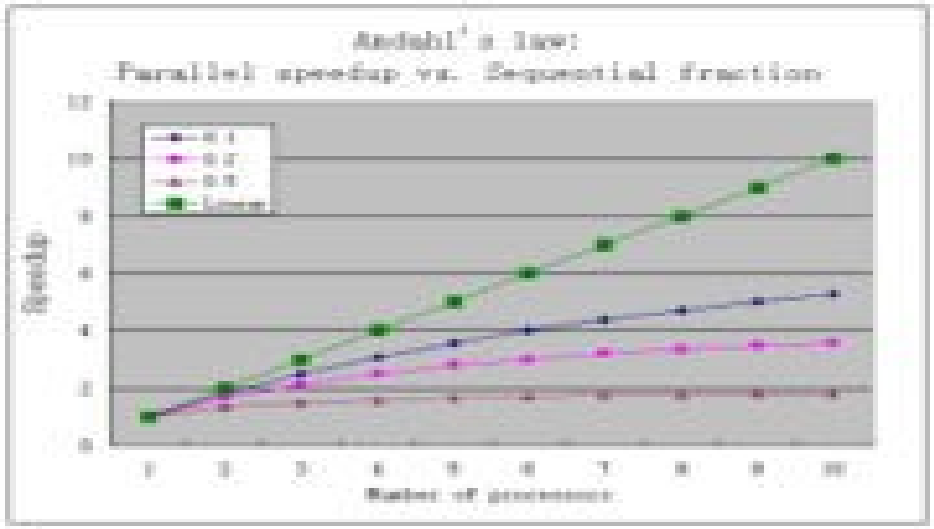
$$S = \frac{1}{a + o(P) + \frac{1-a}{P}} \leq \frac{1}{a}$$

mit a - linerearer Anteil
P - Anzahl der Prozessoren
o(P) – Anteil der Kommunikation
s - Speedup

Räumliche Parallelität (9)

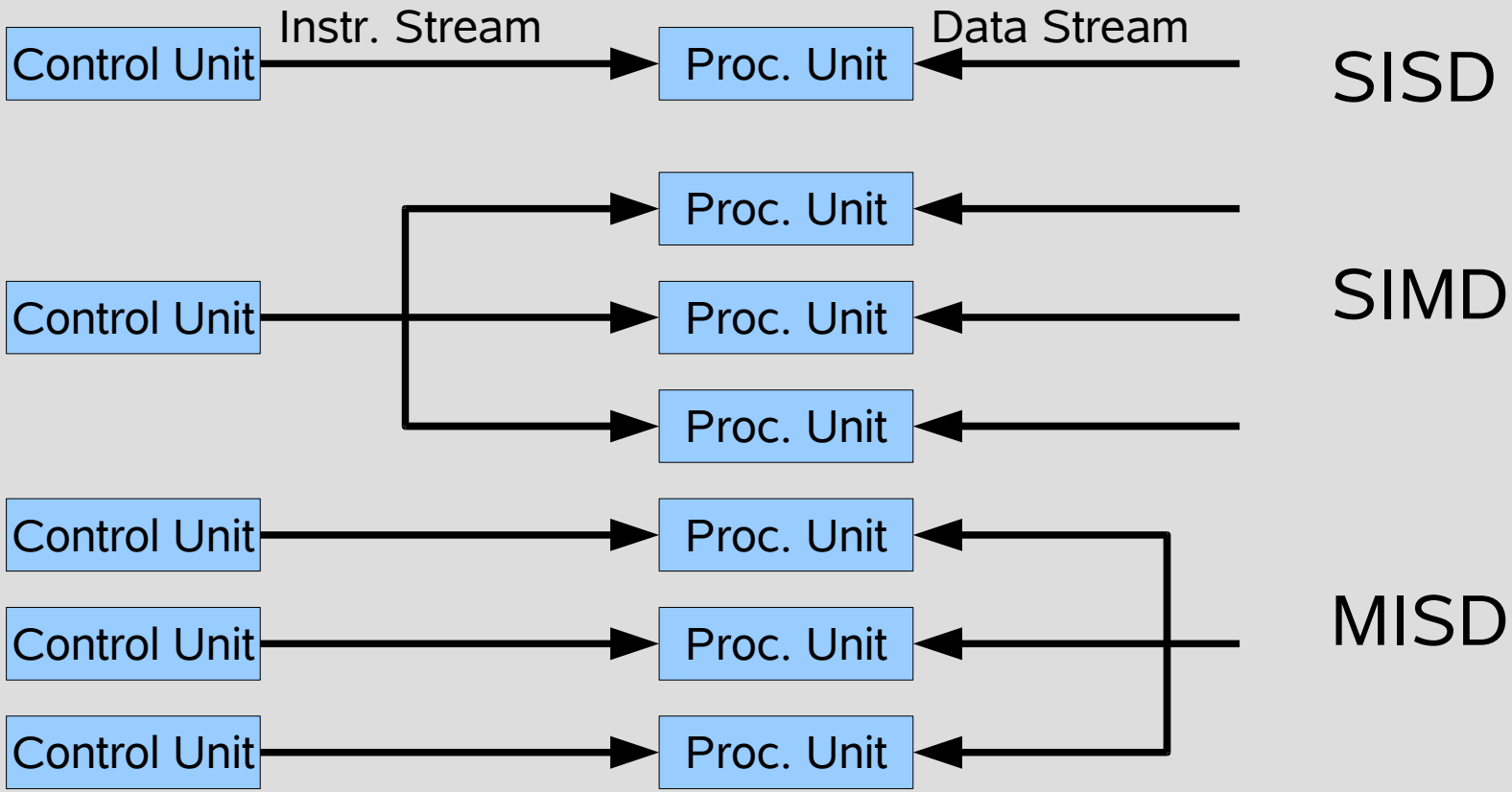
- Weitere Ausprägungen des Amdahlschen Gesetzes:
- Berücksichtigung der Kommunikation:

$$S = \frac{1}{a + o(P) + \frac{1-a}{P}} \leq \frac{1}{a}$$



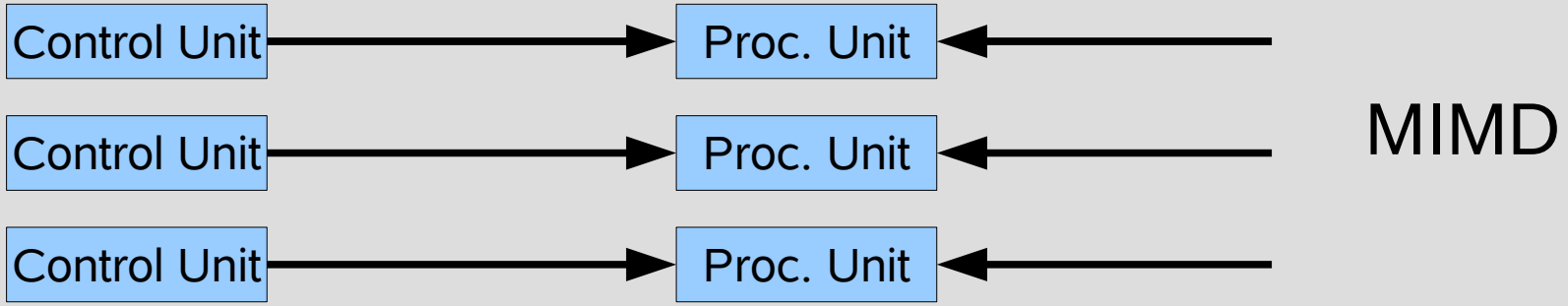
Parallele Rechnerarchitekturen

- Flynn's Klassifikation:



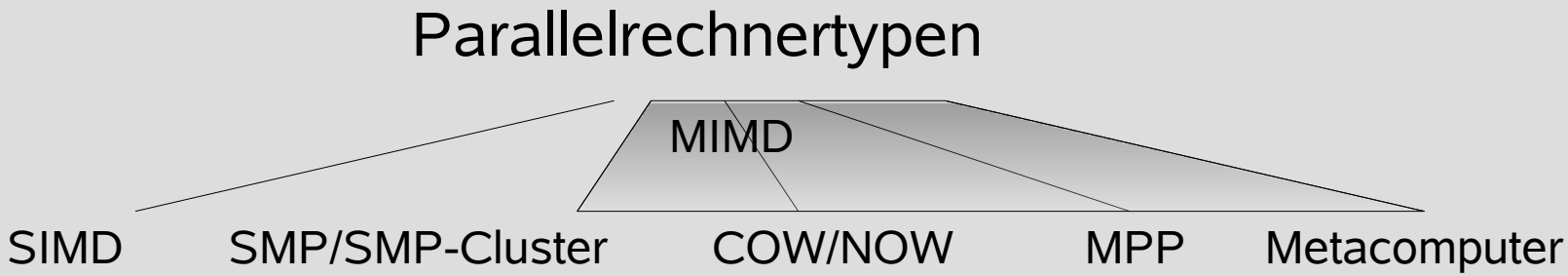
Parallele Rechnerarchitekturen

- Flynn's Klassifikation:



Parallele Rechnerarchitekturen

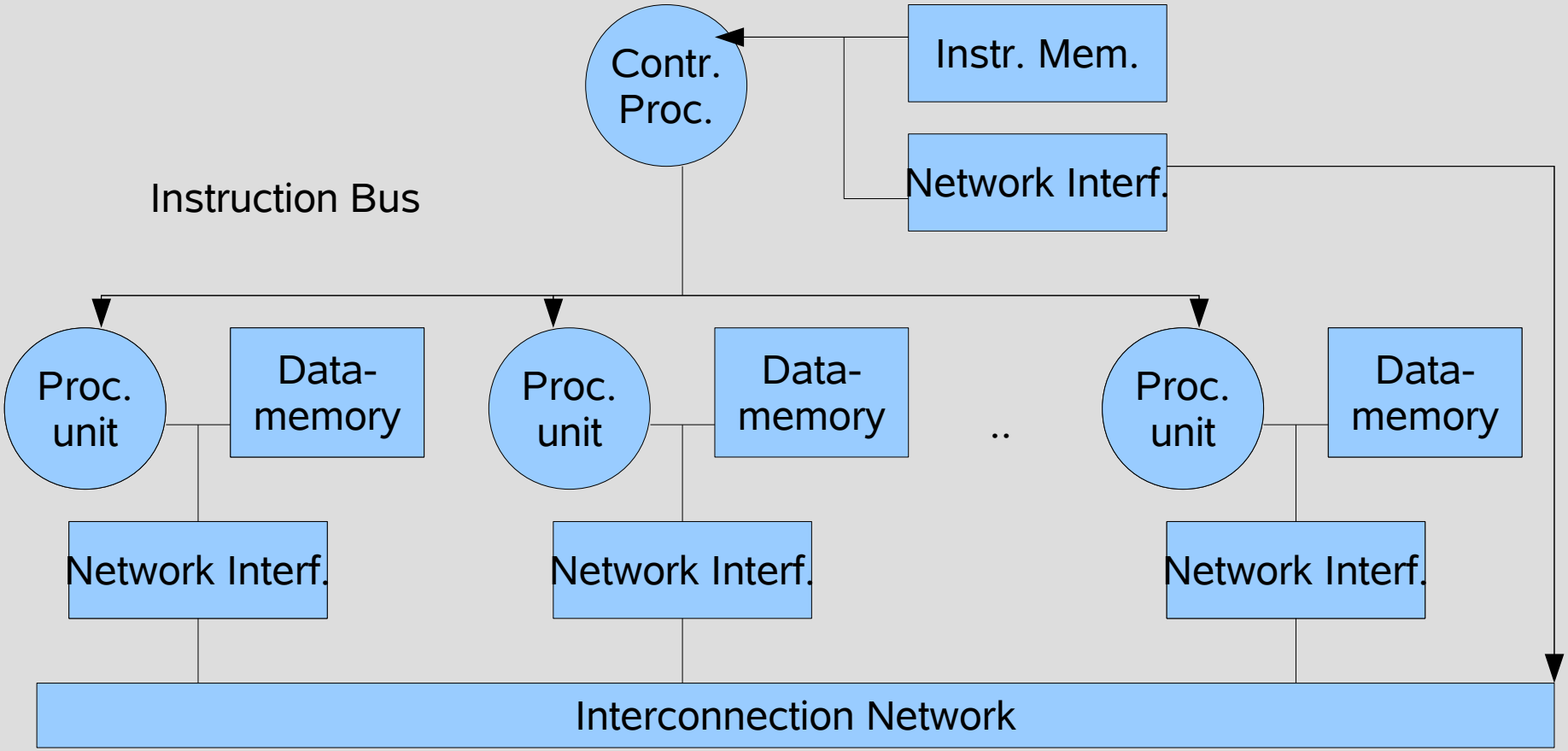
- Pragmatische Einteilung:



SIMD	Single Instruction Multiple Data – Computer (Vektorrechner)
SMP	Symmetric Multiprocessor-System
COW	Cluster of Workstations
NOW	Network of Workstations
MPP	Massive Parallel Processor System
Metacomputer	Cluster vernetzter heterogener Parallelrechnersysteme

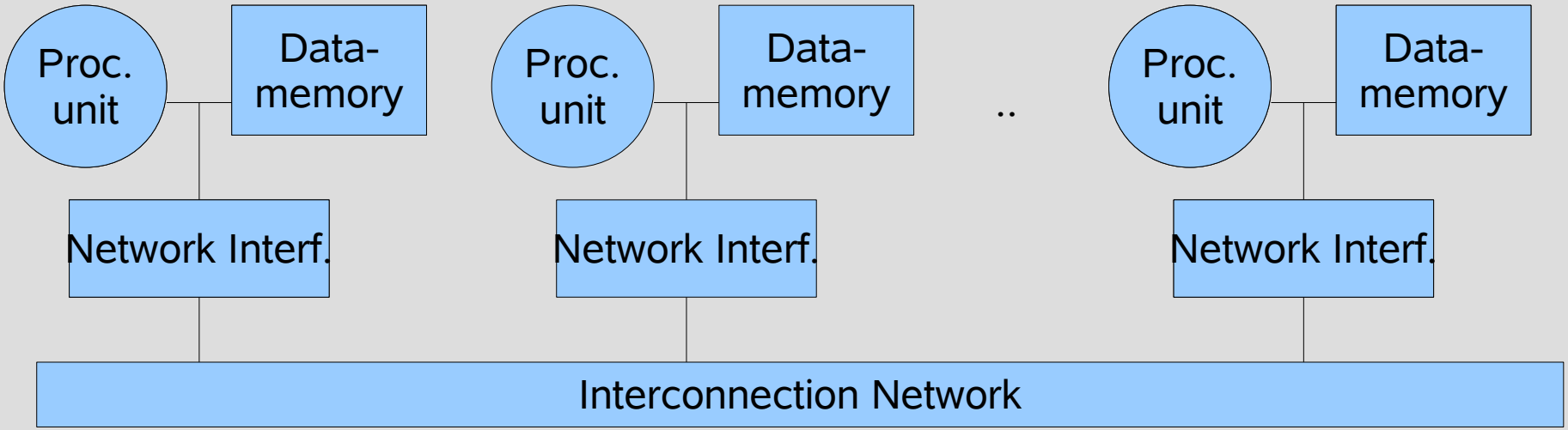
Parallele Rechnerarchitekturen

- SIMD Multiprozessorarchitektur



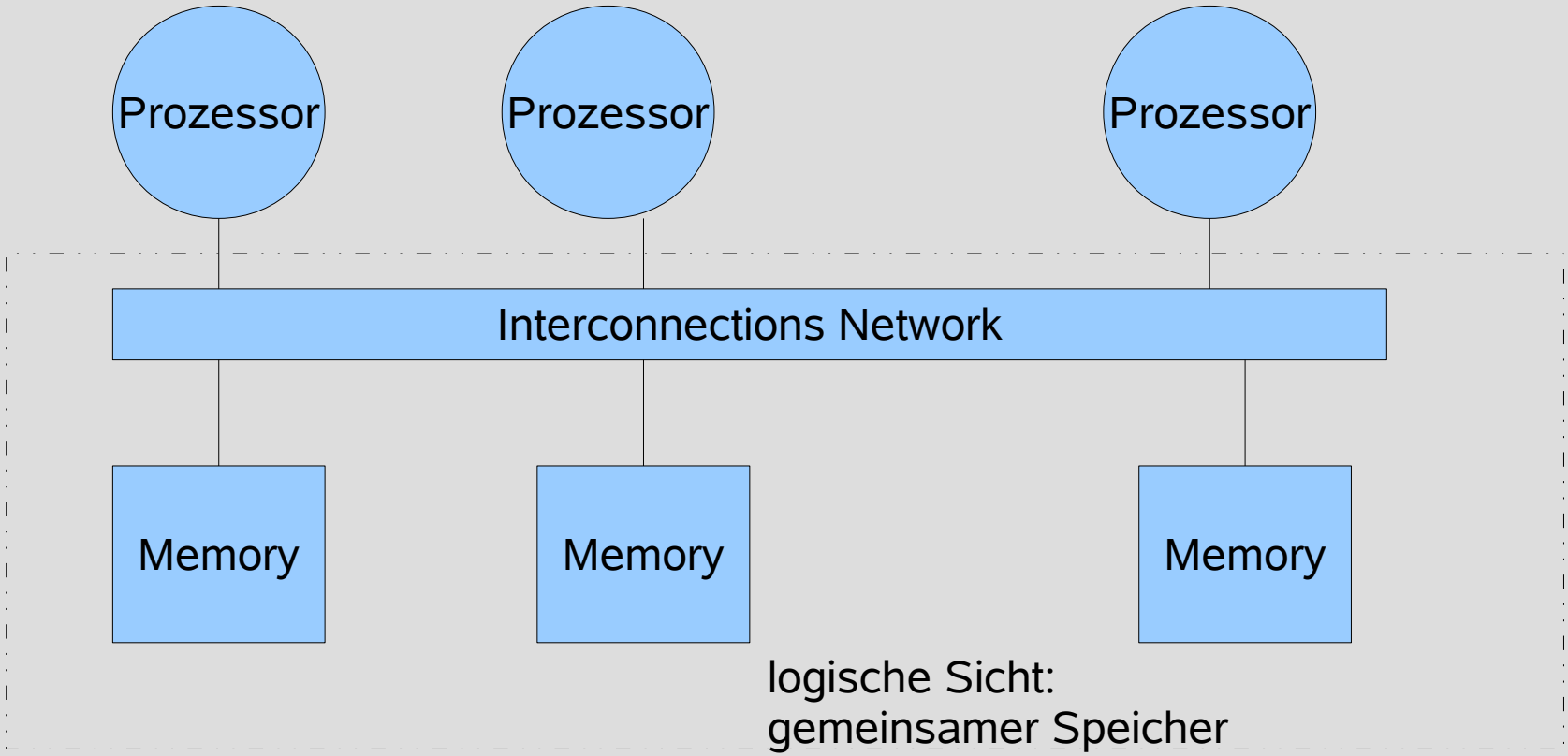
Parallele Rechnerarchitekturen

- MIMD Multiprozessorarchitektur



Parallele Rechnerarchitekturen

- Shared Memory Multiprozessorarchitektur

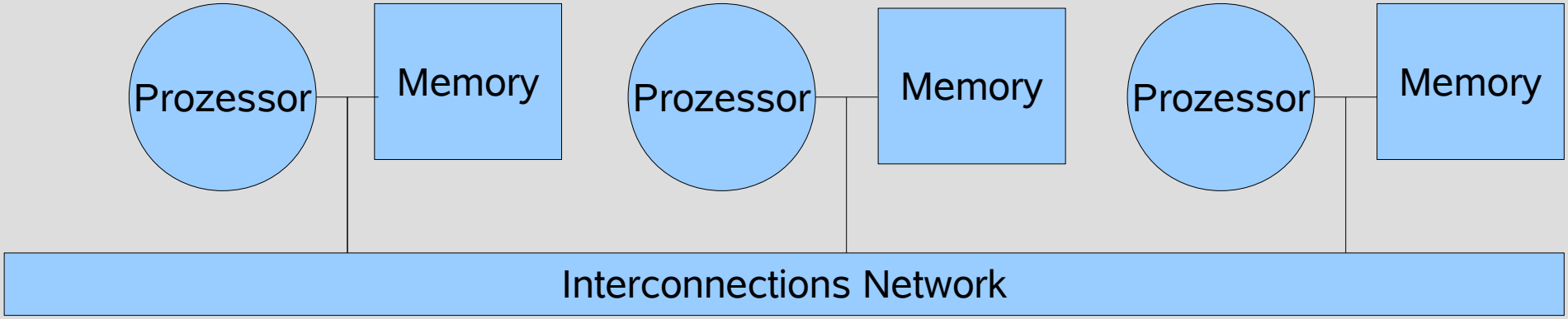


Parallele Rechnerarchitekturen

- Shared Memory Multiprozessorarchitektur
  - globaler Adressraum
  - Vorteil:
    - einfaches Programmiermodell (wie bei Uniprozessorsystem), wonach ein Prozessor auf den gesamten Adressraum zugreifen kann, wird erhalten
  - Nachteil:
    - nur bei kleinen Prozessorenzahlen leicht beherrschbar
    - skaliert nicht

