Multiprozessoren

Gregor Michalicek Marcus Schüler

Vorteile gegenüber Singleprozessoren

- Multiprozessoren sind zuverlässiger. Einige Multiprozessorsysteme können trotz defekter Hardware weiterhin korrekt weiterarbeiten, wenn nämlich ein einzelner Prozessor in einem System mit n Prozessoren ausfällt, arbeitet es mit n-1 Prozessoren weiter

- Fällt ein Single-Chip Rechner aus, muß er in einem teuren und zeitlich aufwendigen Prozeß ersetzt werden.
  Ein Multiprozessorsystem kann in einzelnen Schritten durch Austausch einzelner Komponenten repariert oder erweitert werden, ohne eine komplette Systemunterbrechung herbeizuführen.

- Multiprozessoren besitzen die höchste absolute Leistung – schneller als jeder Single-Chip-Prozessor

- Ein Multiprozessor aus mehreren einzelnen Prozessoren ist effektiver als ein high-performance Singleprozessor mit einer neu entwickelten Technologie.

- Mehr Rechenleistung wird einfach durch die Erhöhung der Anzahl der Prozessoren erreicht. Der Verbraucher ordert soviel Prozessoren wie das Budget erlaubt und erhält die dazu vergleichbare Leistung.

Warum setzen sich Multiprozessoren nicht durch ?!

- Nur wenige wichtige Programme wurden bisher für Multiprozessoren umgeschrieben.
- Viele Programme können prinzipiell nicht durch Multiprozessoren beschleunigt werden.
- Programmierung von Programmen für Multiprozessoren ist schwierig, da der Programmierer z.B. die zugrundeliegende Hardware genau kennen muß.
- Programme sind nicht einfach von einem Multiprozessor auf den nächsten übertragbar.

Wie stark kann ein Programm durch Multiprozessoren beschleunigt werden?

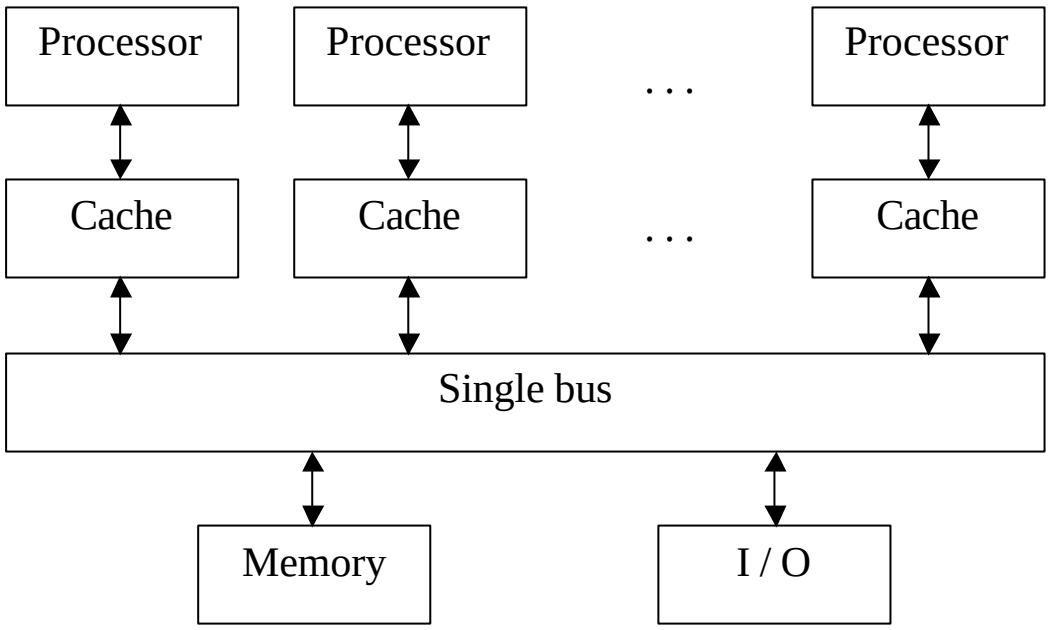
- Amdahl's Gesetz :

$$Zeit\_nach\_Optimierung = \frac{optimierter\_Anteil}{Verbesserungsfaktor} + restliche\_Zeit$$

- Es können nur die Teile eines Programms durch Multiprozessoren beschleunigt werden, die parallel berechnet werden können.
- Sequentielle Programmteile können nicht beschleunigt werden.

>>> Ein Programm darf keine sequentiellen Teile enthalten, damit es mit steigender Anzahl der Prozessoren (in einem Multiprozessor) linear beschleunigt werden kann.

Single-bus Multiprozessor



Wie viele Prozessoren sind sinnvoll?

- Die Bandbreite des Busses beschränkt die sinnvolle Anzahl an Prozessoren.
- Die Menge an Daten, die jeder Prozessor über den Bus schickt beschränkt die sinnvolle Anzahl an Prozessoren.
- Der Bus wird durch die Caches entlastet.
- Single-bus Multiprozessoren bestehen normalerweise aus 2 bis 32 Prozessoren
- Die maximale sinnvolle Anzahl an Prozessoren bei Single-bus Multiprozessoren scheint mit der Zeit abzunehmen.

Eigenschaften von Single-bus Multiprozessoren (1997)

Name	Maximale Anzahl an Prozessoren	Prozessor	Maximale Größe des Speichers / System	Kommunikations-Bandbreite / System
Compaq ProLiant 5000	4	Pentium Pro 200 MHz	2 GB	540 MB / s
Digital AlphaServer 8400	12	Alpha 21164 440 MHz	28 GB	2150 MB / s
HP 9000 K460	4	Pa-8000 180 MHz	4 GB	960 MB / s
IBM RS/6000 R40	8	PowerPC 604 112 MHz	2 GB	1800 MB / s
SGI Power Challenge	36	MIPS R10000 195 MHz	16 GB	1200 MB / s
Sun Enterprise 6000	30	UltraSPARC 1 167 MHz	30 GB	2600 MB / s