Parallele Rechnerarchitekturen

- Distributed Memory Multiprozessorarchitektur



Parallele Rechnerarchitekturen

- Distributed Memory Multiprozessorarchitektur
  - nur lokale Adressräume, Kommunikation erfolgt durch *Message Passing*, Prozessoren arbeiten asynchron
  - Vorteil:
    - einfach herstellbar
    - skalierbar
  - Nachteil
    - Message-Passing muss explizit formuliert werden, was entsprechende Programmiermodell erfordert
    - Technik des „Distributed-Shared Memory“ umgeht diesen Nachteil

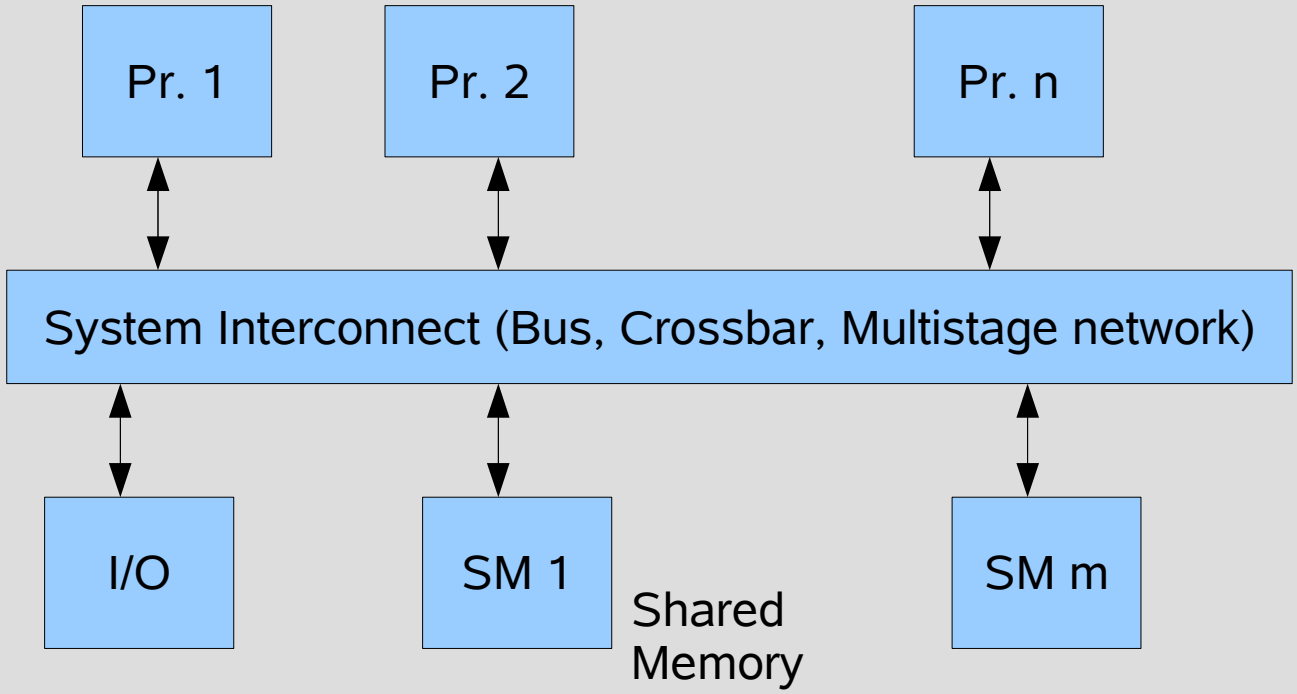
Parallele Rechnerarchitekturen

- Beispiele

	SIMD	MIMD
Message-Passing	Illiac IV TMC CM-1 MasPar MP-1 MasPar MP-2	Intel iPSC, Paragon nCube2, TMC CM-5 Parsytec GC Meiko CS-2 IBM SP1, SP2, TUC CLIC, CHIC
Shared Memory	TMC CM-2	IBM RP3, IBM ES/9000 Cray X-MP, Y-MP, C90 Sequent Symmetry BBN Butterfly KSR-1/8 (TUCh)

Parallele Rechnerarchitekturen

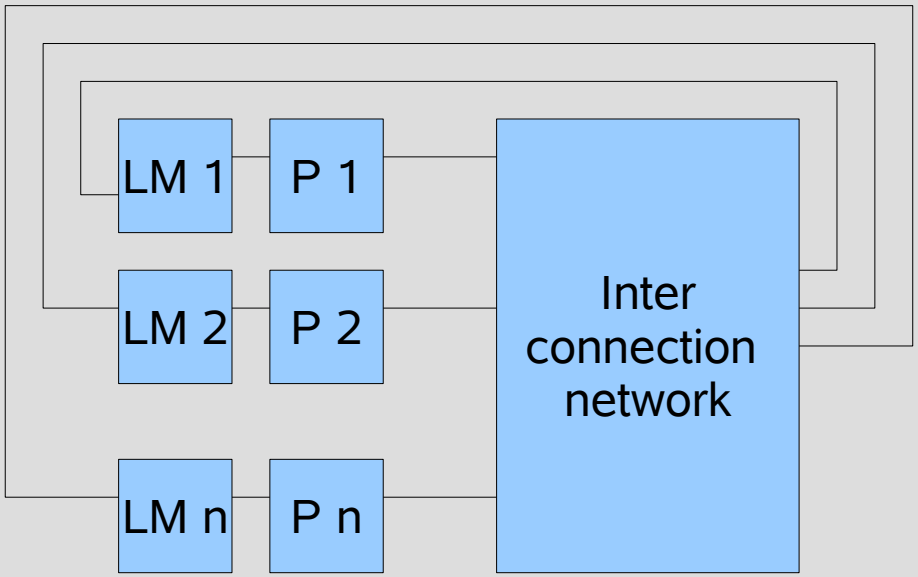
- Uniform Memory Access (UMA)
  Zugriffsverzögerung ist zu jedem Teil des Speichers gleich (uniform)



Beispiel:
Sequent Symmetry

Parallele Rechnerarchitekturen

- Non Uniform Memory Access (NUMA)
  Zugriffsverzögerung differiert zwischen den verschiedenen Speicherkomplexen; lokale Zugriffe erfolgen schneller als entfernte

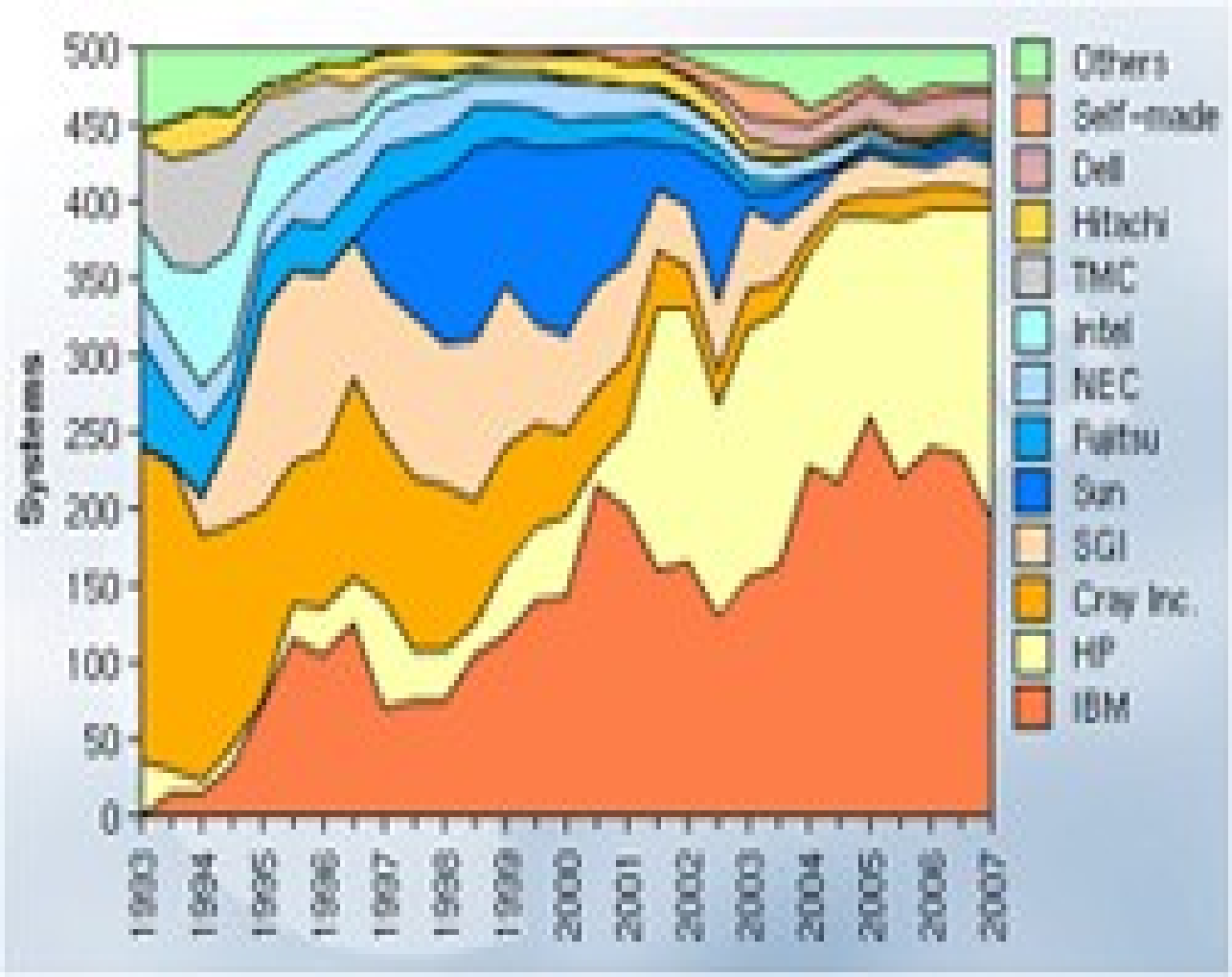


Shared local Memory (z.B. BBN Butterfly)

Top 500

- Supercomputer die TOP 500 Liste
- siehe www.top500.org
- Bedeutende Hersteller:
  - IBM (47 % der Rechner), HP, SGI, Cray, SUN
- Gesamtleistung: 5213548,18 GF (Benchmark PM_{peak})
- Spitzenreiter November 2006
  - Blue Gene LawrenceLivermore National Laboratory (IBM)
  - Cray Red Storm Sandia National Laboratory (Cray)
  - Blue Gene Solution IBM Thomas Watson Research Center

Top 500


Systems
500
450
400
350
300
250
200
150
100
50
0
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
Others
Self-made
Dell
Hitachi
TMC
Intel
NEC
Fujitsu
Sun
SGI
Cray Inc.
HP
IBM

Top 500

117 Technical University of Chemnitz, CHIC

Germany

xSeries x3455 Cluster Opteron, 2.6 GHz, Infiniband IBM

Stand Juni 2007, in der aktuellen Liste nicht mehr aufgeführt ;-(

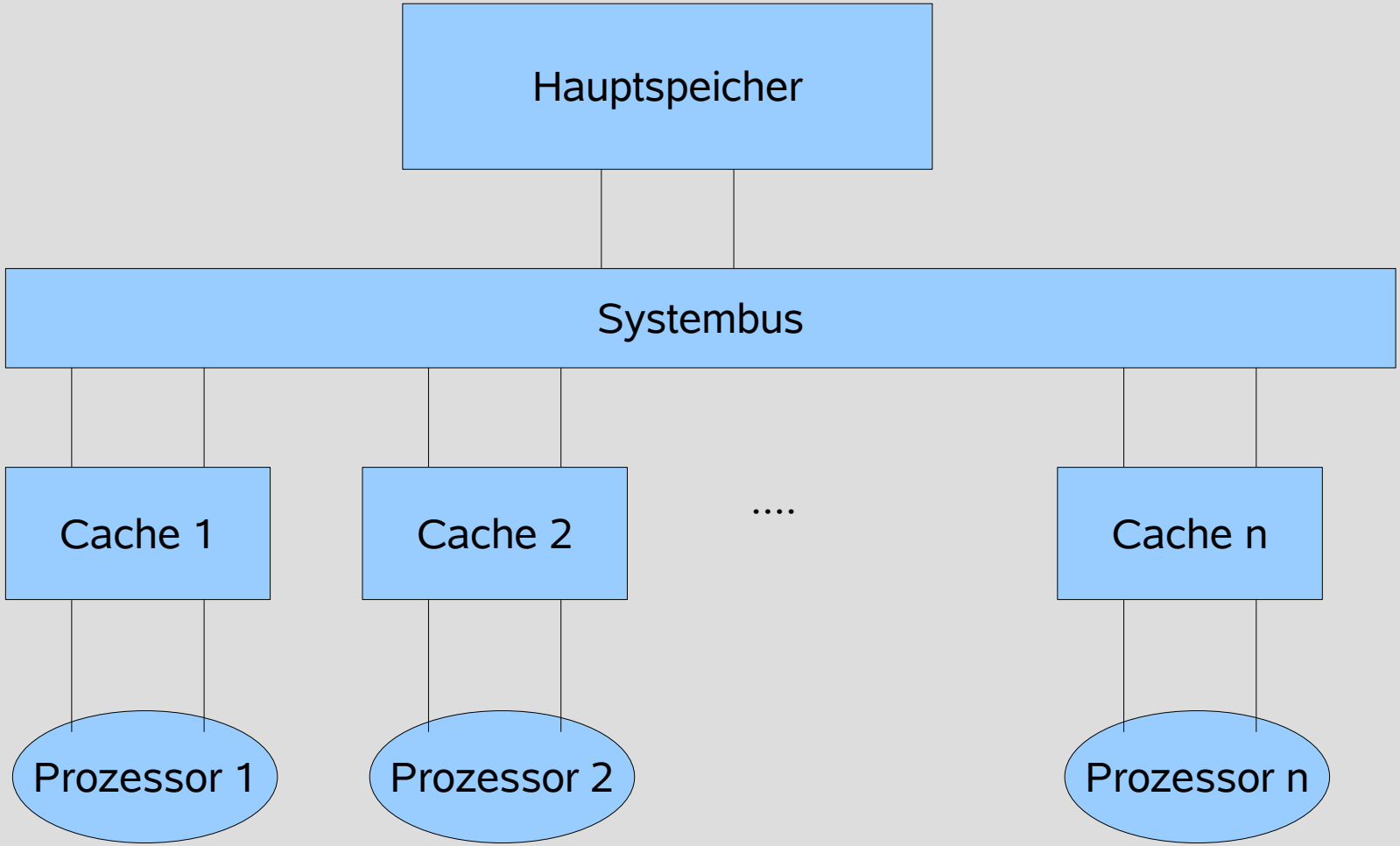
SMP

- kann als natürliche Erweiterung eines Uniprozessorsystems angesehen werden
- jeder Prozessor hat gleichen Zugriff zum Hauptspeicher (UMA-Typ), der hier als *shared memory* genutzt wird

SMP (2)

Beispiel Multiprozessorsystem :
Dualprozessor-Motherboard

SMP (3)


Hauptspeicher
Systembus
Cache 1
Cache 2
....
Cache n
Prozessor 1
Prozessor 2
Prozessor n

SMP (4)

- Der Bus kann zu einem Zeitpunkt jeweils nur von einem Prozessor genutzt werden, d.h. bei ***N*** Prozessoren sinkt die im Mittel von einem Prozessor nutzbare Speicherbandbreite auf ***1/N***.
- Solange ein Prozessor im Mittel ca ***N*** Zugriffe im Cache und erst dann einen Hauptspeicherzugriff tätigt, wirkt obiger Fakt noch nicht leistungsbegrenzend

SMP (5)

- Da einerseits die Busbandbreite und andererseits die Cache-Lokalität der Programme begrenzt ist, „skalieren“ MP's dieser Architektur effizient nur bis etwa 30 Prozessoren.