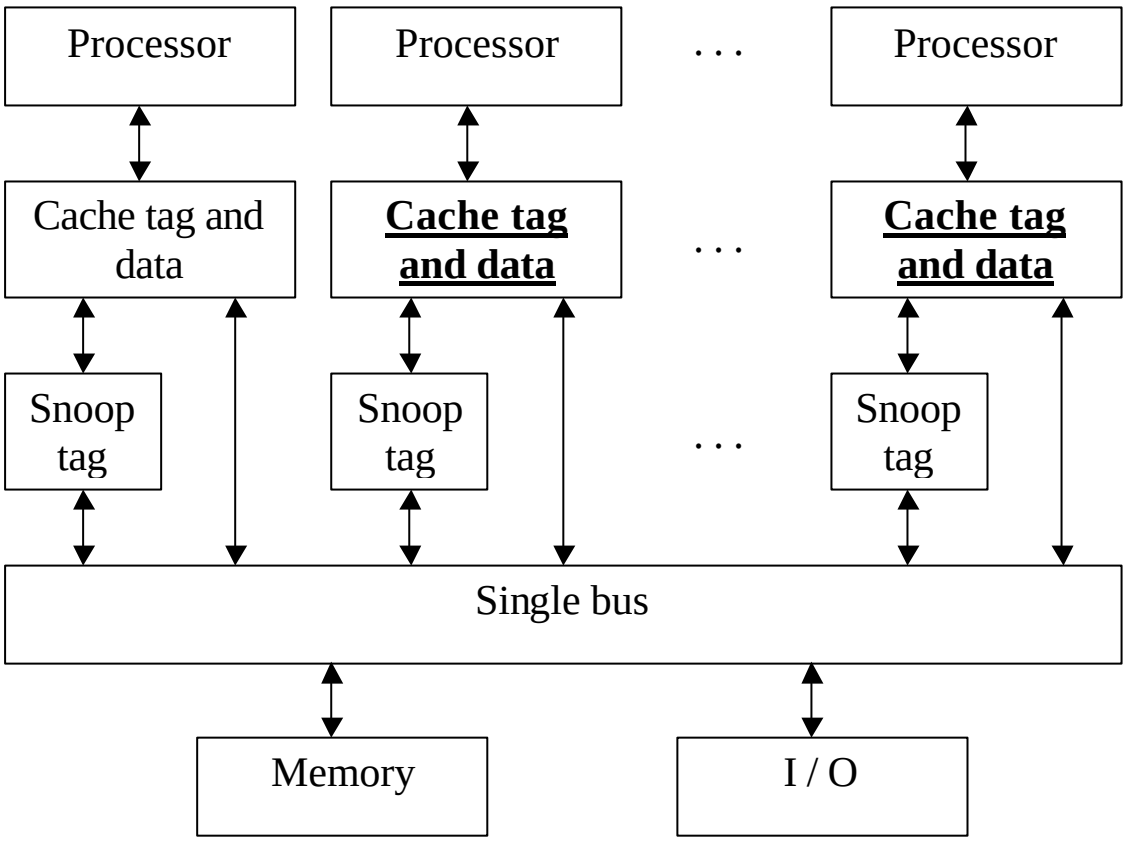
---

Cache Coherency bei Multiprozessoren

Problem : Wie kann sichergestellt werden, daß die Daten in den einzelnen Caches nicht veraltet sind?!

- Ein Prozessor könnte, z.B. durch eine Berechnung, einen neuen Wert für eine Variable erhalten und diesen in seinem Cache und dem Speicher speichern. In den anderen Caches würde der alte Wert gespeichert bleiben.

Cache Coherency durch Snooping



Zwei Arten von Snooping-Protokollen :

1. Write-invalidate :

- Der schreibende Prozessor erklärt die anderen Kopien der betroffenen Daten für ungültig und schreibt die neuen Daten dann in seinen Cache.

2. Write-update :

- Der schreibende Prozessor schreibt die neuen Daten über alle Kopien der betroffenen Daten.

- Normalerweise wird ein *write-invalidate* Protokoll in Verbindung mit *write-back* Cache verwendet, um den Bus zu entlasten.

Was passiert, wenn zwei Prozessoren im gleichen Taktzyklus auf den gleichen geteilten Speicherblock schreiben wollen?!

1. Einem der beiden Prozessoren wird zuerst der Bus zugeteilt.
2. Der erste Prozessor schreibt seine Daten.
3. Dem zweiten Prozessor wird der Bus zugeteilt.
4. Der zweite Prozessor schreibt seine Daten.

- Durch den Bus wird also ein sequentielles Verhalten beim Schreiben erzwungen. Nur so ist es möglich, daß das Schreiben mehrerer Prozessoren in den gleichen Speicherblock korrekt funktioniert.

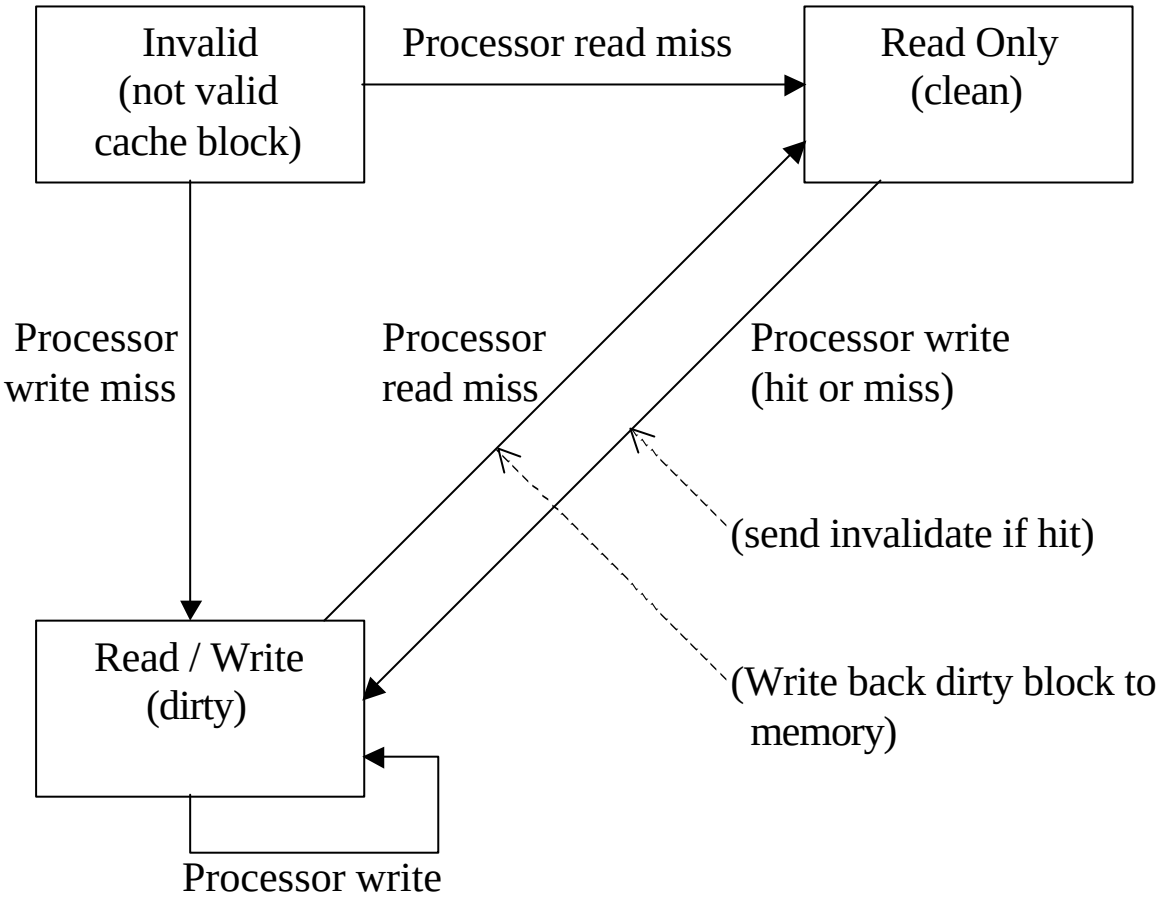
Beispiel für ein Cache Coherency Protokoll