Asymmetrisches Multiprocessing

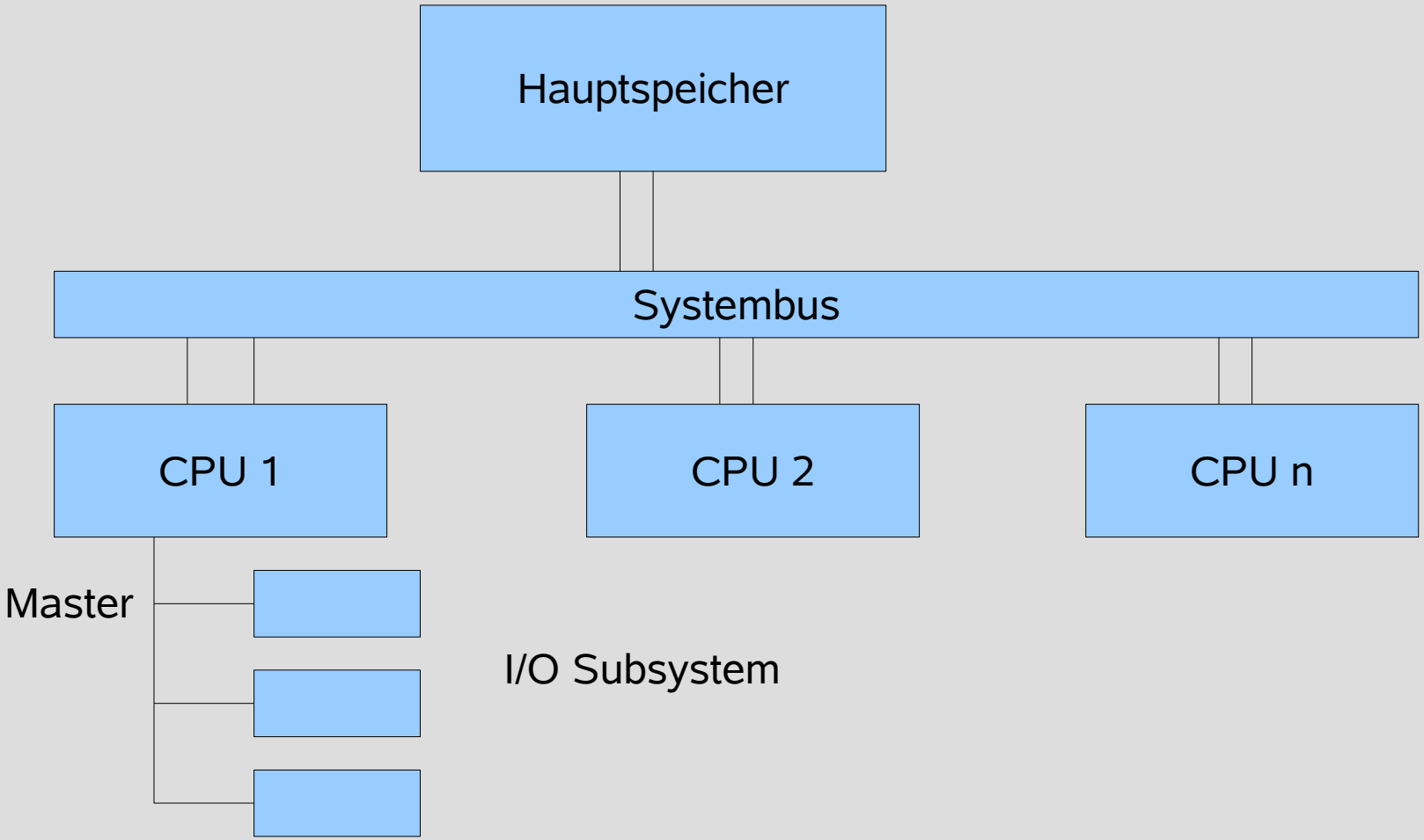
- Ein Prozessor ist als Master vorgesehen, die anderen arbeiten als Slaves.
- Die Beschränkung der I/O-Kommunikation auf den Master liegt typisch darin begründet, dass nur er über I/O-Anschlüsse inklusive der damit verbundenen Interruptleitungen verfügt.
- der Master führt alle priviligierten Operationen wie I/O und das Management des Betriebssystems durch.

Asymmetrisches Multiprocessing (2)

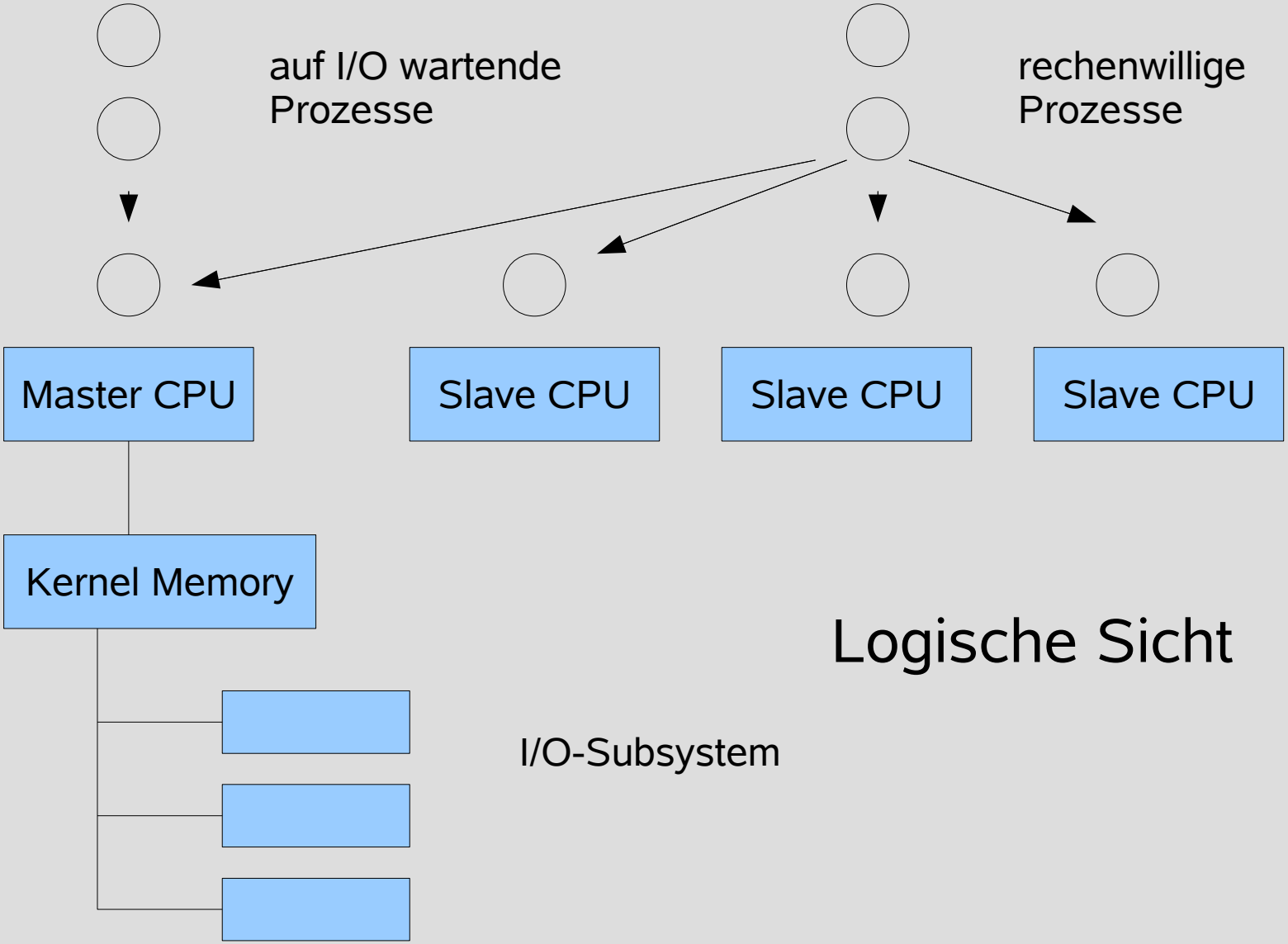
- Nur der Master kann Betriebssystemkode ausführen. Die Slaves arbeiten Anwenderkode ab und rufen den Master zur Ausführung von Systemdiensten

Asymmetrisches Multiprocessing (3)

- Physikalische Sicht



Asymmetrisches Multiprocessing (4)



Logische Sicht

Asymmetrisches Multiprocessing (5)

- Der Durchsatz von ASMP sinkt, wenn viele Betriebssystemdienste (insbesondere I/O's, I/O-Flaschenhals) angefordert werden, was mit steigender Prozessorzahl verstärkt der Fall ist
- ASMP skalieren schlecht, sind jedoch einfach zu implementieren.

Symmetrisches Multiprocessing

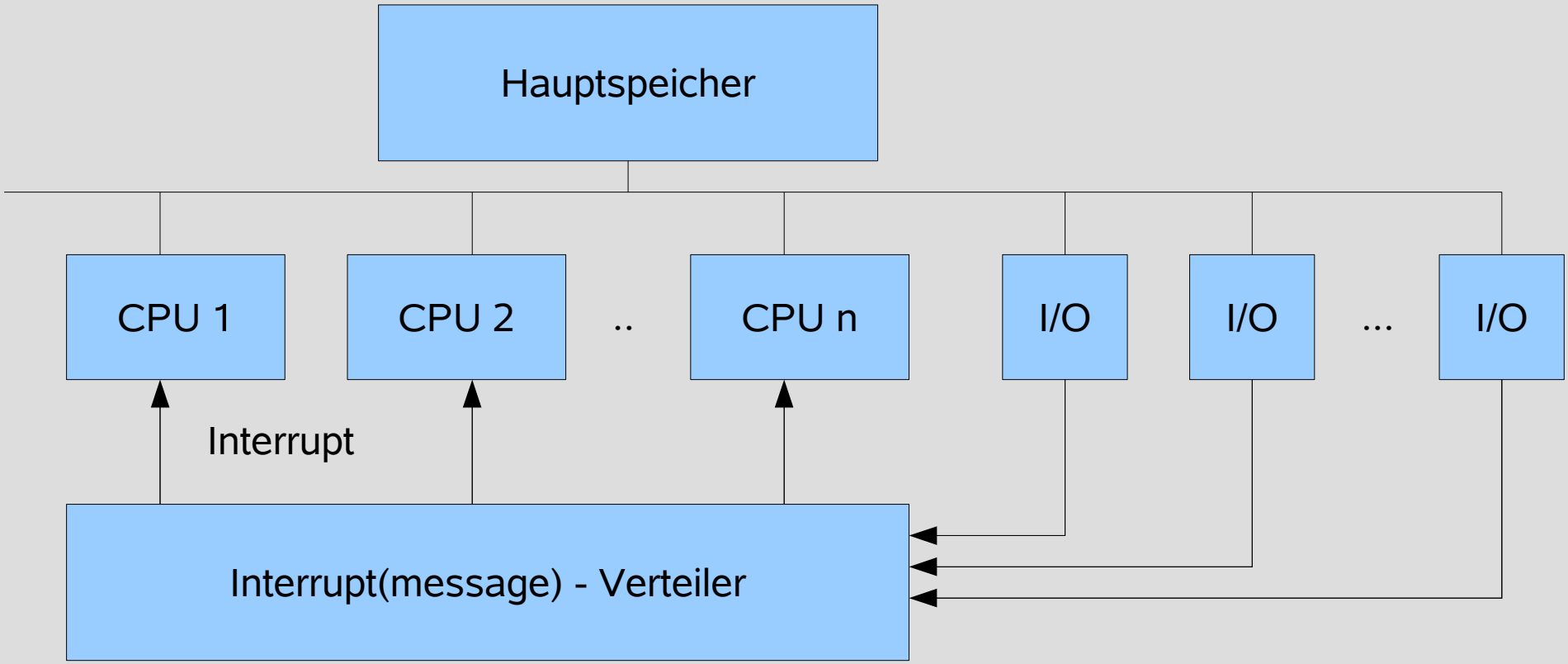
- Alle Prozessoren sind insofern gleich, als das jeder sowohl Betriebssystem als auch Anwenderkode abarbeiten kann.
- Alle Prozessoren haben Zugriff auf das Kernelimage im gemeinsamen Hauptspeicher und können im Kern parallel arbeiten. (Das Betriebssystem muss entsprechend konfiguriert sein (z.B. Linux – SMP - Kernel))

Symmetrisches Multiprocessing (2)

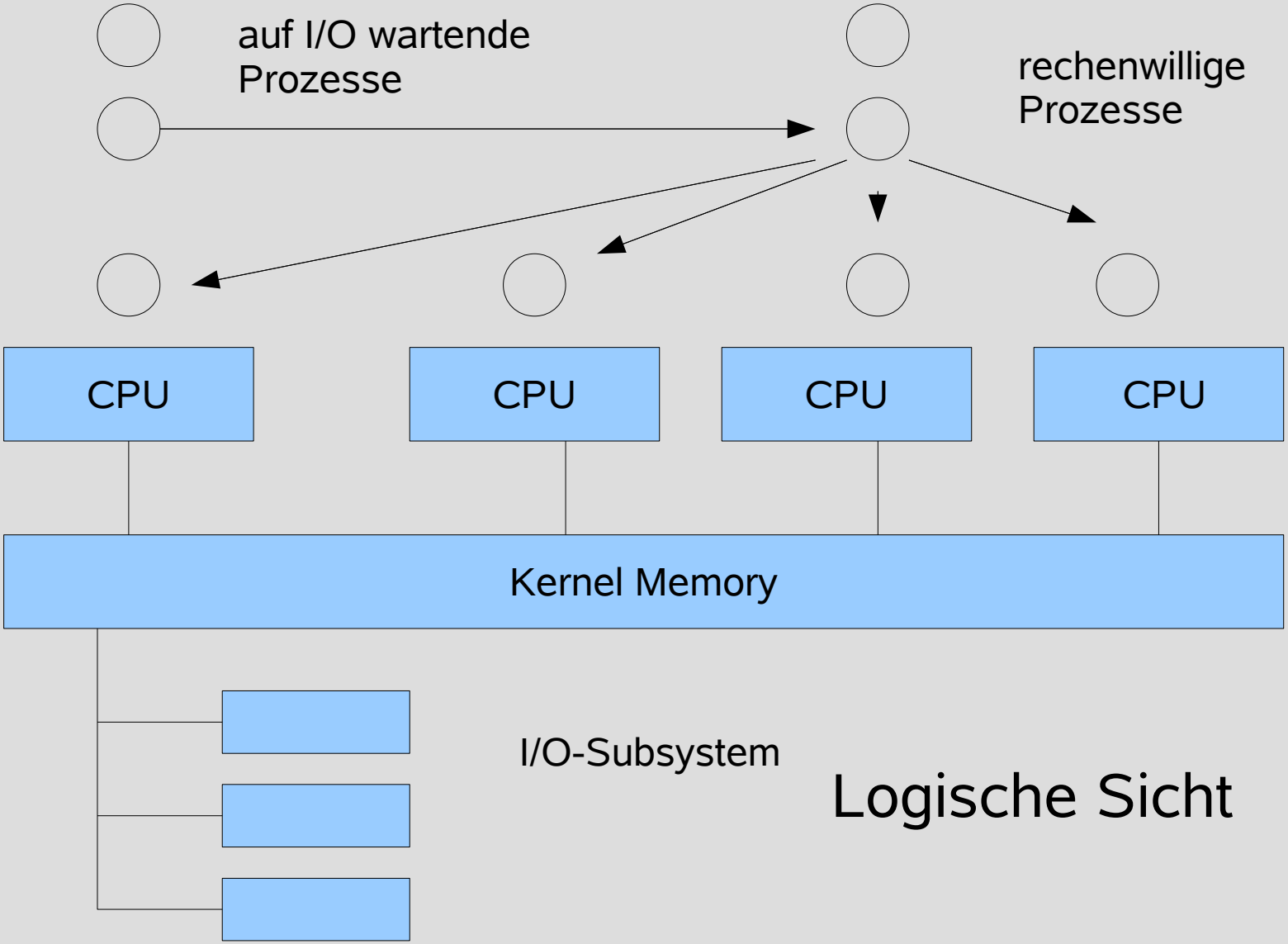
- Insbesondere können I/O-Anforderungen und Gerätetreiber-Interrupts parallel von verschiedenen Prozessoren bearbeitet werden.
- Jedes Hauptspeicher datum kann von jeder CPU wie auch von jedem I/O-Gerät referenziert werden.
- Die asynchrone Kommuniktion zwischen den CPU's kann interrupt-gesteuert erfolgen.

Symmetrisches Multiprocessing (3)

- Physikalische Sicht:



Symmetrisches Multiprocessing (4)



Logische Sicht

Symmetrisches Multiprocessing (5)

- SMP's skalieren besser als ASMP's. Nahezu alle gegenwärtig auf dem Markt befindlichen Multipozesorsysteme sind SMP's.

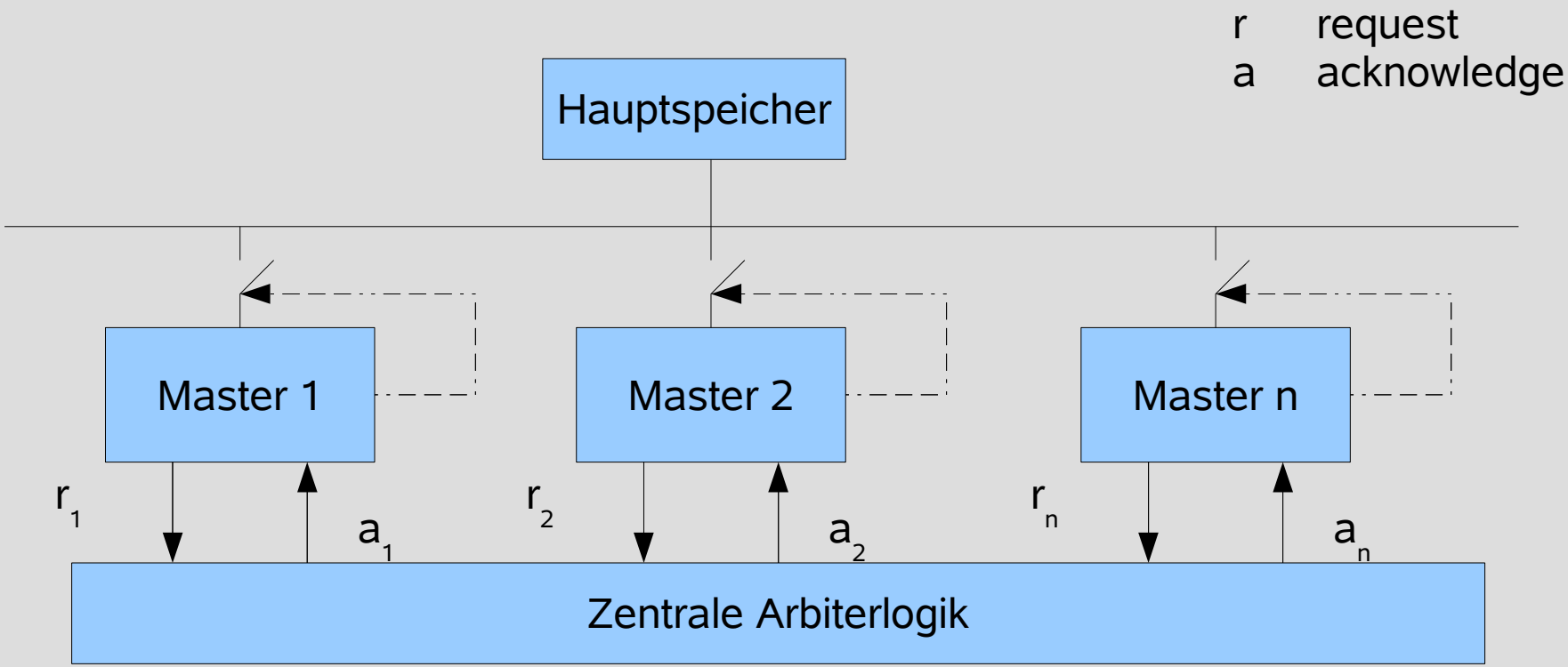
Busarbitrierung

- Vergabe der Buskontrolle
  - Zentral ⟺ Dezentral
  - Kontrolliert ⟺ Zufällig

  - Busse mit **zentraler Busvergabe** besitzen spezielle Arbitirerungseinheit.
  - Busse mit **dezentraler Busvergabe** erfordern eine Absprache der Busteilnehmer, wer den Bus benutzen darf.

Busarbitrierung (2)

- Zentrales Arbitrierungsprinzip:



Busarbitrierung (3)

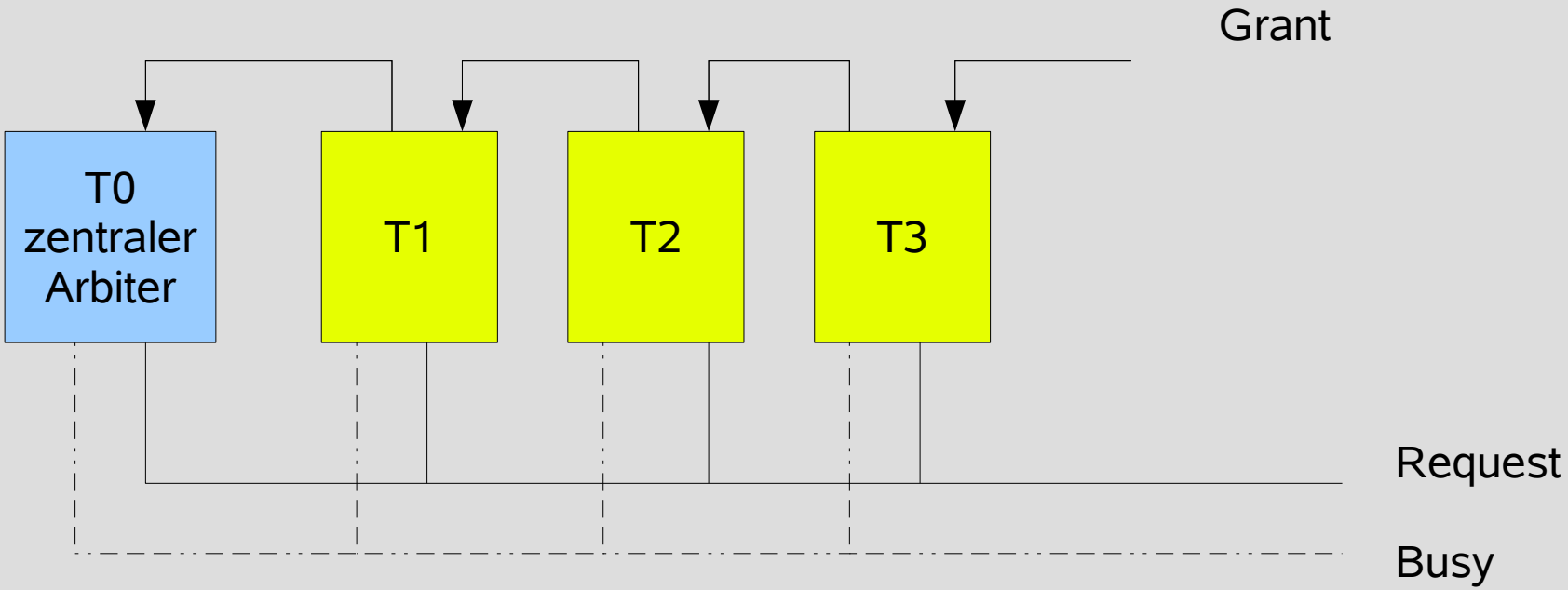
- Bei vielen Bussen wird die Vergabe nach festgelegten Regeln kontrolliert abgewickelt
- Es gibt aber auch Busse, die die Vergabe mehr oder weniger zufallsbasiert ausführen.
- Welches Verfahren angewendet wird, hängt davon ab, ob es sich um einen parallelen oder einen seriellen Bus handelt

Busarbitrierung (4)

- Parallele Busse (nur kontrollierte Arbitrierung)
  - Daisy Chaining
  - Polling
  - Stichleitungen
- Serielle Busse
  - Kontrolliert (Token Passing, CSMA/CA)
  - Zufällig (Aloha, CSMA/CD)

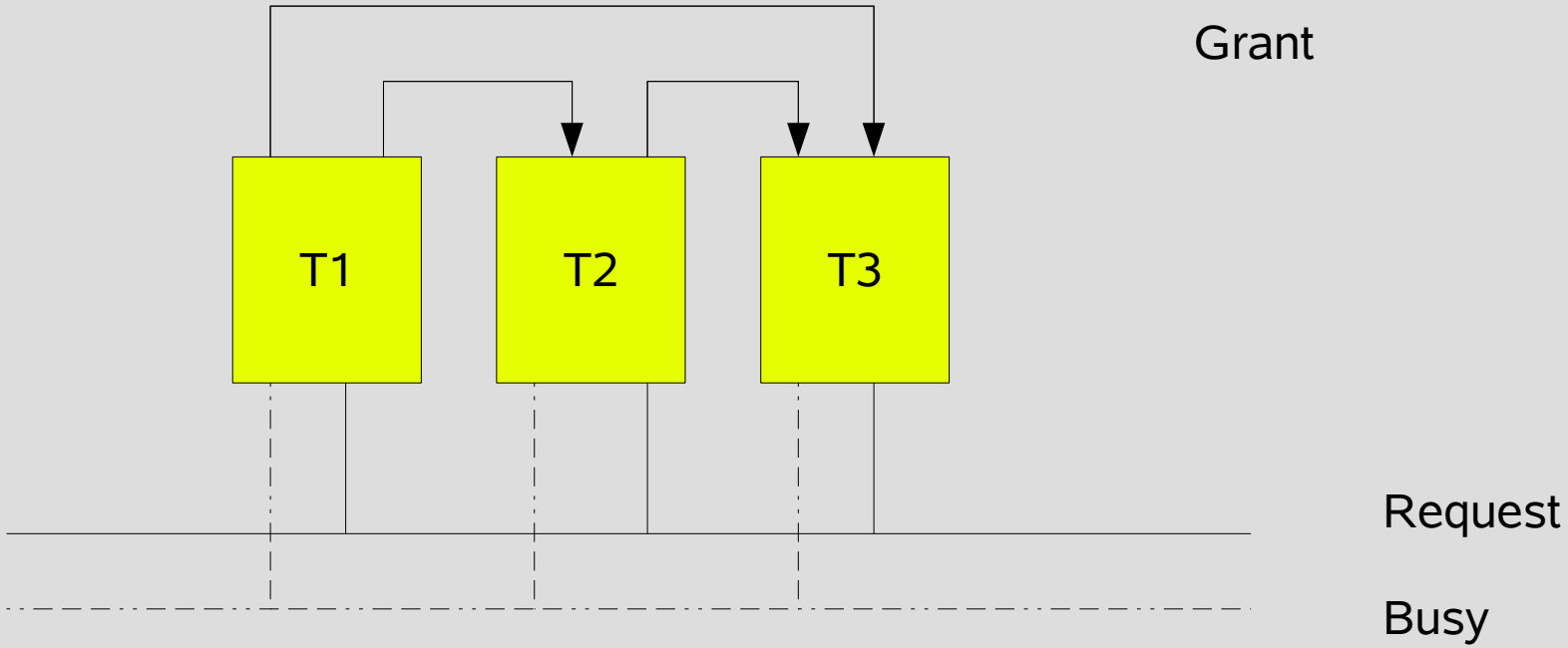
Busarbitrierung (5)

- Daisy Chaining (zentral)



Busarbitrierung (6)

- Daisy Chaining (dezentral)



Cache Kohärenz

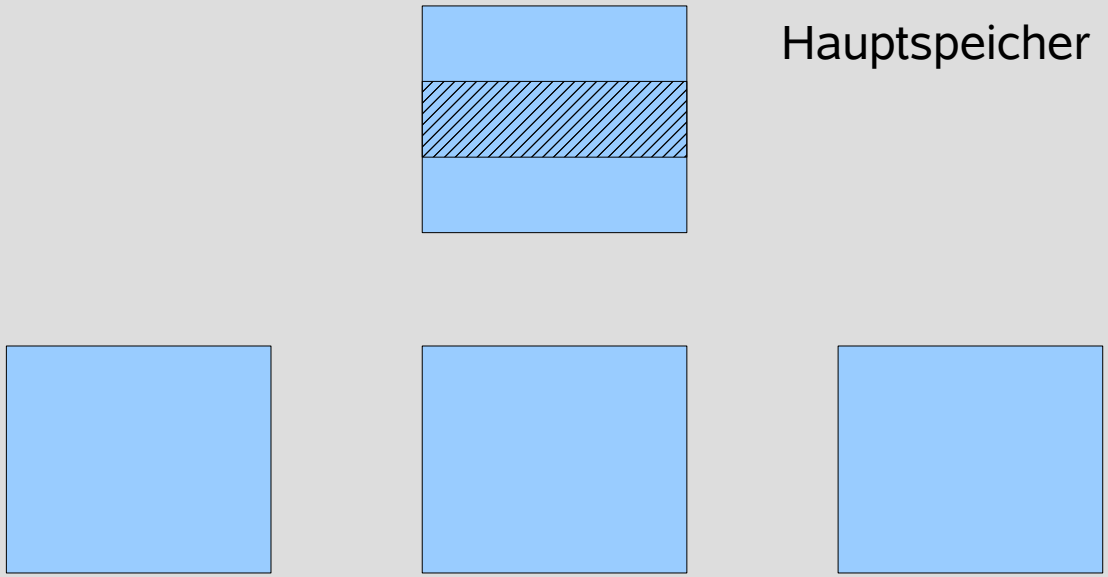
- In Multiprozessorsystemen können von einer Hauptspeichervariablen in mehreren Caches Kopien enthalten sein.
- Typisch  Write-Back-Caches
- Problematisch:
  - es können vorübergehend Inkonsistenzen zwischen Cache- und Hauptspeicherdaten auftreten
  - siehe Vorlesung 6

Cache Kohärenz (2)

- Inkonsistenzen können in Mehrprozessorsystemen zu
  - *nicht-kohärenten Mehrfachkopien* führen, **falls**
  - keine speziellen *Cache-Kohärenzmechanismen* zur Anwendung kommen.

Cache Kohärenz (3)

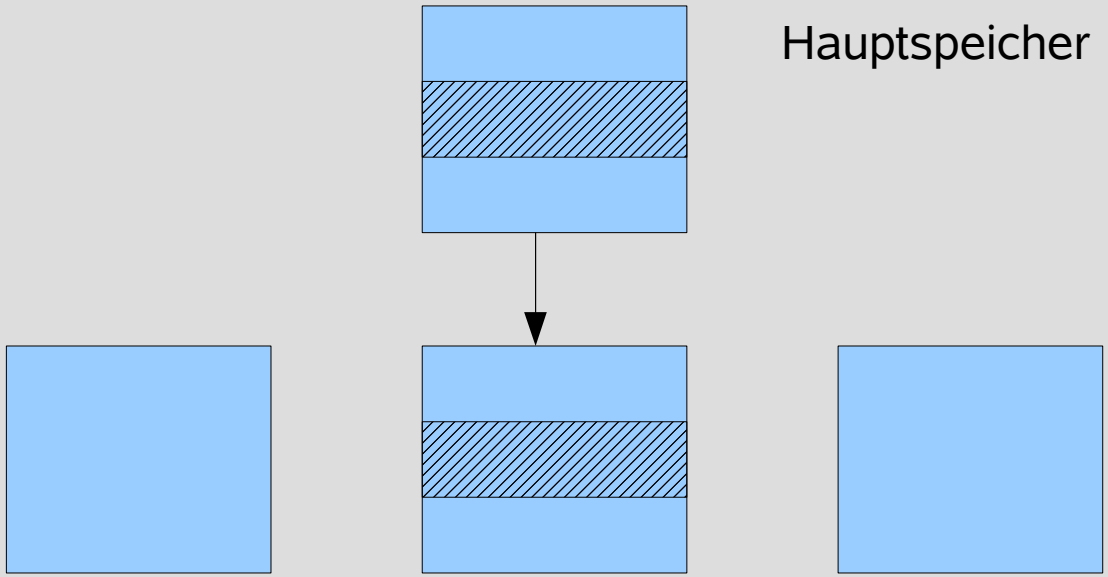
- Szenario:



Ausgangspunkt: keine Kopie vorhanden

Cache Kohärenz (3)

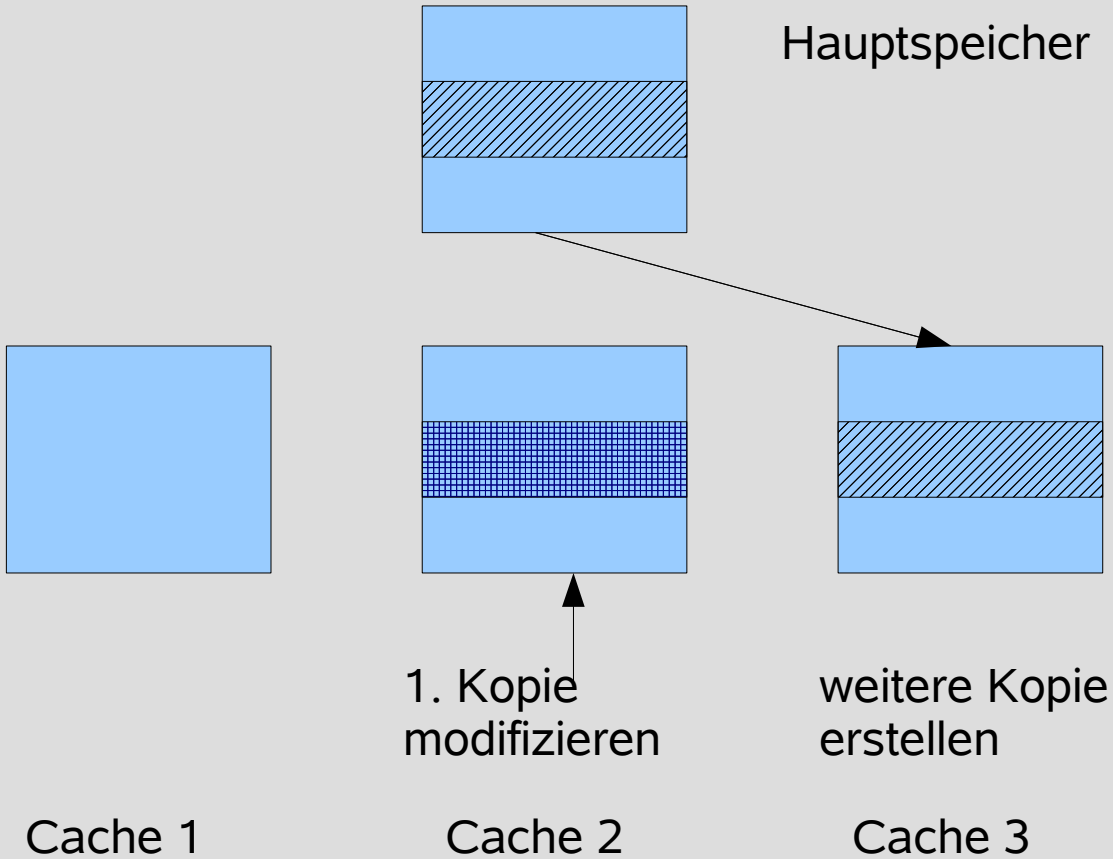
- Szenario:



eine Kopie erstellen

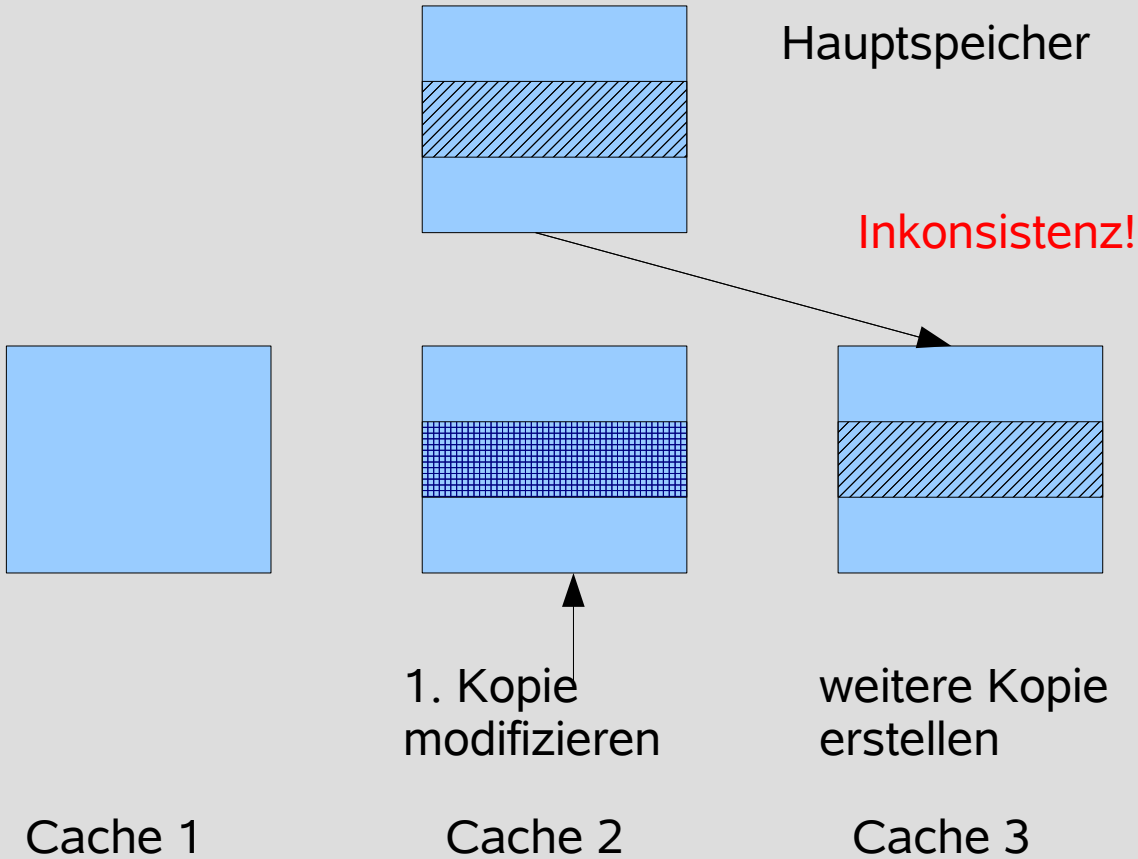
Cache Kohärenz (3)

- Szenario:



Cache Kohärenz (3)

- Szenario:



Cache Kohärenz (4)

- Methoden zur Kohärenzerhaltung
  - Ziel: Leser einer Hauptspeicherstelle erhält stets aktuellen Wert
  - Systembus und Caches werden um geeignete Funktionen erweitert
  - Systembus:
    - zusätzliche Bustransaktionen
  - Cache:
    - zusätzliche Statusbits, die von einem Finite-State-Controller im Cache in Abhängigkeit von den CPU- und Busaktionen gesteuert werden.