- Jeder Cache-Block ist in einem von drei verschiedenen Zuständen :

  1. ***Read Only*** : Dieser Cache-Block ist nicht beschrieben (*clean*) und kann geteilt werden.
  2. ***Read / Write*** : Dieser Cache-Block ist beschrieben (*dirty*) und kann nicht geteilt werden.
  3. ***Invalid*** : Dieser Cache-Block enthält keine gültigen Daten.

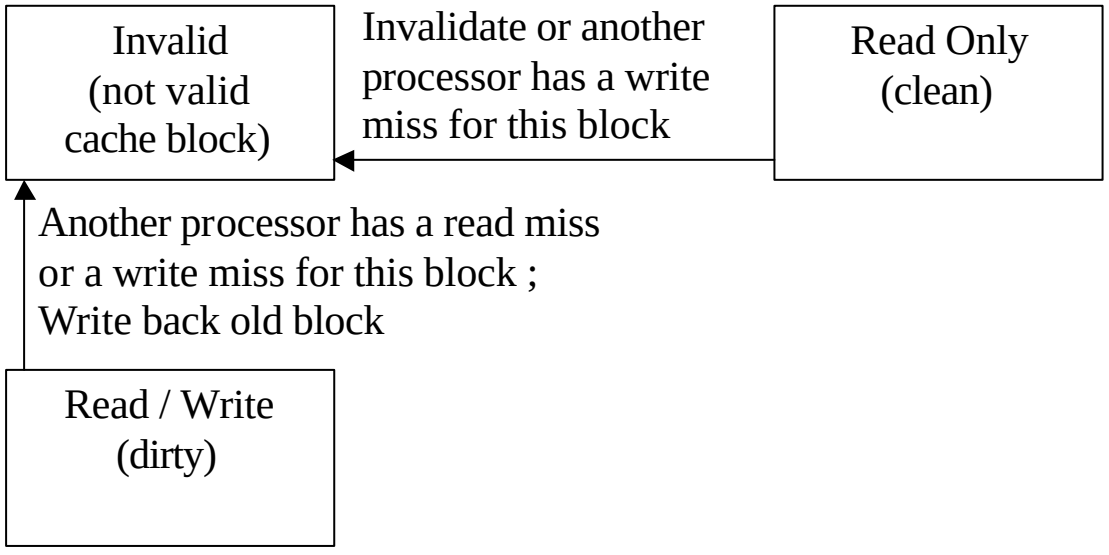
Beispiel für ein Cache Coherency Protokoll (Fortsetzung)

a) Cache-Status Veränderungen durch den zugehörigen Prozessor :



---

b) Cache-Status Veränderungen durch einen anderen Prozessor :



Beispiel für ein Cache Coherency Protokoll (Fortsetzung)

Wann kommt es zu Cache-Status-Veränderungen?

- Der Cache-Status wird verändert, wenn ein Leseversuch nicht funktioniert (*read miss*).
    - Der Cache-Status wird verändert, wenn ein Schreibversuch nicht funktioniert (*write miss*).
    - Der Cache-Status wird verändert, wenn ein Schreibversuch funktioniert (*write hit*).

Was passiert bei einem *read miss*?

1. Der Status des Cache-Blocks wird auf *Read Only* gesetzt.
2. Wenn der alte Cache-Block auf *Read / Write* gesetzt war, so schreibt der Cache den alten Block in den Speicher.
3. Alle Caches der anderen Prozessoren überprüfen, ob sie eine Kopie des betroffenen Cache-Blocks haben, der auf *Read / Write* gesetzt ist. Ist dies der Fall, so setzen sie ihn auf *Invalid* (oder *Read only*).