Cache Kohärenz (5)

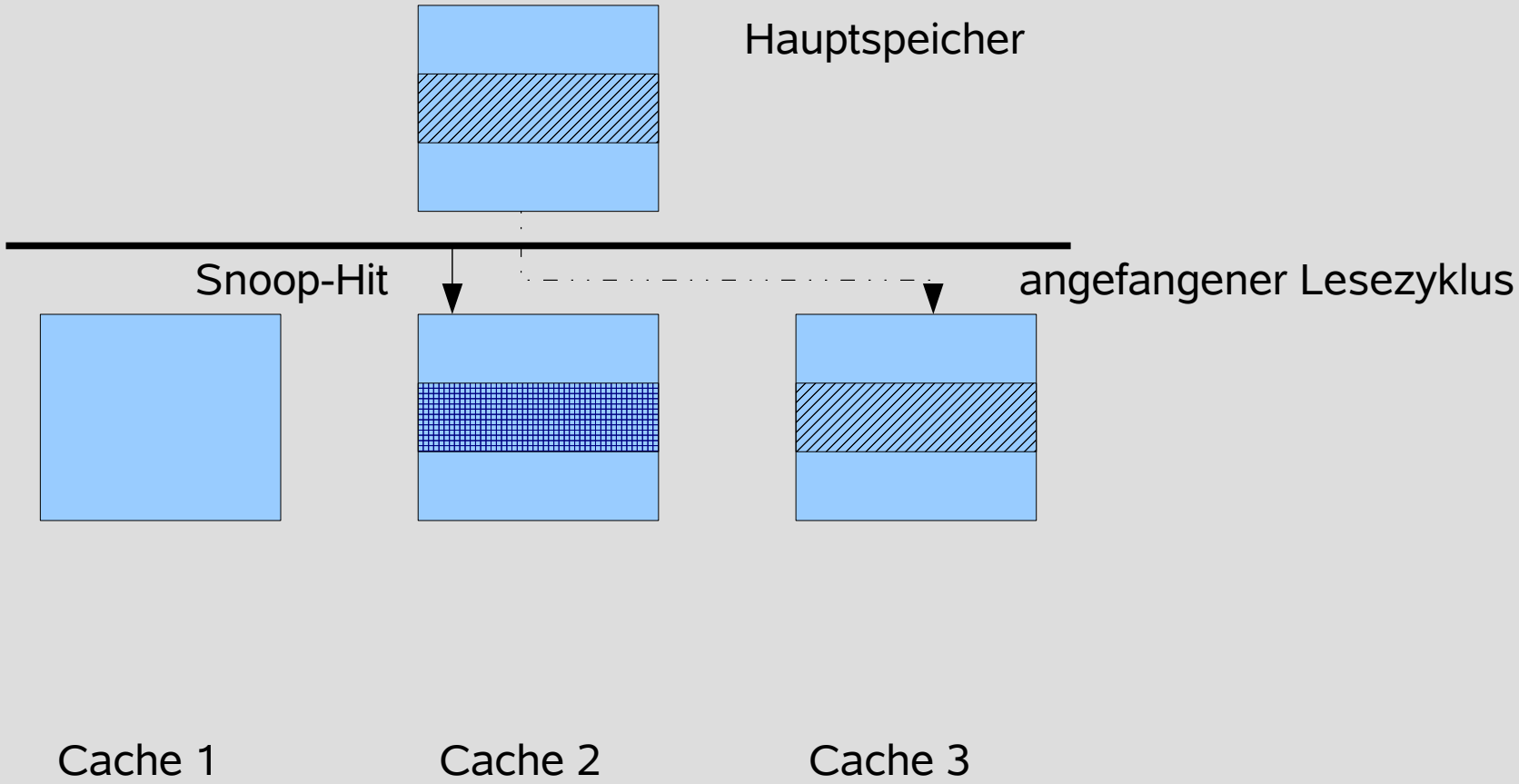
- Methoden zur Kohärenzerhaltung (2)
  - Bus entspricht von der Wirkung her einem *Broadcast-Medium*, ist von seiten der Caches ein ständiges Mithören (***Snooping***) der Busaktivitäten möglich. Dies machen sich sogenannte ***Snoop-Protokolle*** zu nutzen.

Cache Kohärenz (6)

- Snooping-Protokolle:
  - ***Write invalidate***
    der schreibende Prozessor bewirkt, dass alle Kopien in anderen Caches ungültig gemacht werden, ehe er seine lokale Kopie überschreibt.
    danach existiert nur noch die Kopie des *Schreibers*
  - ***Write broadcast***
    der *Schreiber* sendet die neuen Daten über den Bus, alle Kopien werden ständig aktualisiert
    lohnt sich nur, wenn die aktualisierten Kopien auch verwendet werden. Anderenfalls ist ***Write invalidate*** im allgemeinen schneller
  - kein eindeutig besseres Protokoll

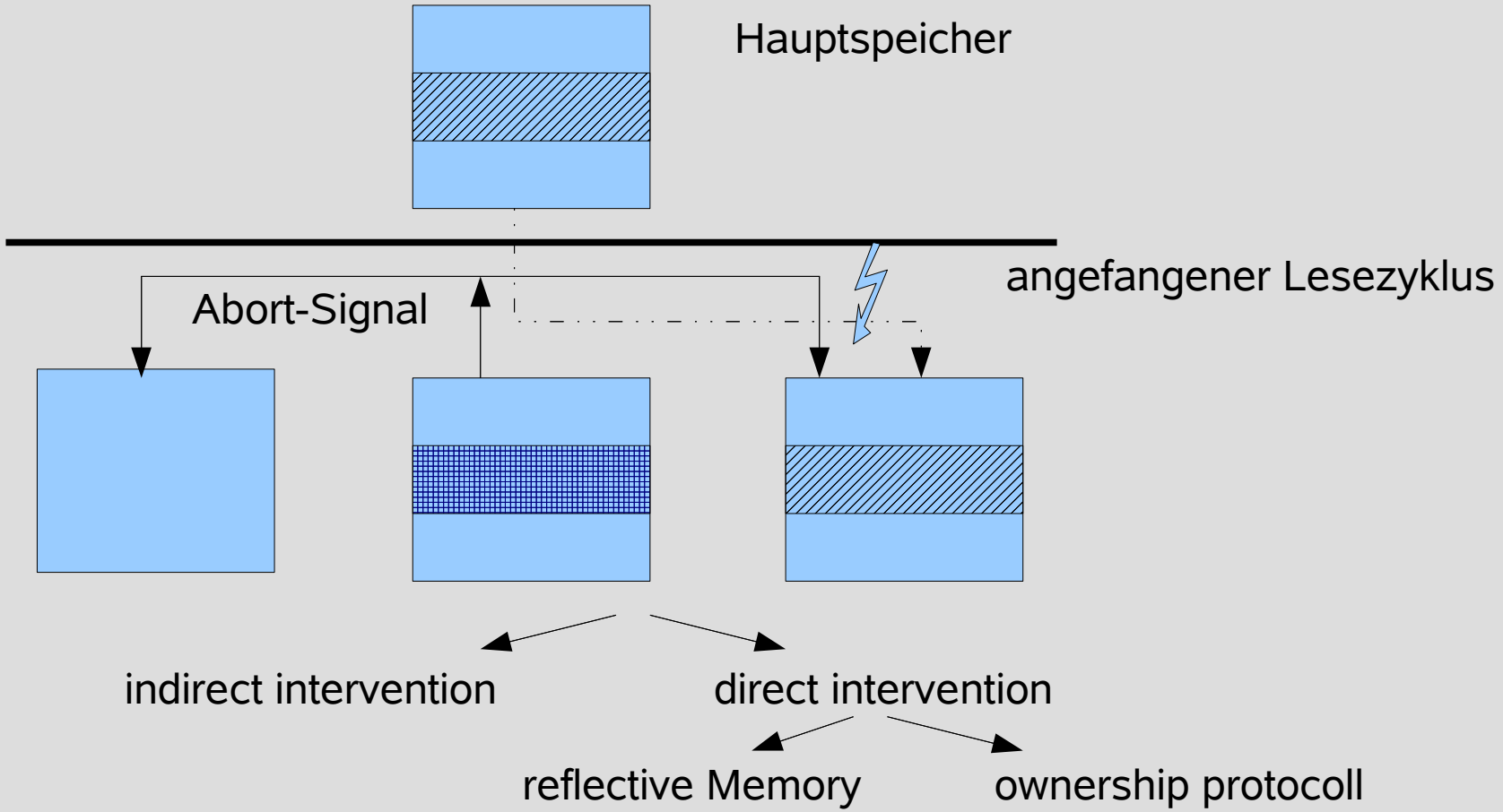
Cache Kohärenz (7)

- Prinzipielle Methoden zur Kohärenzsicherung:



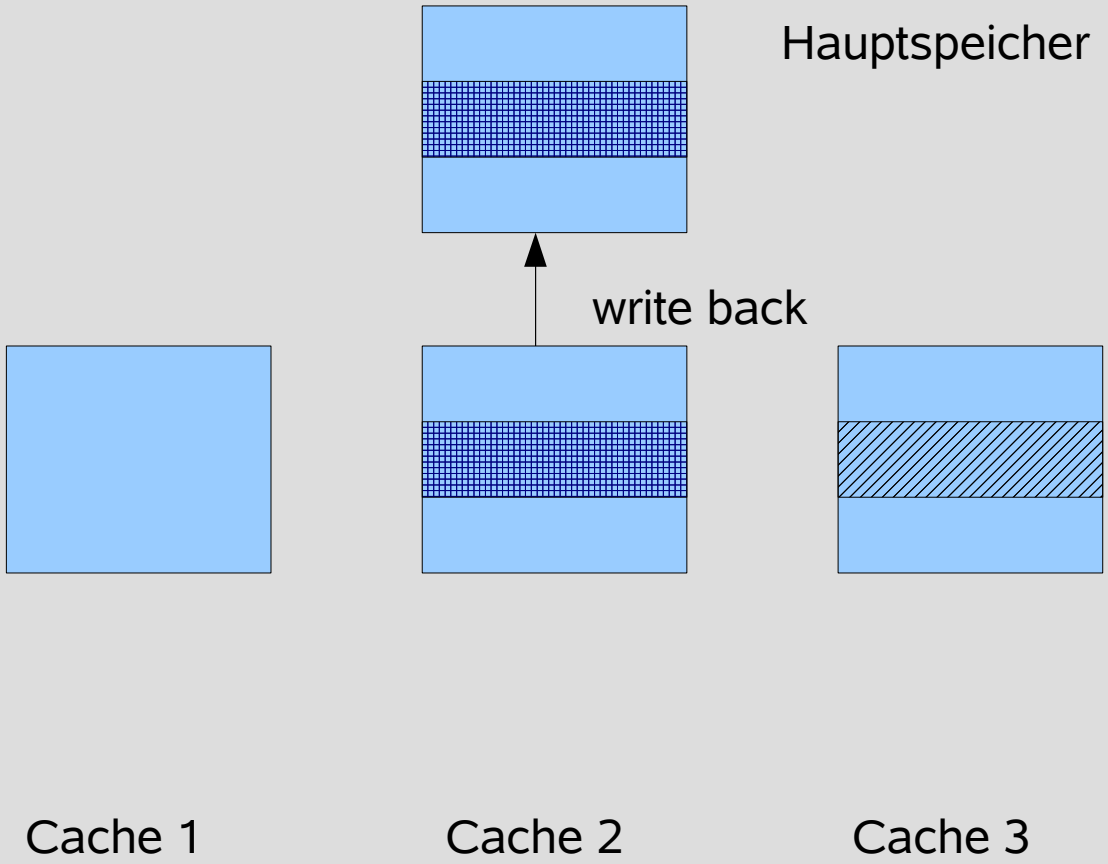
Cache Kohärenz (8)

- Intervention:



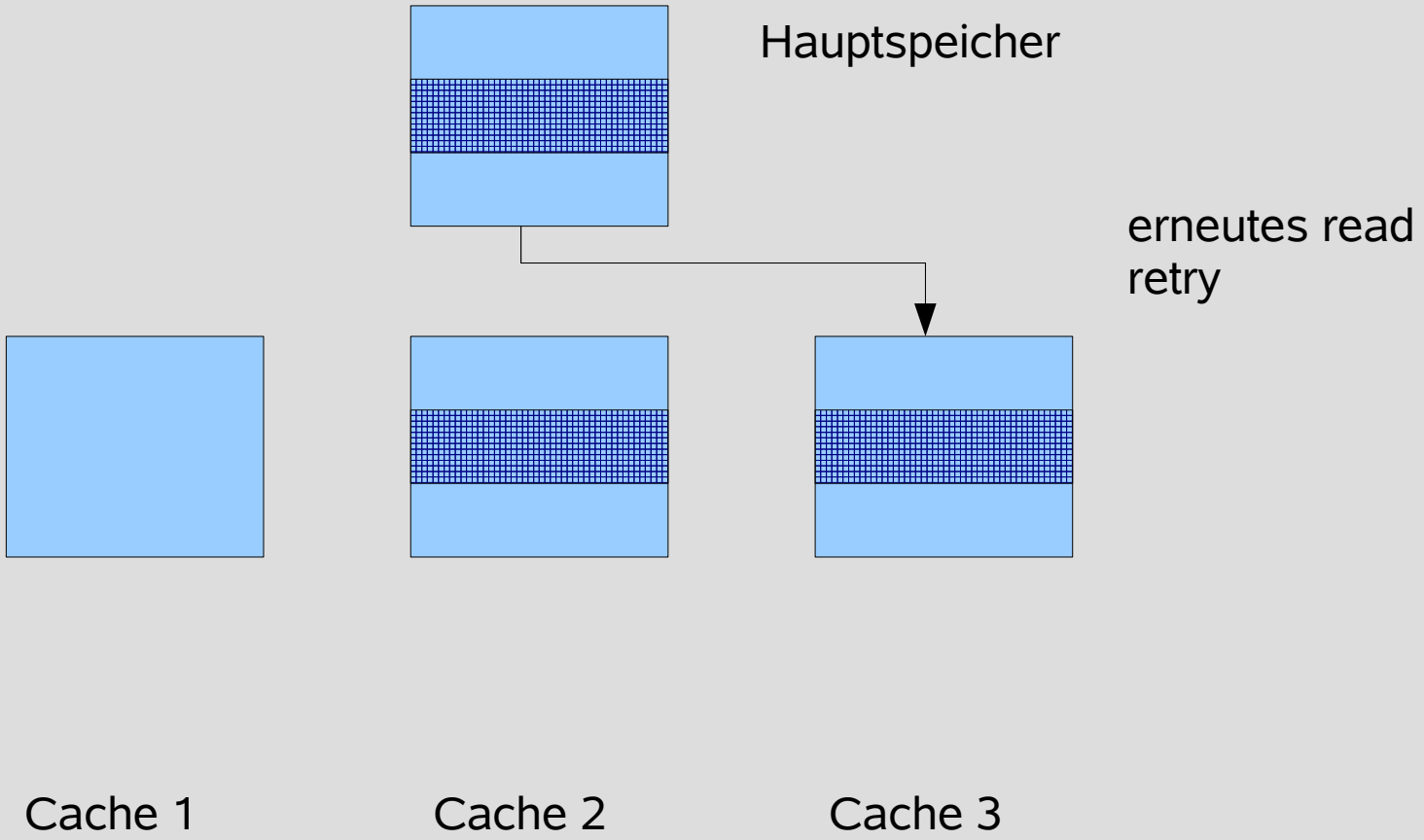
Cache Kohärenz (9)

- Indirect Intervention



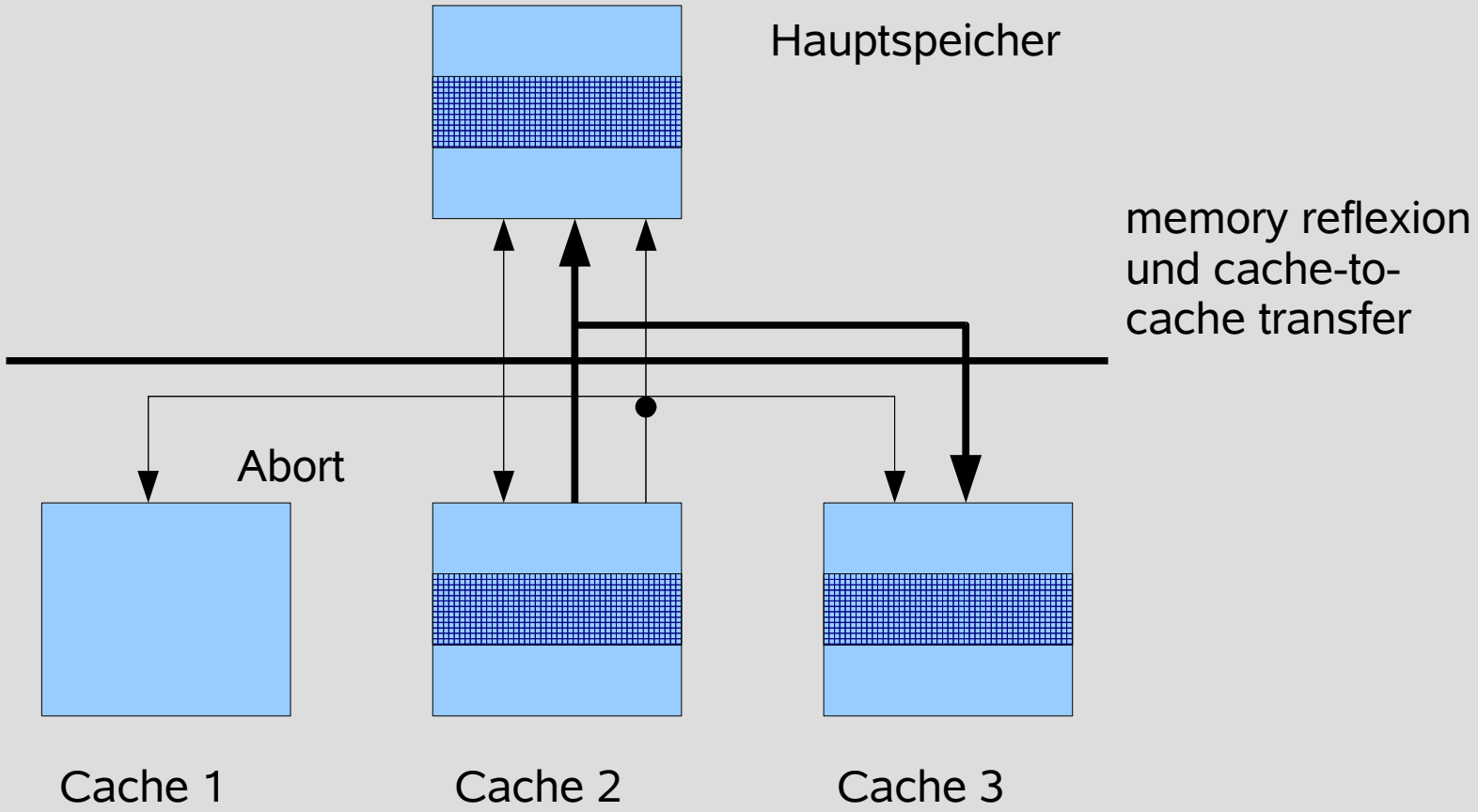
Cache Kohärenz (10)