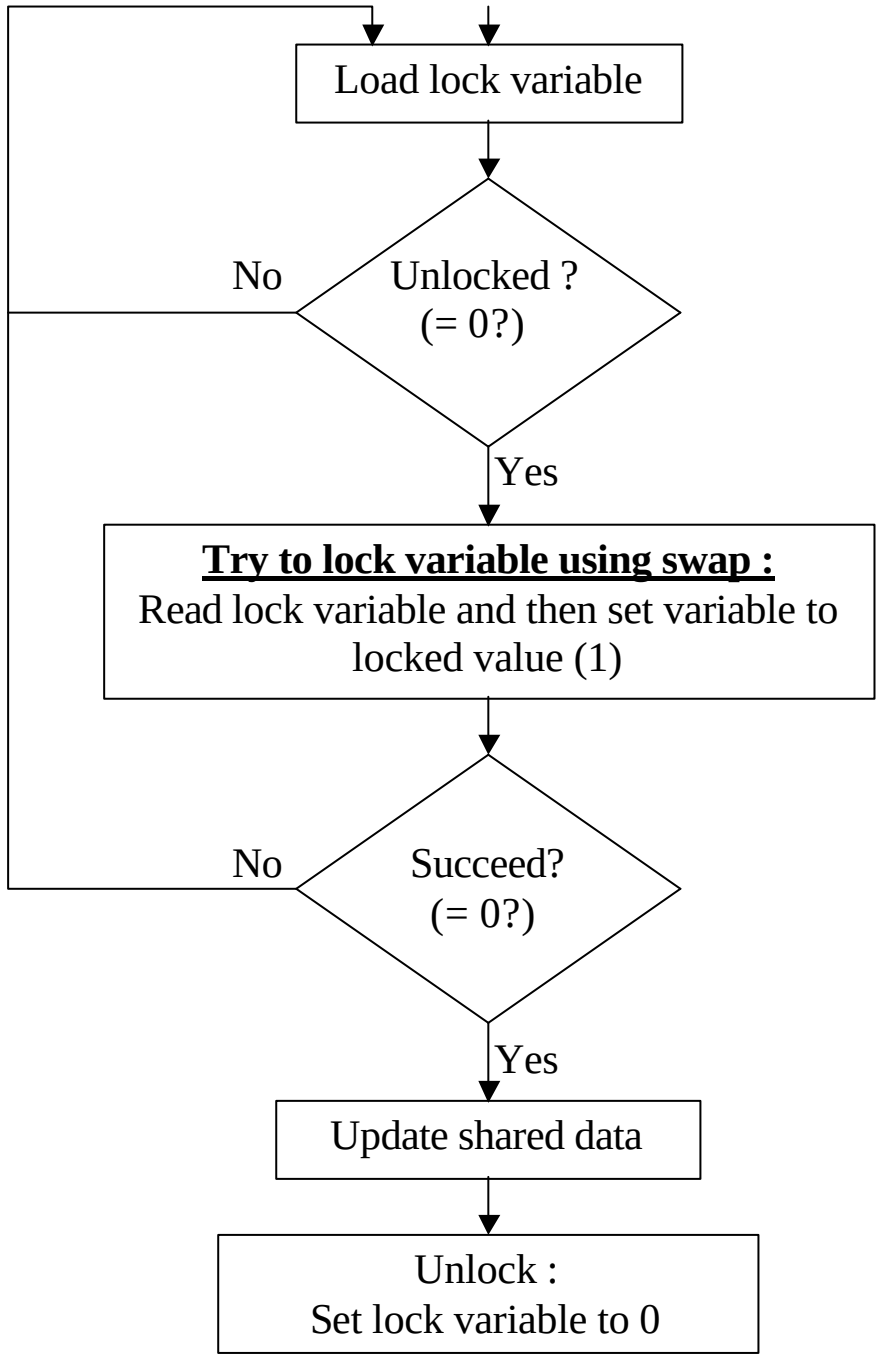
Was passiert, wenn der Prozessor auf einen Cache-Block schreiben will?

1. Der Prozessor sendet ein *Invalidate*-Signal über den Bus. Die Caches der anderen Prozessoren überprüfen, ob sie eine Kopie des Blocks haben und setzen diese, falls vorhanden, auf *Invalid*.
2. Der schreibende Prozessor schreibt in seinen Cache-Block und setzt ihn auf *Read / Write*.

Durch *Cache Coherency* synchronisieren

- Es müssen oft Prozesse miteinander koordiniert werden, die in einem Programm laufen.

- Synchronisierung der Prozesse wird normalerweise durch *lock variables* erreicht. Diese sorgen dafür, daß immer nur ein Prozeß auf die geteilten Daten zugreifen kann.



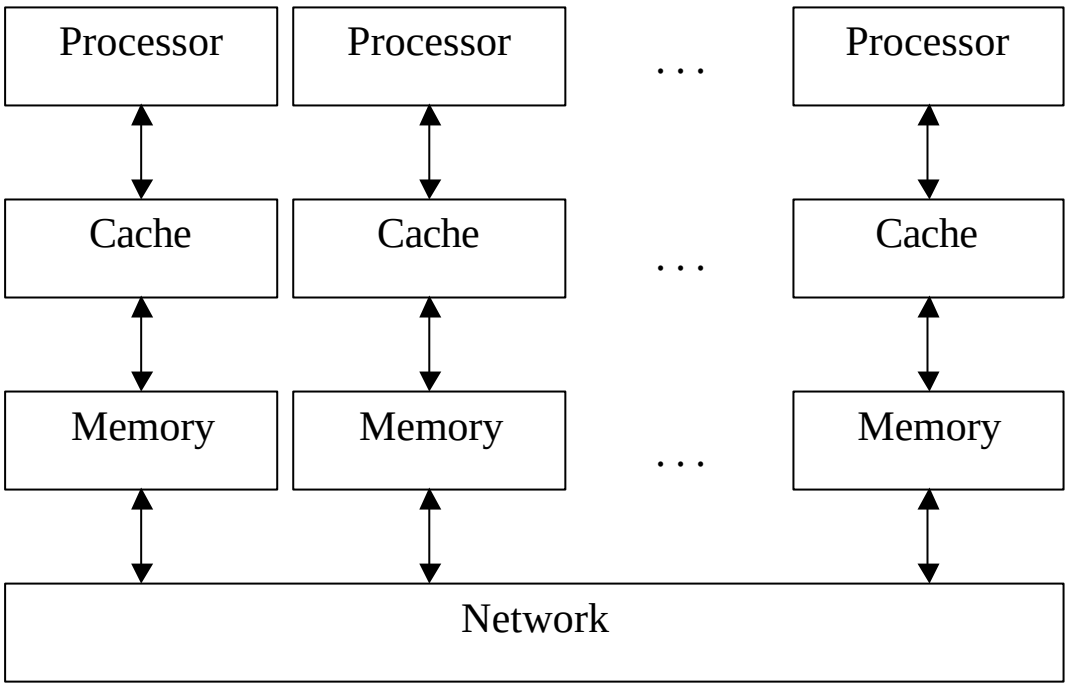
Durch *Cache Coherency* synchronisieren (Fortsetzung)

Step	Processor P0	Processor P1	Processor P2	Bus activity
1	Has lock	Spins, testing if lock = 0	Spins, testing if lock = 0	None
2	Sets lock to 0 and 0 sent over bus	Spins, testing if lock = 0	Spins, testing if lock = 0	Write-invalidate of lock variable from P0
3		Cache miss	Cache miss	Bus decides to service P2 cache miss
4		(Waits while bus busy)	Lock = 0	Cache miss for P2 satisfied
5		Lock = 0	Swap : reads lock and sets to 1	Cache miss for P1 satisfied
6		Swap : reads lock and sets to 1	Value from swap = 0 and 1 sent over bus	Write-invalidate of lock variable form P2
7		Value from swap = 1 and 1 sent over bus	Owns the lock, so can update shared data	Write-invalidate of lock variable from P1
8		Spins, testing if lock = 0		None

Vorteile und Nachteile

- *Spin waiting* entlastet den Bus.
- Der Bus wird stark belastet, wenn der *lock* freigegeben wird. Dies wirkt sich besonders stark aus, wenn der Multiprozessor aus vielen Prozessoren besteht.

Durch ein Netzwerk verbundene Multiprozessoren



Warum kein Single-bus Multiprozessor?

- Ein Bus kann nicht gleichzeitig eine hohe Bandbreite, eine lange Länge und eine niedrige Zugriffszeit bieten.

  >>> Es können nur wenige Prozessoren sinnvoll durch einen Bus verbunden werden.

- Bei einem Single-bus Multiprozessor wird der Bus für jeden Speicherzugriff usw. beansprucht. Bei einem Multiprozessor, der durch ein Netzwerk verbunden ist, wird das Netzwerk nur zur Kommunikation zwischen den Prozessoren beansprucht.

- Viele Prozessoren kann man gut durch ein Netzwerk miteinander verbinden.

Eigenschaften von Multiprozessoren, die durch ein Netzwerk verbunden sind (1997)

Name	Maximale Anzahl an Prozessoren	Prozessor	Größe des Speichers / System	Kommunikations-Bandbreite / Link
Cray Research T3E	2048	Alpha 21164 450 MHz	512 GB	1200 MB / s
HP /Convex Exemplar X-class	64	PA-8000 180 MHz	64 GB	980 MB / s
Sequent NUMA-Q	32	Pentium Pro 200 MHz	128 GB	1024 MB / s
SGI Origin2000	128	MIPS R10000 195 MHz	128 GB	800 MB / s
Sun Enterprise 10000	64	UltraSPARC 1 250 MHz	64 GB	1600 MB / s

---

Organisation des Speichers

- **Shared memory :** Ein Adress-Raum für den ganzen Speicher, Kommunikation mit *loads* und *stores*
- **Multiple private memories :** Jeder Prozessor hat seinen eigenen Adress-Raum, Kommunikation mit *sends* und *receives*
- **Centralized memory :** Speicher liegt physikalisch zusammen, jeder Prozessor hat die gleiche Zugriffszeit
- **Distributed memory :** Speicher ist physikalisch unterteilt, jeder Prozessor hat Speichermodule in seiner Nähe

loads und stores vs. sends und receives

- Die meisten großen Multiprozessoren haben *distributed memory*.
- Sends und receives sind auf Hardwareebene leichter zu realisieren, als loads und stores.
- Sends und receives machen es dem Programmierer leichter, die Kommunikation zwischen den Prozessoren zu optimieren.
- Loads und stores führen normalerweise zu weniger überflüssiger Kommunikation. Dies entlastet das Netzwerk.
    - Es ist effizienter, Daten anzufordern, wenn sie gebraucht werden, als Daten zu erhalten, die möglicherweise gebraucht werden könnten.

    >>> Systeme mit distributed shared memory (DSM)

- Es ist möglich, durch Software die Illusion eines gemeinsamen Adress-Raumes zu schaffen

    >>> shared virtual memory

Cache Coherency bei Multiprozessoren, die durch ein Netzwerk verbunden sind

- Bus-snooping Protokolle funktionieren nicht, da die Prozessoren nicht durch einen gemeinsamen Bus miteinander verbunden sind.
- Eine Alternative zu bus-snooping sind Verzeichnisse (*directories*).

Verzeichnisse (*directories*)

- Protokolle, die auf Verzeichnissen aufbauen, haben ein Verzeichnis, daß den Status jedes Speicherblocks des Hauptspeichers enthält.

- Information in einem Verzeichnis :

  - Welche Caches haben Kopien des Speicherblocks

  - Ist der Block *dirty* / *clean*

  - ...

- Verzeichnisse können auf verschiedenen Speichern vervielfältigt auftreten.

  >>> Dies verringert die Anzahl der Zugriffe auf einen Speicher.

Negative Auswirkungen bei einheitlichem Adress-Raum und Verzeichnissen

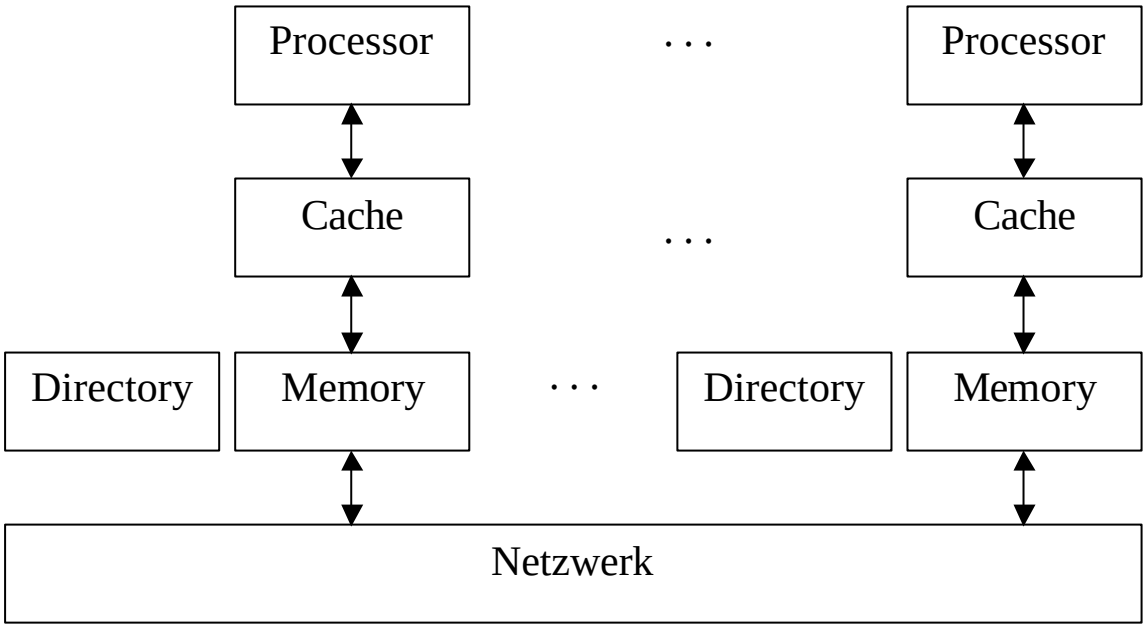
- Verzeichnisse können über verschiedene Speicher verteilt sein.