- Indirect Intervention

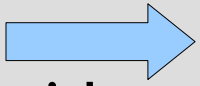


Cache Kohärenz (11)

- Reflective Memory

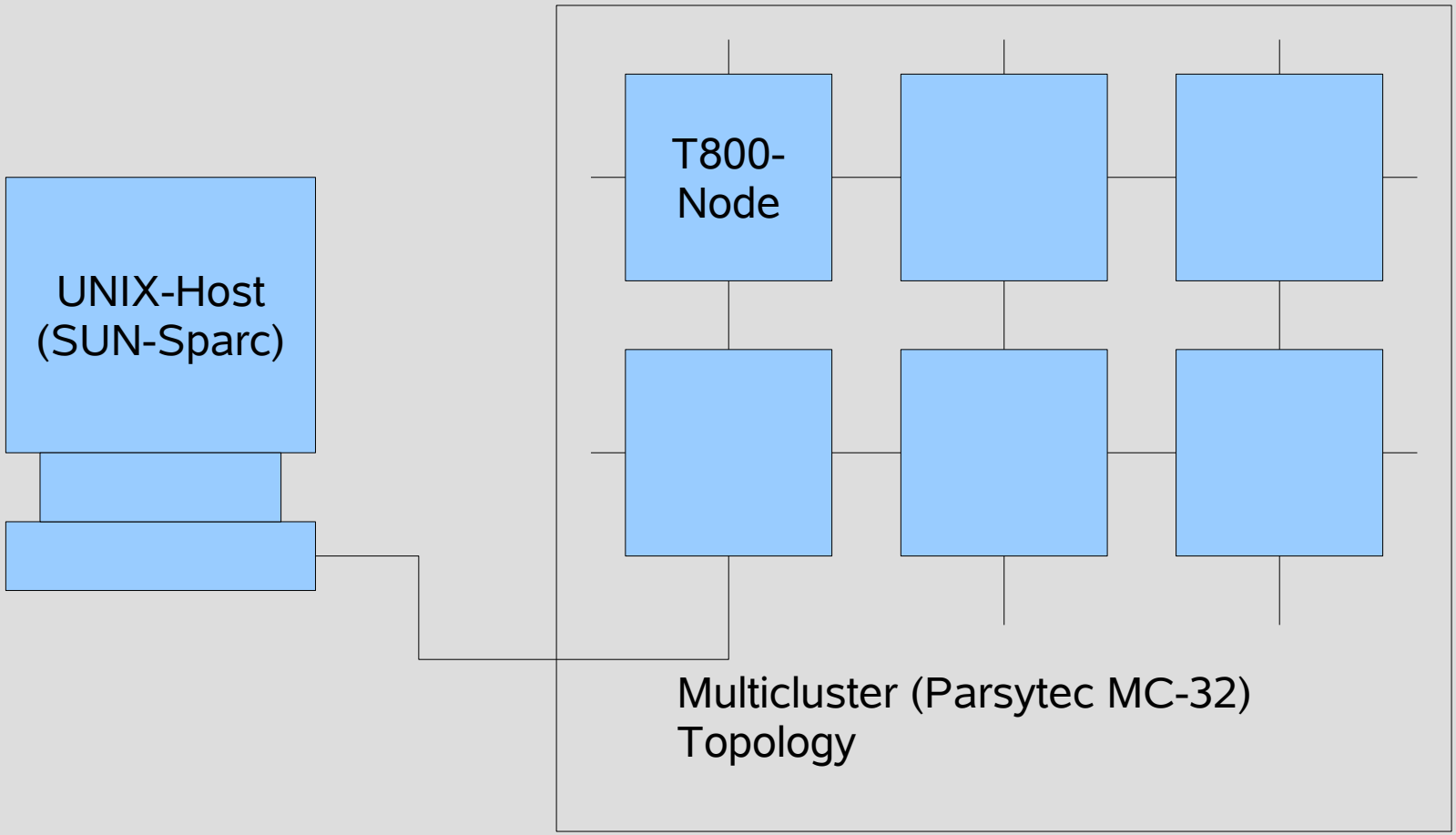


Cache Kohärenz (13)

- Synchronisation unter Nutzung der Kohärenz
  - Mutual Exclusion:
    - Prozesse, die auf Sperrvariable warten, fragen ständig den Wert dieser Variablen ab („spin waiting“)
  -  hohe Busbelastung
  - bei busbasierter Kohärenz belastet zusätzlich noch Kohärenzverkehr den Bus
  - Modifikation: Test-and_Test&Set Spin-Lock
    - Spin waiting findet in lokaler Kopie statt, drastische Reduktion des Busverkehrs
  - (siehe auch Semaphore in Betriebssystemen)

Computercluster

- Historisches: Transputer
- Hardware:

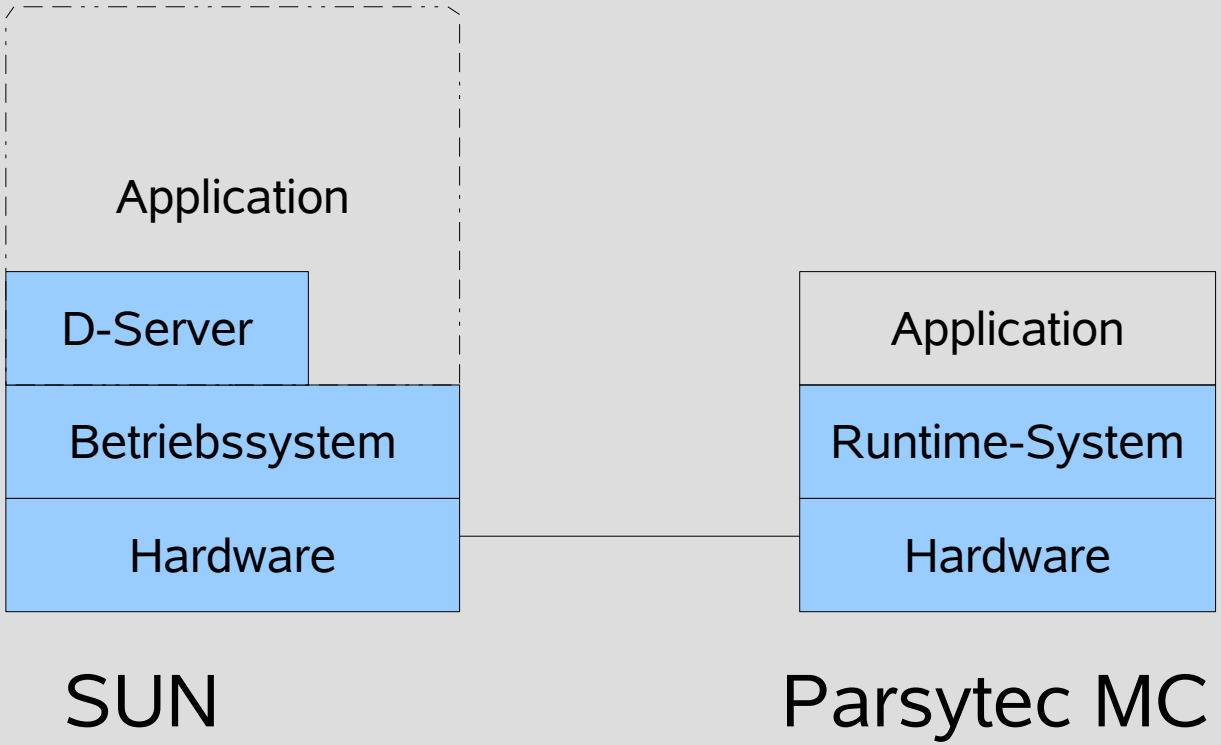


Computercluster (2)

- Node: T800 (10 MIPS) + 4 Mbyte RAM
- Link: 20 Mbyte/s
- Topologie: 2D-Grid

Computercluster (3)

- Software
  - PARIX + C **Par**allel Extensions to UN**IX**

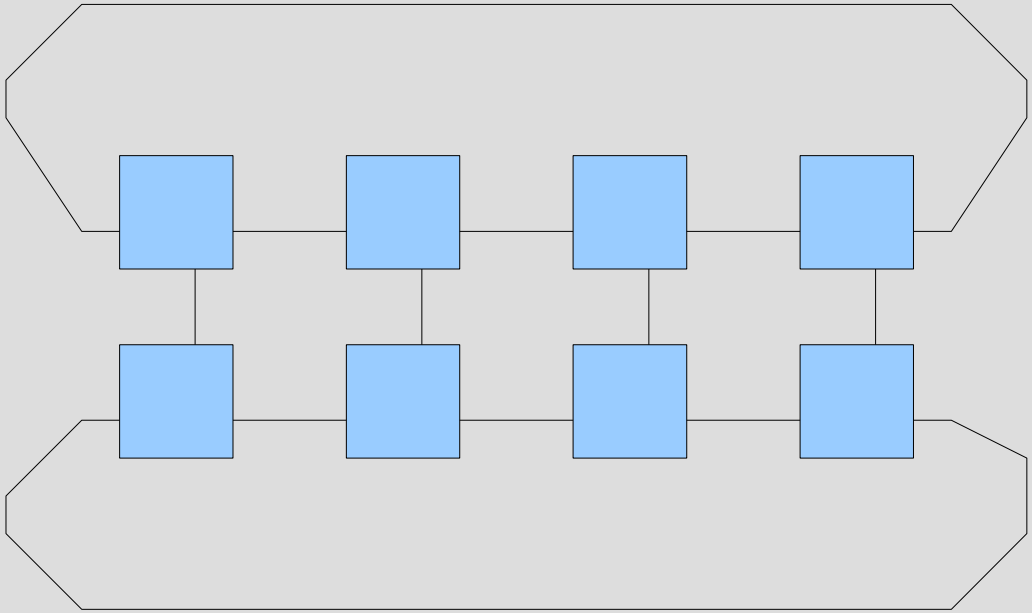


Computercluster (4)

- D-Server: **D**ata-Server, I/O-Management, Support of CSP model
- Runtime: Parallel Process Handling (Threads), Process Communication, Host Access Management (RPC, SPMD-Boot)
- Application (T800): z.B. „Domain Decomposition“ - basierte Simulation (FEM)

Computercluster (5)

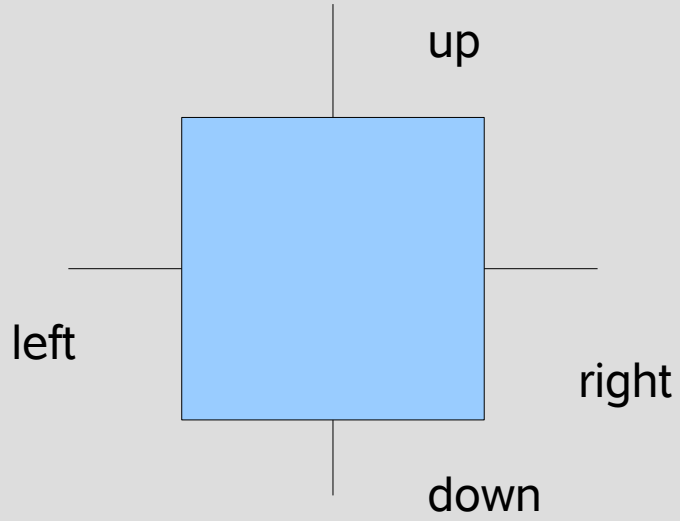
- Logische Kommunikation
  - virtuelle Kanäle



HC = 2^3
(HC - Hypercube)

Computercluster (6)

- Virtuelle Topologien
  - vordefinierte Topologien, wie Hypercube, Tree, ... mit
  - local link naming, e.g. up, right, ...



  - Communication types
    - linkbounded synchron
    - mailbox asynchron

Computercluster (7)

- Transputer sind Mikroprozessoren, die CPU und Kommunikationshardware auf einem Chip vereinigen.
- Dieses ermöglicht effiziente Verflechtungen zwischen Prozessor- und Kommunikationssteuerung.
- Eine Kommunikationsoperation kann durch einen einzigen Maschinenbefehl gesteuert werden.

Computercluster (8)

- Dieser übergibt den Kommunikationsauftrag an die Kommunikationshardware und veranlasst gegebenenfalls den mikrokodierten Scheduler, den laufenden Prozess zu inaktivieren, bis er durch die Kommunikationshardware nach Ausführung des Auftrages erneut geweckt wird.

Computercluster (9)

- Die hardwaremäßige gesamtheitliche Steuerung von Prozessverwaltung (Umschaltzeit < 600µs) und Kommunikationssteuerung führt zu minimalen Overhead bei der Synchronisation von Rechen- und Kommunikationsprozessen.
- Ein Kommunikationsbefehl überprüft automatisch, ob es sich um eine prozessorinterne oder -externe Kommunikation handelt und veranlasst entsprechend alternative Steuerabläufe. Damit ist bereits auf Maschinenniveau die Prozesskommunikation vereinheitlicht.

Computercluster (10)

- Transputer versuchen das ***Communicating Sequenetial Processes (CSP)*** – Konzept von Hoare weitestgehend durch Hardware zu unterstützen.
- Nach diesem Konzept besteht ein paralleles System aus einer Menge paralleler Prozesse, die entweder mit ihren lokalen Variablen lokal arbeiten oder Messages mit anderen Prozessen über ***synchrone unidirektionale Kanäle*** austauschen.
- Der Hardware-Scheduler unterstützt die im CSP-Modell zugelassenen bzw. definierten Kommunikationsbeziehungen (z.B. alternative oder guarded alternative) und Prozessablaufsteuerungen (z.B. seq bzw. par).

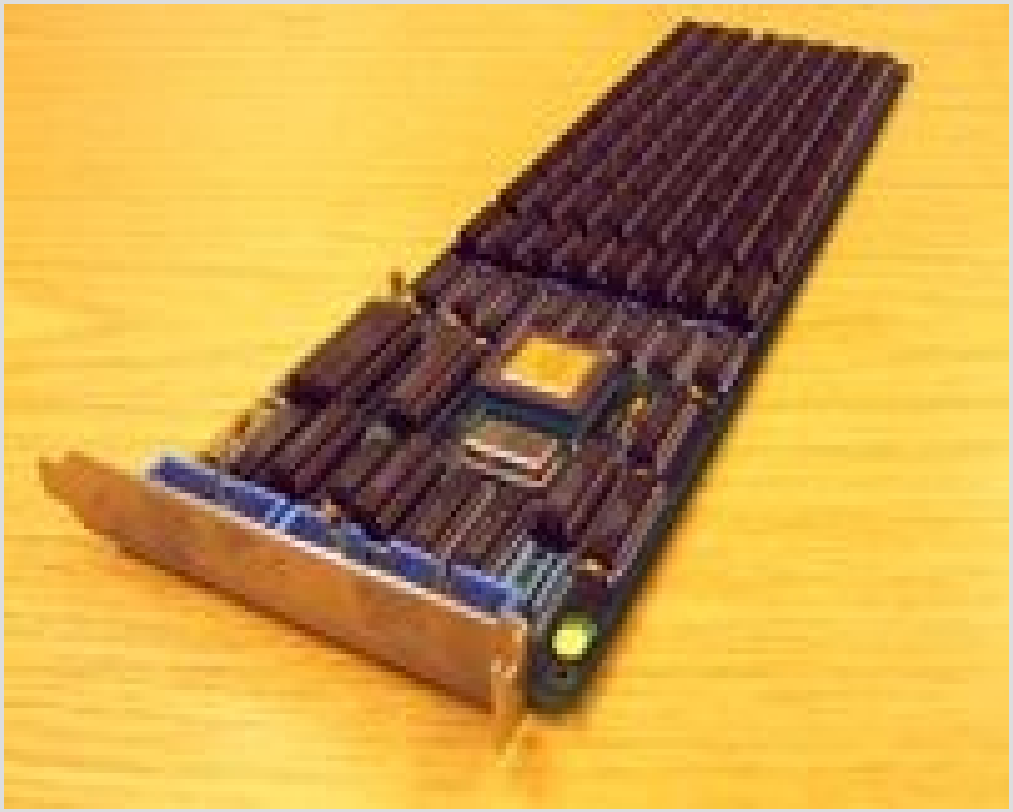
Computercluster (11)

- Transputer – entwickelt insbesondere für den effizienten Aufbau von Distributed Memory Parallelrechnern – unterstützen durch den sog. ***Boot-from-Link***-Mechanismus das einfache Booten derartiger Systeme.
- Weitere Unterstützung wird durch einen on-chip RAM, einen on-chip Timer und einen reset analyze-Mechanismus zum freeze post mortem debugging in Verbindung mit einer error daisy chain über alle Prozessoren gegeben.

Computercluster (12)

- Transputer enthielten, auf ihre Zeit bezogen (erstes Erscheinen 1985), viele architektonische Neuigkeiten.
- Sie realisieren ein relativ ausgewogenes Verhältnis zwischen Rechen- und Kommunikationsleistung.
- Als proprietäre Parallelverarbeitungschips konnten sie den enormen Leistungsentwicklungen des Mainstream-RISC-Prozessor-Marktes nicht standhalten.