  >>> Zugriffe auf das Verzeichnis dauert lange, da über das Netzwerk zugegriffen werden muß.

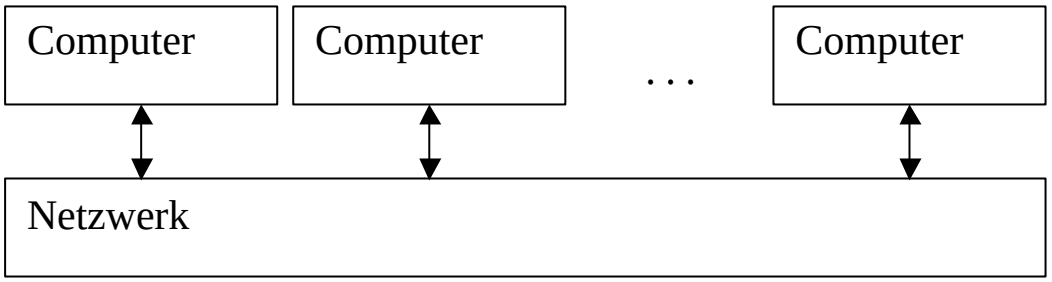
  >>> Das Netzwerk wird durch Zugriffe auf das Verzeichnis stark belastet.

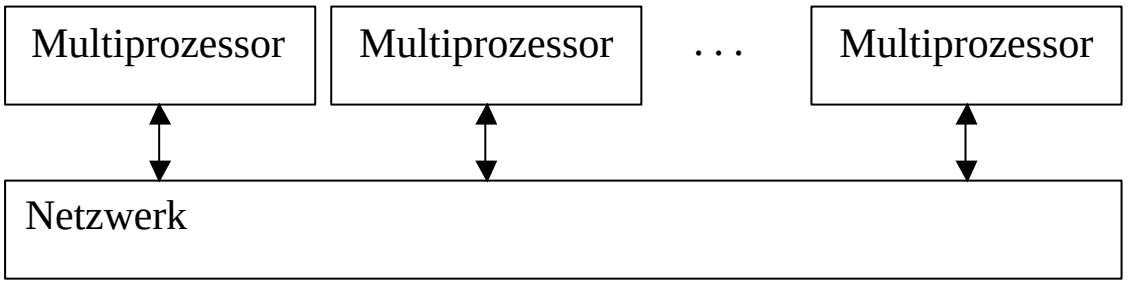
  - Diese negativen Auswirkungen führen nicht bei allen Programmen zu deutlichen Leistungseinbrüchen und können durch gute Programmierung vermindert werden.

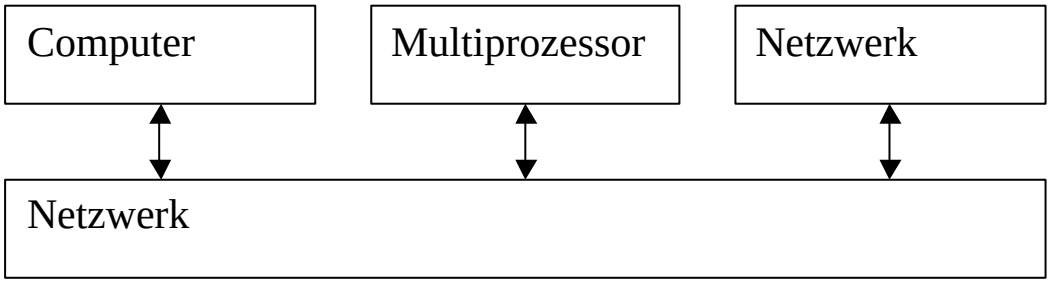
Durch ein Netzwerk verbundene Multiprozessoren mit Verzeichnissen



Cluster


Computer
Computer
. . .
Computer
Netzwerk


Multiprozessor
Multiprozessor
. . .
Multiprozessor
Netzwerk


Computer
Multiprozessor
Netzwerk
Netzwerk

Vorteile von Clustern

- Der Ausfall eines Systems hat nur geringe Auswirkungen auf den gesamten Cluster.

  >>> Cluster sind als Systeme geeignet, die immer und zu jeder Zeit funktionieren müssen.

- Es können sehr viele einzelne Systeme als Cluster sinnvoll miteinander verbunden werden.

- Cluster können leicht mit zusätzlichen Systemen erweitert werden.

- Große Cluster sind billiger, als große Multiprozessoren.

Nachteile von Clustern

- Cluster sind meistens durch den I /O Bus der einzelnen Systeme vernetzt. Dieser Bus hat nur eine sehr geringe Bandbreite.

- Die Betreuung eines Clusters kostet sehr viel mehr, als die Betreuung eines einzelnen Multiprozessors.

- Jedes System eines Clusters hat seinen eigenen Speicher. Ein Programm hat also nicht den gesamten Speicher des Clusters zur Verfügung, sondern nur den Speicher eines Systems.

Nutzung der Vorteile von Clustern und Multiprozessoren

- Cluster, die aus kleineren SMP’s bestehen weisen Vorteile von Clustern und Multiprozessoren auf.