Computercluster (13)



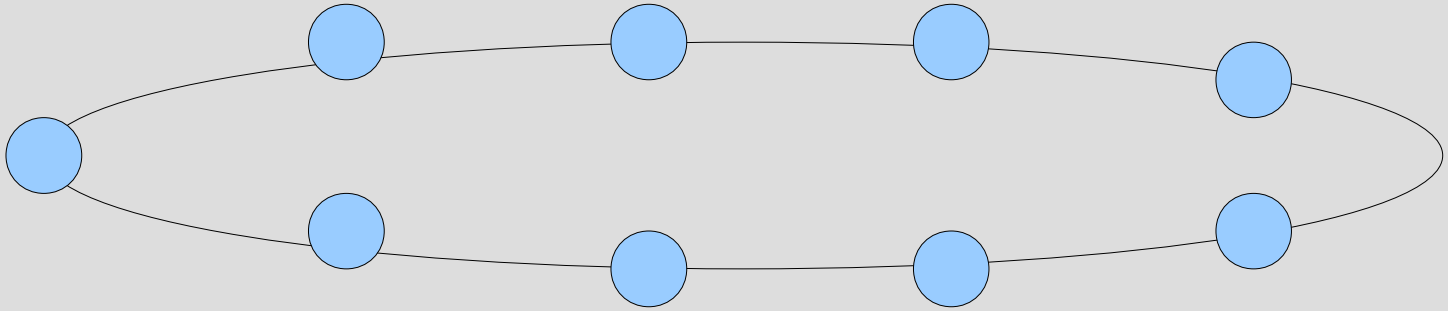
Transputerkarte für PC

Computercluster (14)

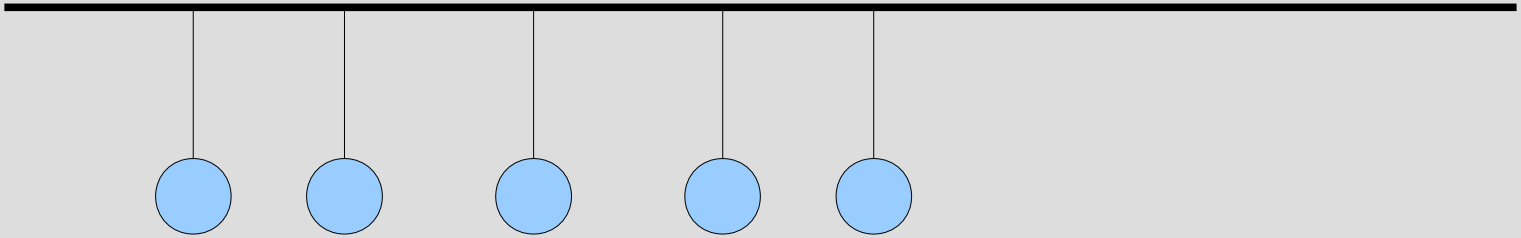
- weitere Informationen zu Transputern:
  - http://www.classiccmp.org/transputer
  - http://vl.fmnet.info/transputer/

Netzwerktopologien

- Ring
  - Übertragungen verlaufen gleichzeitig

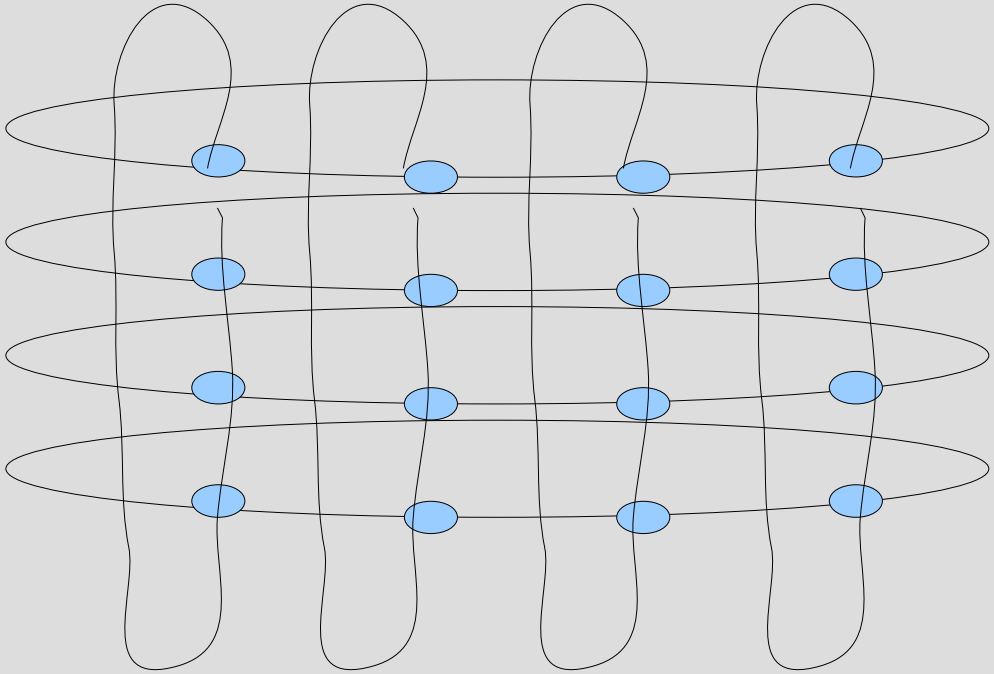


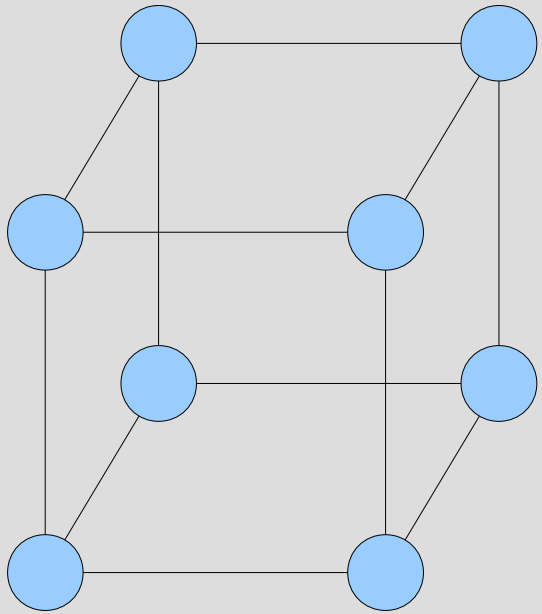
- Bus



Netzwerktopologien (2)

- Vollständig verbundenes Netzwerk
  - Jeder Knoten ist mit jedem anderen verbunden
  - Enorme Leistungsverbesserung (siehe später)
  - Enorme Steigerung der Kosten
- Kommerziell verwendet:



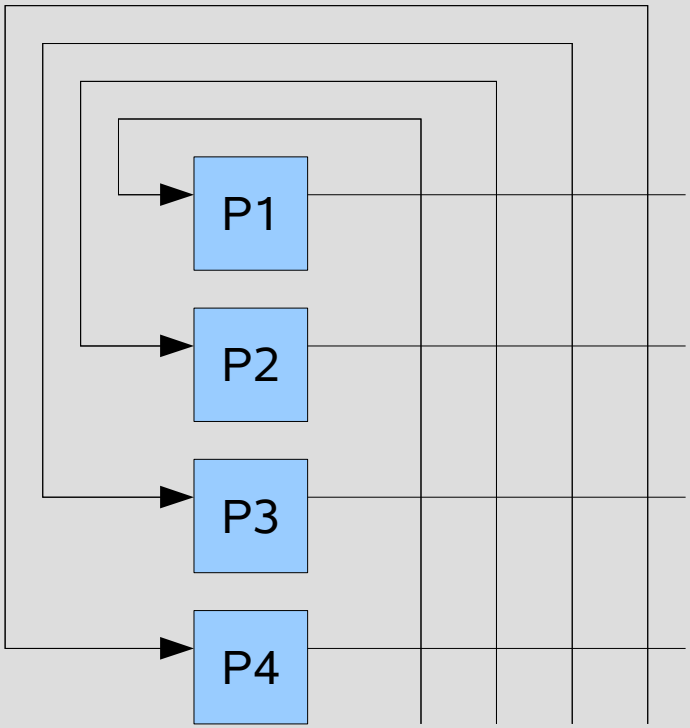


Netzwerktopologien (3)

- Mehrstufige Netzwerke
  - Statt wie bisher Prozessor-Switch-Knoten zu verwenden sind hier nur Switches im Einsatz (kleiner)

Netzwerktopologien (3)

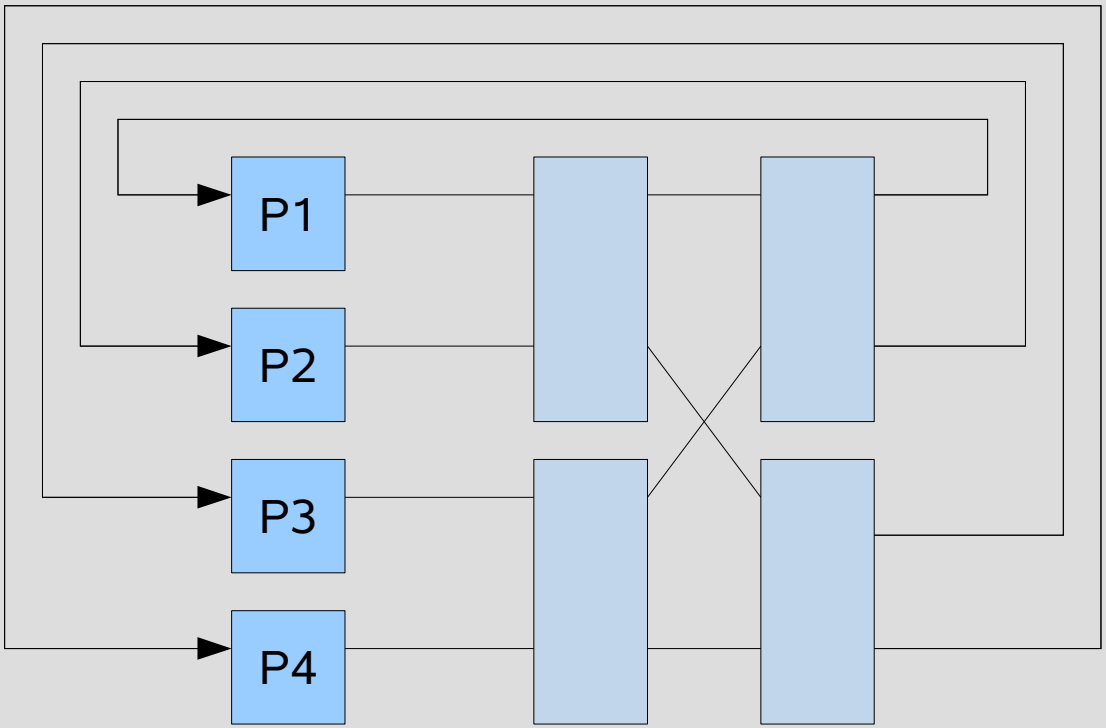
- Mehrstufige Netzwerke
  - Statt wie bisher Prozessor-Switch-Knoten zu verwenden sind hier nur Switches im Einsatz



Kreuzschienenverteiler

Netzwerktopologien (3)

- Mehrstufige Netzwerke
  - Statt wie bisher Prozessor-Switch-Knoten zu verwenden sind hier nur Switches im Einsatz



Omega-Netzwerk

Netzwerktopologien (4)

- Vergleich
  - Netzwerkbandbreite
    - Bandbreite * Anzahl der Verbindungen (Best Case)
      - Ring: P * Bandbreite einer Verbindung
      - Bus: Bandbreite des Busse
      - Vollständig: P * (P-1)/2
  - Bisektionsbandbreite
    - Teilung des Clusters in 2 Teile, Addition der Bandbreiten der Verbindungen zwischen den Teilen
      - Ring: 2 * Bandbreite einer Verbindung
      - Bus: Bandbreite des Busses
      - Vollständig: $(P/2)^2$
    - Bei nichtsymmetrischen Clustern: Trennlinie zwischen Teilen, die den Worst Case bedeuten

Chip-Multiprozessoren (CMP)

- Mehrere Prozessoren (Kerne, Cores) auf einem Chip
  - Häufig teilen sich die Prozessoren Teile oder den gesamten Cache sowie die externe Speicherschnittstelle
  - Speicherhierarchie?
    - Gleicher Code wird ausgeführt (Datenbankanwendung) --> gemeinsame Befehlszugriffe
    - Gleiche Datenstrukturen: Probleme werden geringer bei gemeinsamen Cache

Mehrfädigkeit (multithreading)

- Möglichkeit, dass mehrere Threads die funktionalen Einheiten des Prozessors gemeinsam nutzen.
- „Hardwarethreads“:
  - vom Compiler erzeugte sequentielle Befehlsfolgen
  - Betriebssystem-Threads
  - Prozesse

Mehrfädigkeit (multithreading) (2)

- Prozessor muss Zustand jedes Threads auf dem Chip bereitstellen
  - Separate Registersatzkopie
  - Separater Befehlszähler
  - Separate Seitentabelle
  - Speicher wird durch virtuelle Speicherverwaltung gemeinsam genutzt
  - Threadwechsel muss von Hardware unterstützt sein (effizienter als Prozesswechsel)

Mehrfädigkeit (multithreading) (3)

- Feinkörnige Mehrfädigkeit finegrained m.)
  - Nach jedem Befehl wird Thread gewechselt
- Grobkörnige Mehrfädigkeit (coarsegrained m.)
  - Threadwechsel nur beit aufwändigen Stillständen der Pipeline (z.B. Fehlzugriff im Sekundärcache)

Mehrfädigkeit (multithreading) (4)

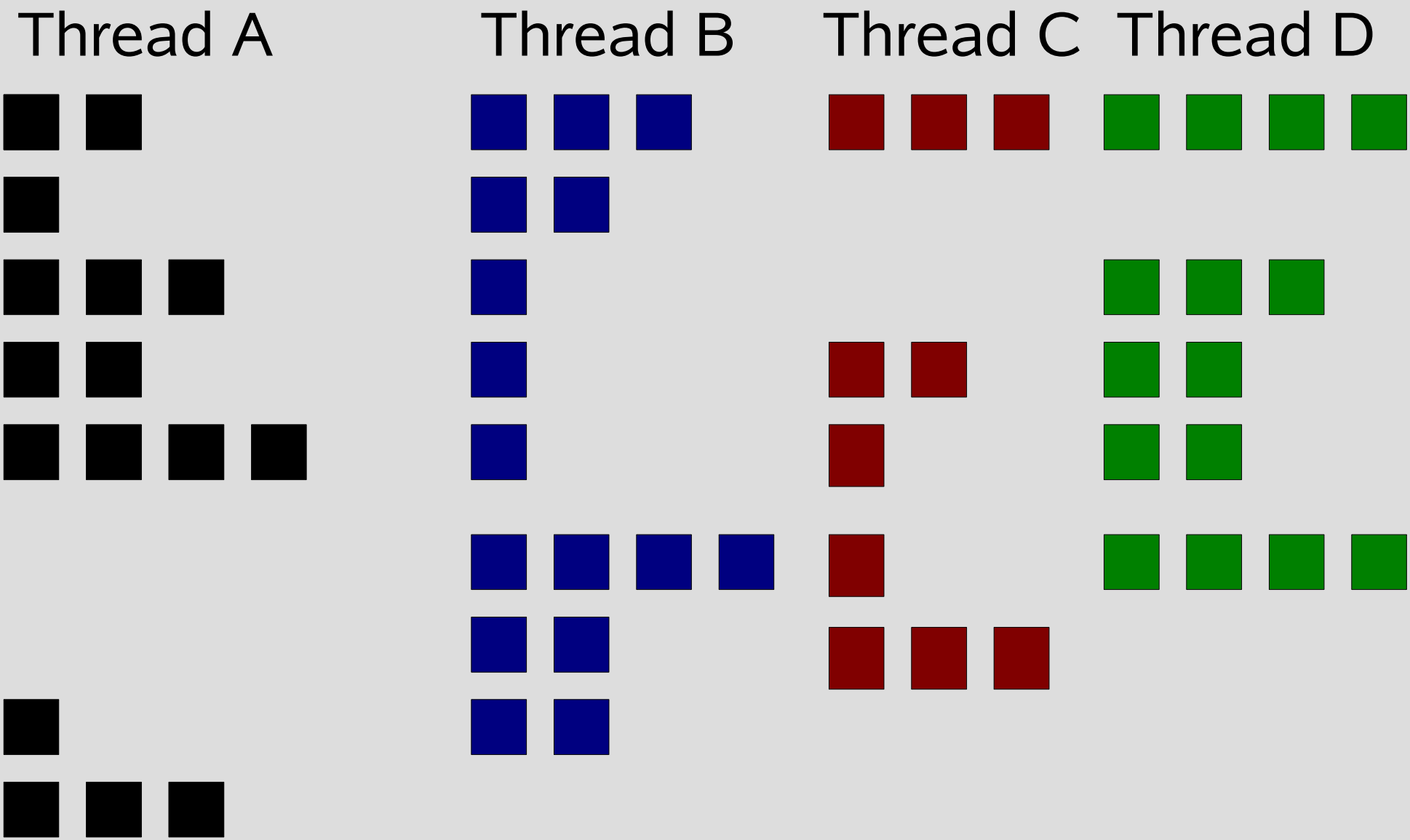
- Vorteile:
  - Feinkörnig: wenig Probleme mit Pipelines (bleiben nicht stehen)
  - Grobkörnig: kein sehr aufwändiger Threadwechselmechanismus

Mehrfädigkeit (multithreading) (5)

- Simultane Mehrfädigkeit (simultaneous m., SMT)
  - Mehrere Befehle verschiedener Threads können gleichzeitig den verschiedenen funktionalen Einheiten eines mehrfachzuweisungsfähigen Prozessors zugeordnet werden.

Multithreading (6)

- Beispiel: superskalare Ressourcen, ohne MT



Multithreading (7)

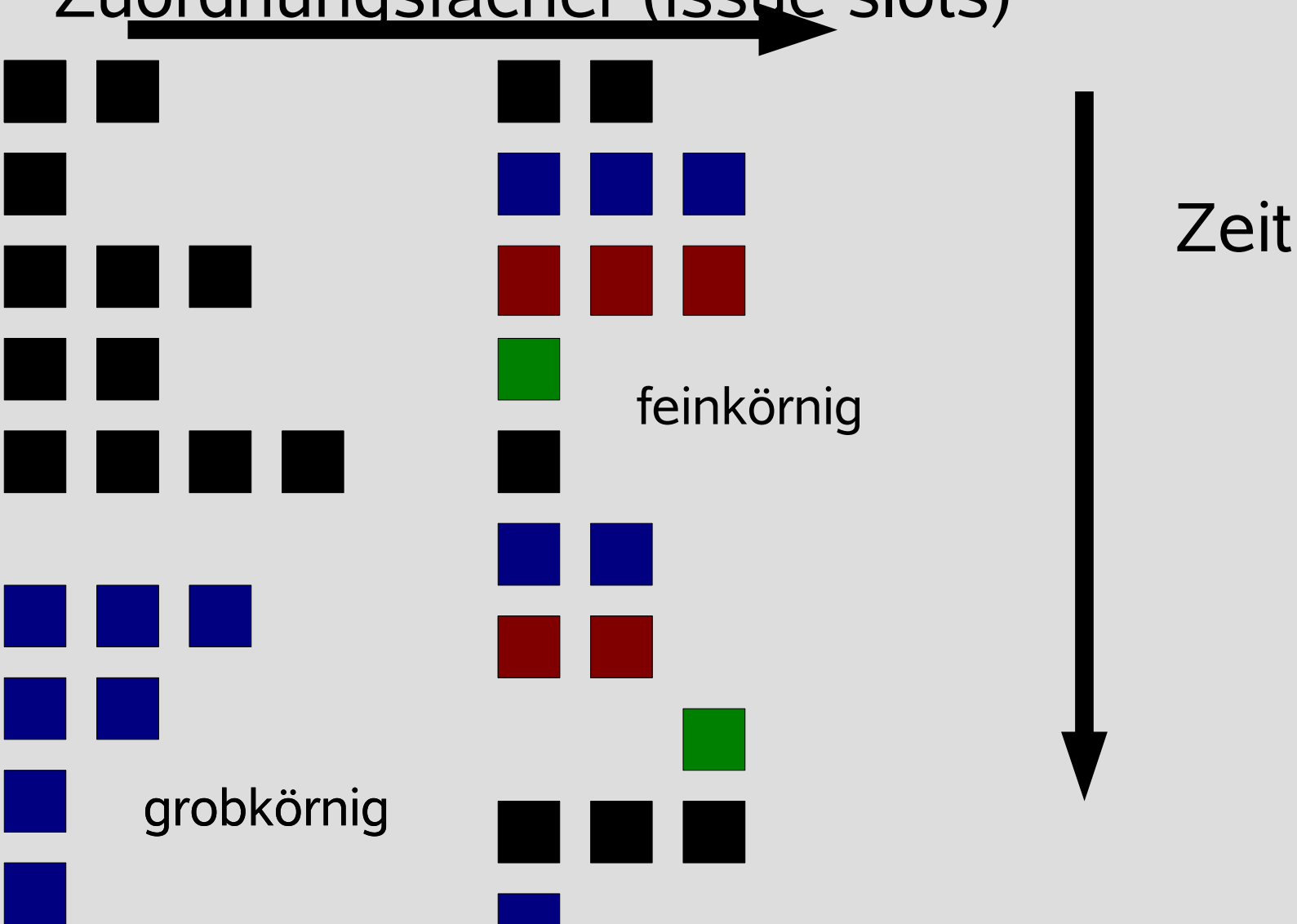
- Beispiel: superskalare Ressourcen

Zuordnungsfächer (issue slots)