Eigenschaften von Clustern (1997)

Name	Maximale Anzahl an Prozessoren	Prozessor	Maximale Größe des Speichers / System	Kommunikations-Bandbreite / Link
HP 9000 EPS21	64	PA-8000 180 MHz	64 GB	532 MB / s
IBM RS/6000 HACMP R40	16	PowerPC 604 112 MHz	4 GB	12 MB / s
IBM RS/6000 SP2	512	Power2 SC 135 MHz	1024 GB	150 MB / s
Sun Enterprise Cluster 6000 HA	60	UltraSPARC 167 MHz	60 GB	100 MB / s
Tandem NonStop Himalaya S70000	4096	MIPS R10000 195 MHz	1024 GB	40 MB / s

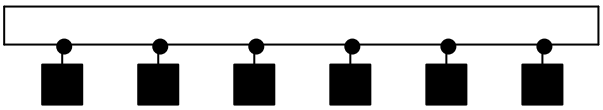
- Alle hier aufgelisteten Cluster, bis auf den IBM RS/6000 SP2, bestehen aus SMP’s.

Netzwerke

- Wie kann man Prozessor-Speicher Einheiten miteinander verbinden?

  - ***Fully connected network*** : Jede Einheit hat zu jeder anderen Einheit eine direkte Verbindung.

  - ***Ring*** : Jeder Prozessor-Speicher Knoten ist mit zwei anderen verbunden.



Ringe vs. vollkommen verbundene Netzwerke

- Vollkommen verbundene Netzwerke sind sehr teuer und sind ab einer gewissen Größe nicht mehr realisierbar.

- Ringe sind billig und haben keine Größenbegrenzung.

- Vollkommen verbundene Netzwerke bieten sehr viel mehr Leistung, als Ringe.

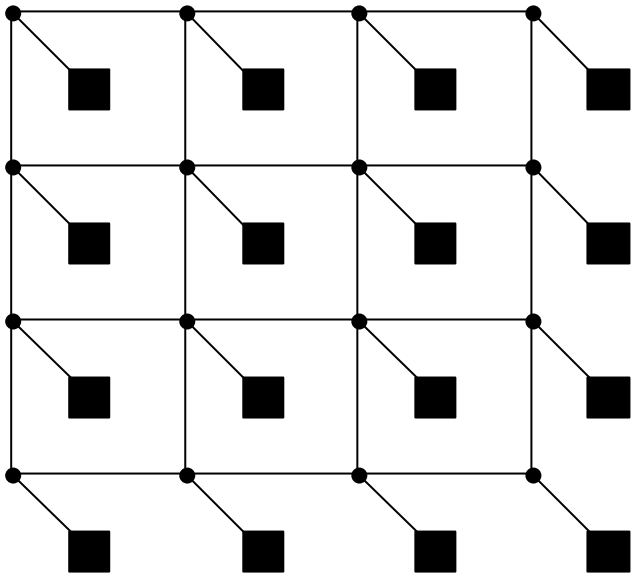
- Es werden meistens Netzwerke gebaut, die geringere Kosten, als ein vollkommen vernetztes Netzwerk haben, aber eine höhere Leistung, als ein Ring.

Alternativen

- Gitter (*2-D grids*)

- *n-cubes*

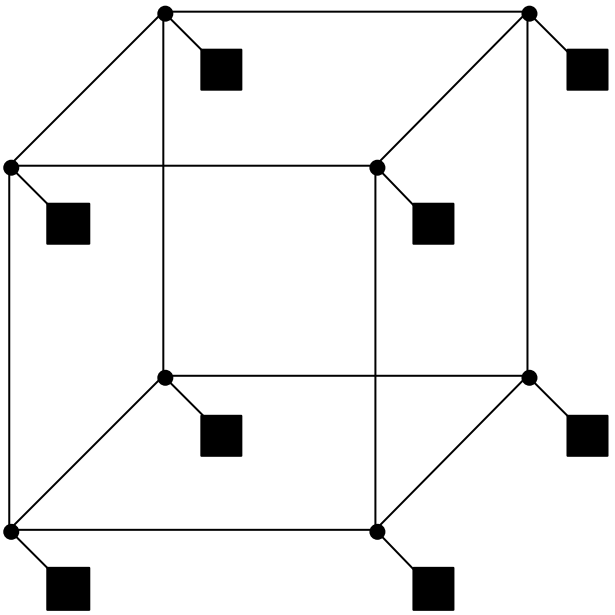
2-D grid



- 2-D Gitter mit 16 Knoten

---

n-cube



- n-cube mit 8 Knoten ($8 = 2^3 \rightarrow n = 3$)

Leistung bei Netzwerken

- ***total network bandwidth*** : Die Bandbreiten aller Verbindungen im Netzwerk werden addiert.

- ***bisection bandwidth*** : Das Netzwerk wird in zwei Teile geteilt, wobei jeder Teil die Hälfte der gesamten Knoten enthält. Leistung ergibt sich als Summe der Bandbreiten der Verbindungen, die von einem Teil in den anderen verlaufen.

    - Das Netzwerk muß immer an der ungünstigsten Stelle geteilt werden, da die geringste Bandbreite meist die Gesamtleistung des Netzwerks bestimmt.

Weitere Alternativen

- Man muß nicht an jeden Knoten einen Prozessor anschließen. Der Knoten kann durch ein *switch* gebildet werden.

    - Ein *switch* ist kleiner, als eine Einheit aus Prozessor, Speicher und *switch*.

    >>> *Crossbar*

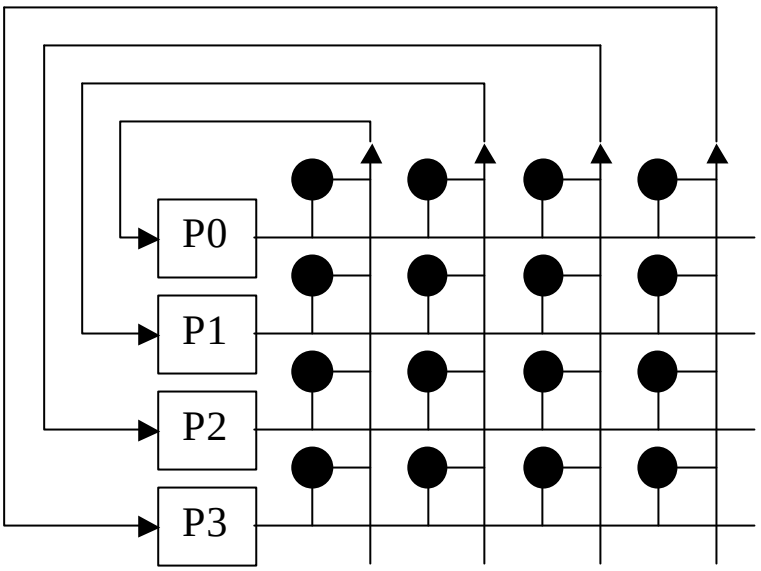
    >>> *Omega network*

Probleme bei Netzwerken

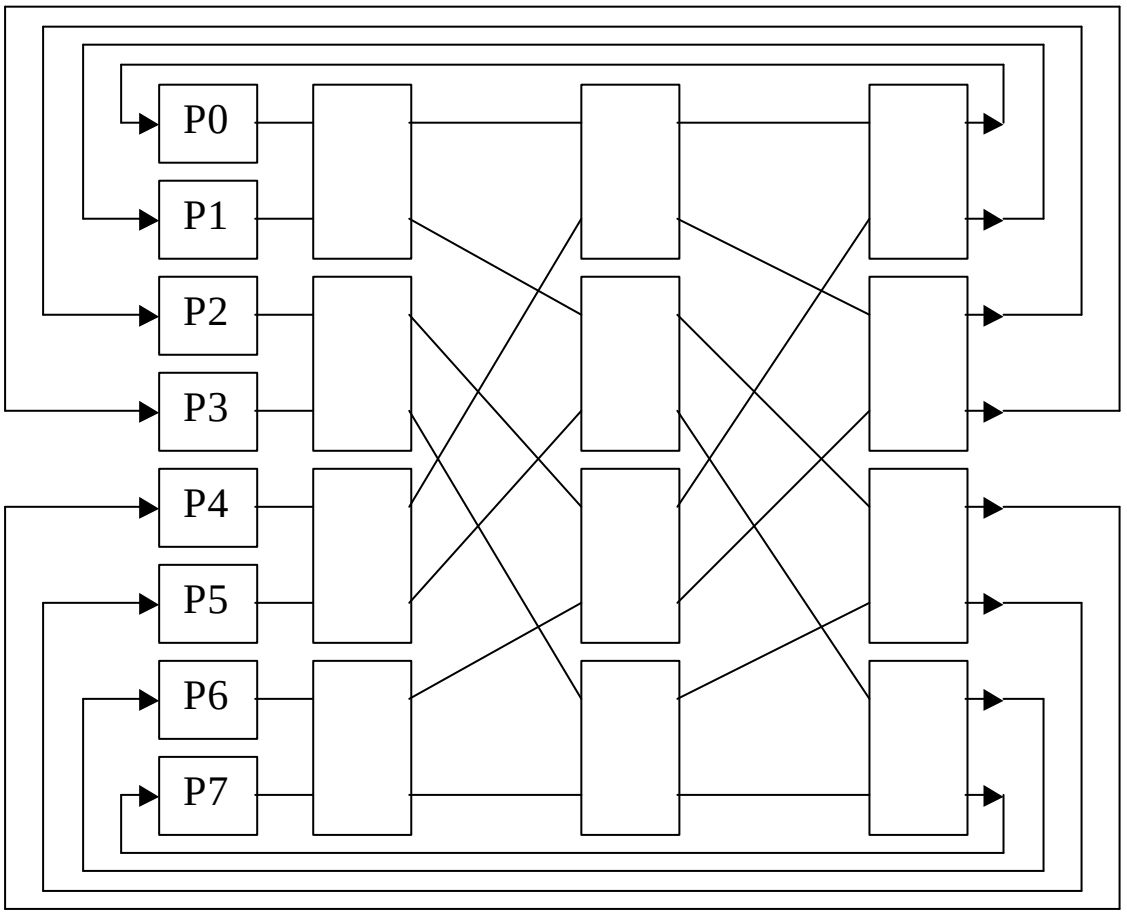
- Lange Verbindungen können nur durch großen Aufwand mit einer hohen Taktrate laufen.

- Netzwerke, die auf dem Papier übersichtlich und elegant aussehen, sind nicht automatisch leicht zu realisieren.

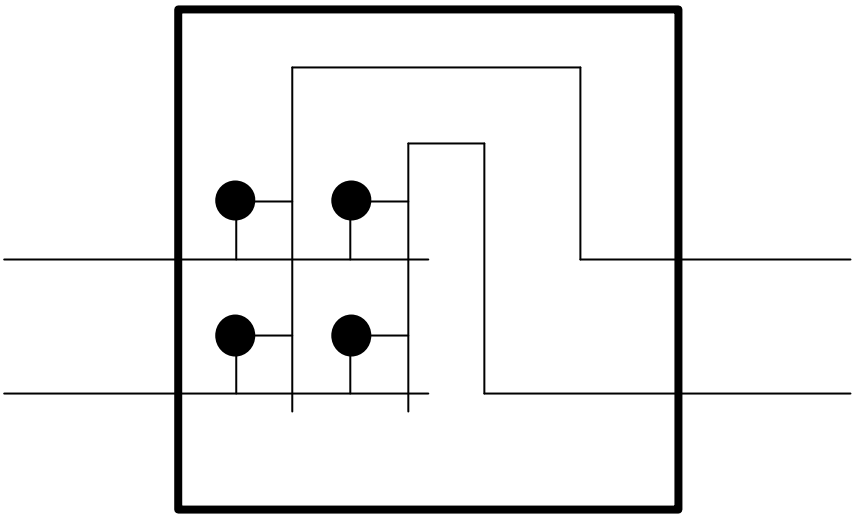
Crossbar



Omega network



Omega network switch box



Trugschluß : Praktisch erreichbare Leistung ist nahe der Spitzenleistung

- 1. Definition : Spitzenleistung = Spitzenleistung eines Prozessors * Anzahl der Prozessoren

- 2. Definition : Spitzenleistung ist die Leistung, die garantiert nie erreicht wird.

Name	Spitzenleistung in MFLOPS	Leistung bei 3D FFT PDE in MFLOPS
Cray YMP (8 Prozessoren)	2666	1795 (67 %)
IBM SP2 (64 Prozessoren)	14636	1093 (7 %)

Trugschluß : Amdahl's Gesetz gilt nicht für Multiprozessoren

- Ein Programm, daß auf einem Multiprozessor mit 1000 Prozessoren in der gleichen Zeit das tausendfache von dem berechnet, was das Programm auf einem Prozessor berechnet, ist nicht automatisch 1000 mal so schnell.

- Programme mit sequentiellen Programmteilen können nicht auf n Prozessoren n-mal schneller sein.