Multithreading (7)

- Beispiel: superskalare Ressourcen

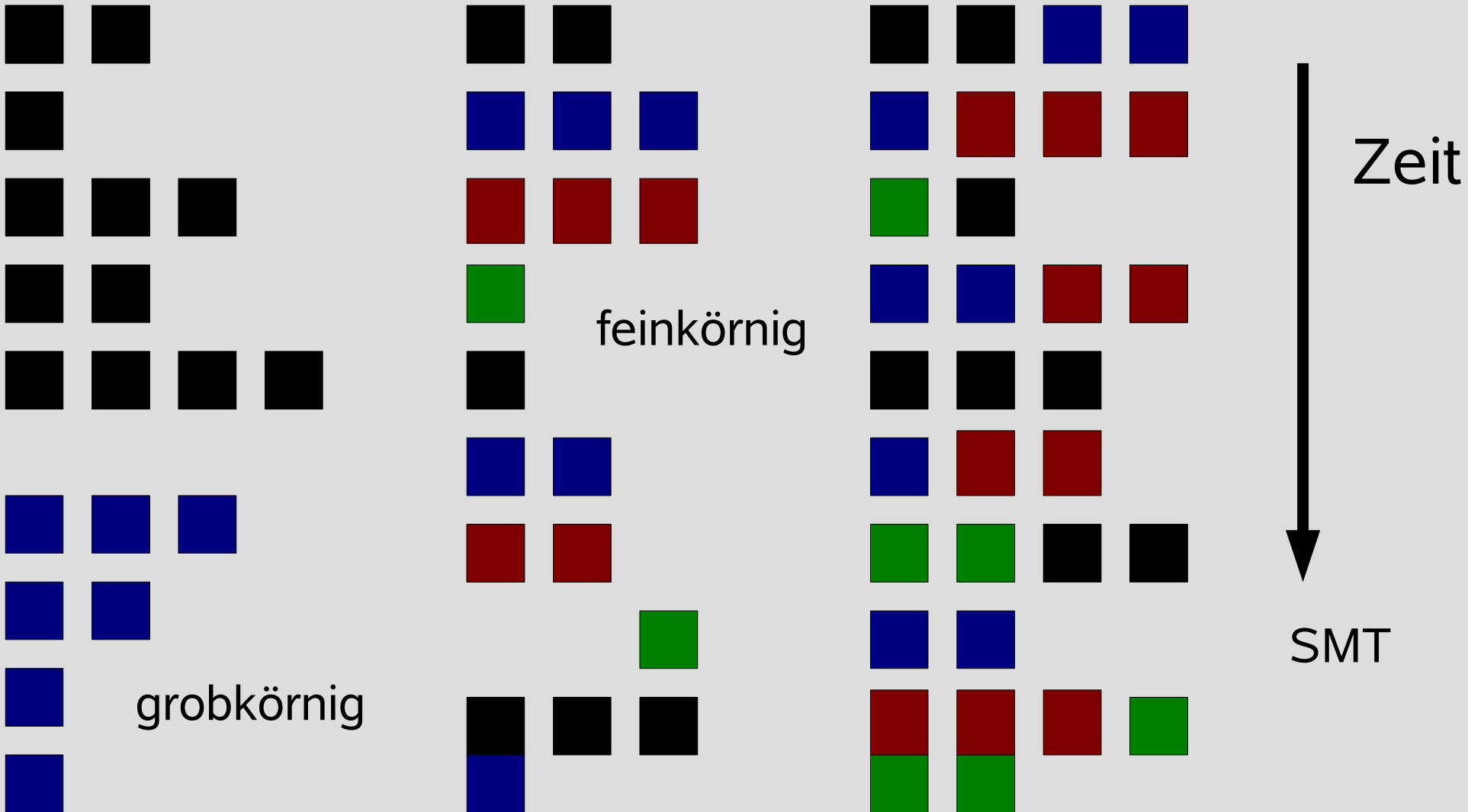
Zuordnungsfächer (issue slots)



Multithreading (7)

- Beispiel: superskalare Ressourcen

Zuordnungsfächer (issue slots)

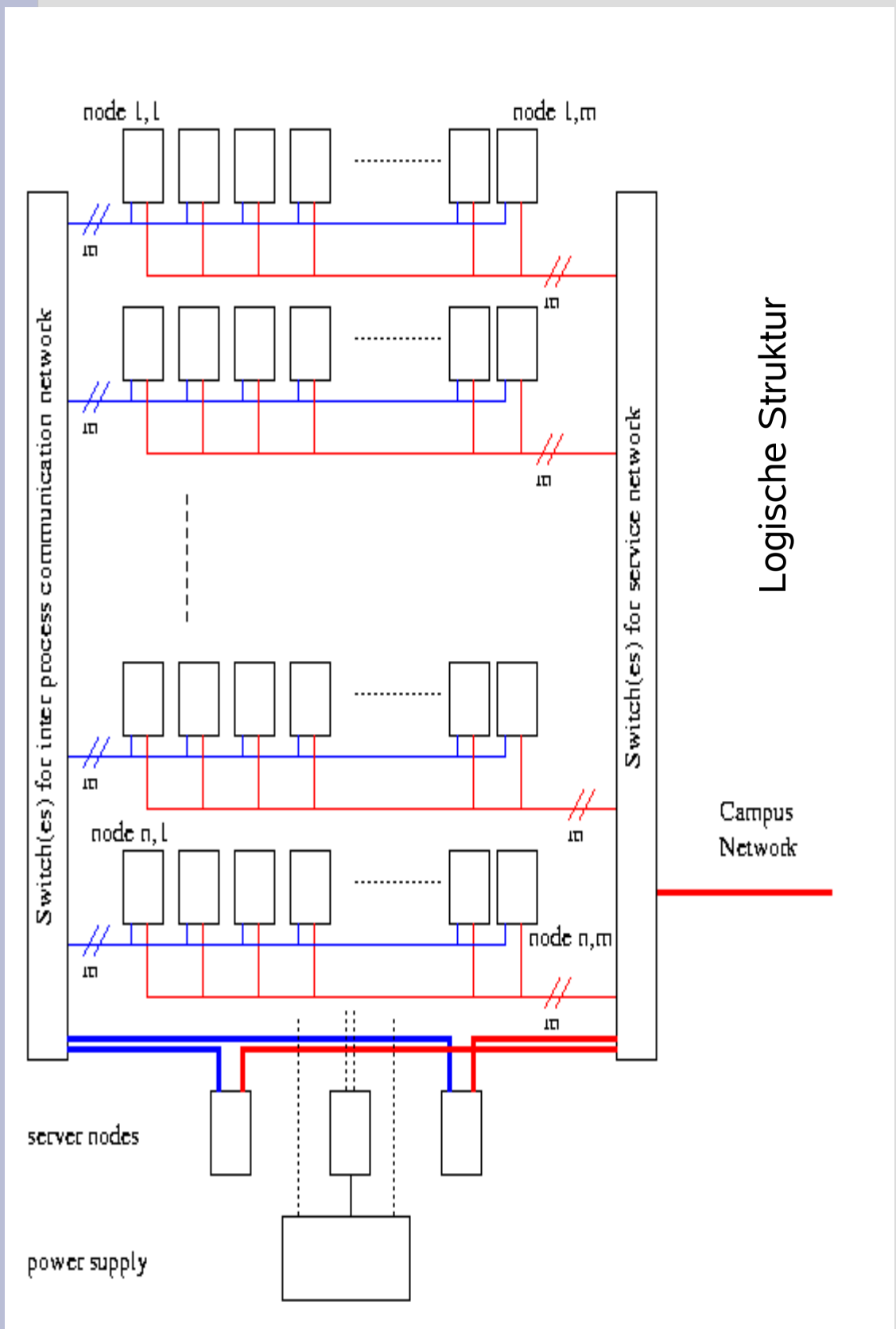


Multithreading (8)

- Beispiel: superskalare Ressourcen
  - **Grobkörnig:** jeweils ein Thread wird vollständig abgearbeitet
  - **Feinkörnig:** Verzahnung von Threads, zu einem Taktzeitpunkt werden nur Befehle eines Threads ausgeführt
  - **SMT:** Nutzung der Parallelität auf Threadebene (TLP) und Befehlsebene (ILP). Zu einem Taktzeitpunkt können Befehle unterschiedlicher Threads ausgeführt werden

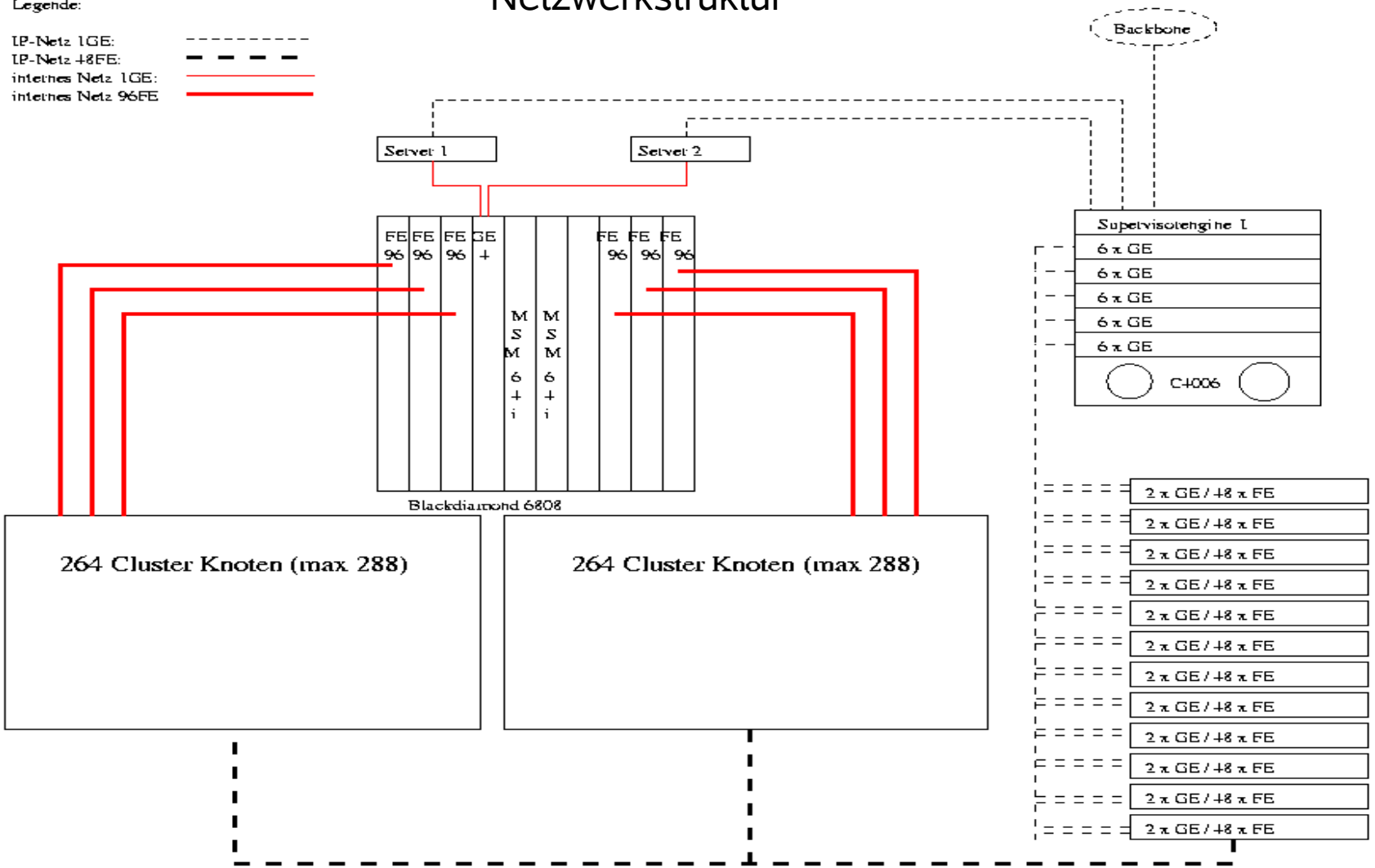
Computercluster (15)

- Chemnitzer Linux Cluster CLIC
  - „Eigenbau“-Parallelrechner (Eigenbau insofern, dass das Konzept von der TU Chemnitz erarbeitet wurde)
  - Jeder der 528 Standard-PC's ist über 2 Netzwerkkarten (Fast-Ethernet) in den Parallelrechner integriert
    - Kommunikationsnetz für parallele Anwendungen
    - Servicenetz für Patches, externen Zugang, ..)
  - Durch spezielle Bibliotheken Aufbau virtueller Netzwerke



Logische Struktur

Netzwerkstruktur



Computercluster (19)

- **C**hemnitzer **H**ochleistungs L**i**nux **C**omputer CHIC
- offizielle Inbetriebnahme am 7.2.2007
- neuartige Prozessoren (AMD Opteron Rev. F) und Netzwerktechnologien (Infiniband)

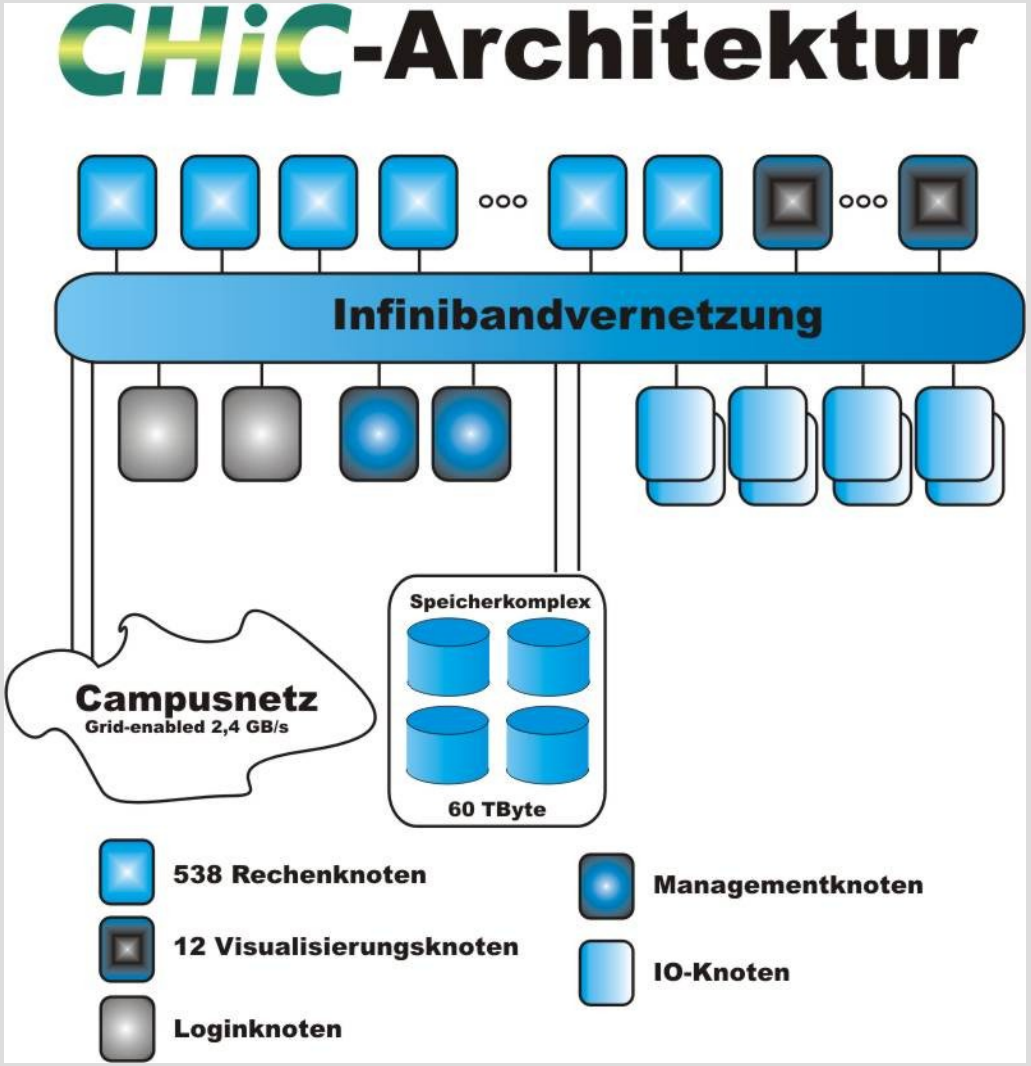
Computercluster (20)

- 538 Computer-Knoten (4GB RAM, diskless)
- 2 Management-Knoten (6 GB RAM, 4* 300 GB SAS)
- 2 Login-Knoten (4 Prozessoren, 16 GB RAM, 4*300 GB SAS)
- 8 I/O-Knoten (16 GB RAM, 80 GB SATA)
- 12 Visualisierungs-Knoten (4 GB RAM, 2 * 250 GB SATA, nVidia Corporation Quadro FX 4500 X2)
  - Prozessor: (2) Dual-Core AMD Opteron™ Processor 2218 Stepping 2

Computercluster (21)

- 10 * Xiranet XAS1000
  - Intel Server chassis SR2500
  - Intel Server Board S5000 Series
  - Intel Xeon Processor 5100 Series
  - Intel Pro/1000MT
  - Mellanox InfiniBand HCA
  - LSI Logic Inc. MegaRAID SAS Storage Adaptes
  - Xyratex SAS/SATA-JBOD with 16 * 500 GB Seagate Barracuda ES Series

Computercluster (22)



Computercluster (23)

Beispiel Cluster: Chic

Computercluster (24)

- Schlussbetrachtung
  - Gesetz von Amdahl gilt auch hier
  - Marketing ist nicht gleich reale Leistung

Typ	Spitzenwert MFLOPS	Harmonisches Mittel MFLOPS für die Perfect-Club BM	Prozent Spitzen-MFLOPS
CRAY X-MP/416	940	114,8	1%
IBM 3090-600S	800	8,3	1%
NEC SX/2	1300	16,6	1%

Computercluster (25)

- Schlussbetrachtung (2)
  - Effektivste Möglichkeit, Rechner zu bauen, der mehr Leistung als ein Einzelchip-Mikroprozessor ist, ist der Aufbau eines Multiprozessors oder Clusters
  - Multiprozessoren und Cluster sind äußerst effektiv für Arbeitslasten aus vielen Programmen.
  - Vor allem für wissenschaftliche und ingenieurmäßige Berechnungen genutzt