Parallelrechner: Motivation

immer höhere Performance gefordert

=> schnellere Einzelprozessoren
aber Takte oberhalb von 10 GHz unrealistisch

=> mehrere Prozessoren

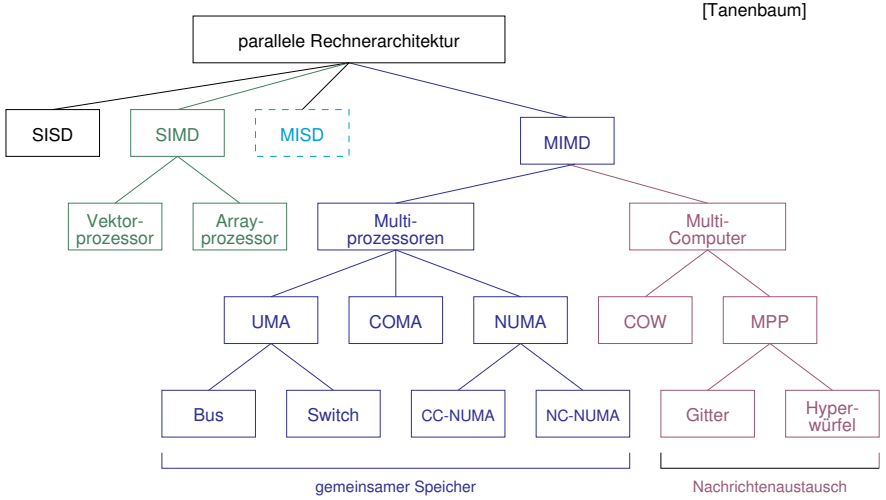
- diverse Architekturkonzepte
- shared-memory vs. message-passing
- Overhead durch Kommunikation
- Programmierung ist ungelöstes Problem

derzeit beliebtester Kompromiss:

- bus-basierte SMPs mit 2-16 Prozessoren

Parallelrechner: Klassifikation

[Tanenbaum]



Parallelrechner: Literatur

Tanenbaum, Computerarchitektur (4. Auflage)
Hennessy & Patterson, computer architecture

Messmer, PC-Hardwarebuch
Intel Pentium Manual

Intel ITJ (ASCI red)
diverse c´t-Artikel, insbesondere Benchmarks

Parallelrechner:

- Programmierung ist ungelöstes Problem
- Aufteilung der Programme auf CPUs/nodes?
- insbesondere bei komplexen Kommunikationsnetzwerken

- Parallelität typischer Programme (gcc, spice, ...): kleiner 8
- massiv parallele Rechner sind dann Verschwendung
- aber SMP-Lösungen mit 4..16 Prozessoren attraktiv

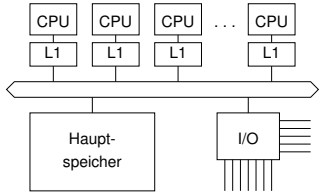
- Datenbankanwendungen oft gut parallelisierbar
- z.B. je ein Thread/Prozeß pro Anfrage
- Vektor/Feld-Rechner für Numerik, Simulation

- Supercomputer derzeit nur für Numerik / Militär
- ansonsten "kleine" SMP-basierte Rechner

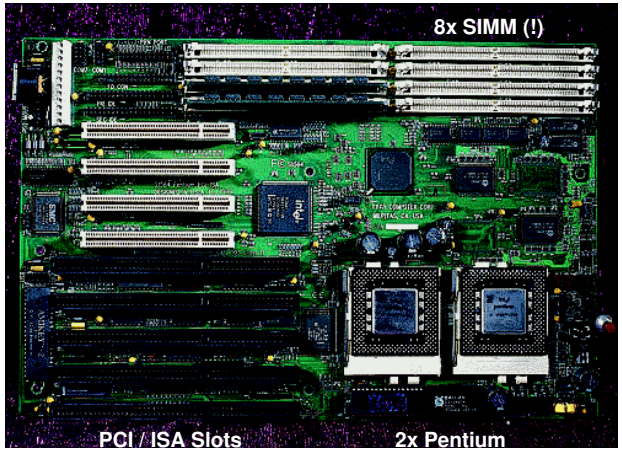
SMP: "Symmetric multiprocessing"

- mehrere Prozessoren teilen gemeinsamen Hauptspeicher
- Zugriff über Verbindungsnetzwerk oder Bus
- geringer Kommunikationsoverhead

- bus-basierte Systeme sind sehr kostengünstig
- aber schlecht skalierbar (Bus wird Flaschenhals)
- lokale Caches für gute Performance notwendig
- MESI-Protokoll und Snooping für Cache-Kohärenz



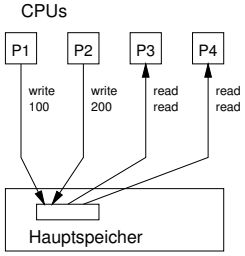
SMP: dual Pentium-Board (1998)



SMP: Eigenschaften ...

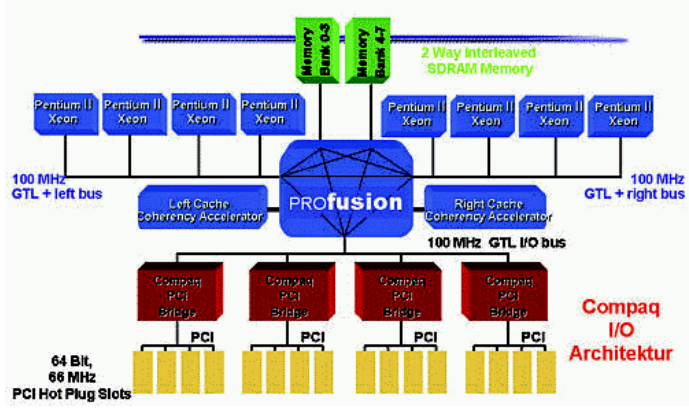
"symmetric multiprocessing":

- alle CPUs gleichrangig, Zugriff auf Speicher und I/O
- gleichzeitiger Zugriff auf eine Speicheradresse?
- strikte / sequentielle / Prozessor- / schwache Konsistenz



W1 100	W1 100	W2 200
W2 200	R3 = 100	R4 = 200
R3 = 200	W2 200	W1 100
R3 = 200	R3 = 200	R3 = 100
R4 = 200	R4 = 200	R4 = 100
R4 = 200	R4 = 200	R4 = 100

SMP: Pentium II (Compaq Profusion)



SMP: Cache-Kohärenz

aus Performancegründen:

- jeder Prozessor hat seinen eigenen Cache (L1, L2, ...)
- aber gemeinsamer Hauptspeicher

=> Problem: "Cache-Kohärenz"

Prozessor X greift auf Daten zu, die im Cache von Y liegen

1) Lesezugriff von X: Y muß seinen Wert liefern
2) Schreibzugriff von X: Y muß Wert von X übernehmen
3) was soll bei gleichzeitigem Zugriff passieren?!
   (vgl. Java synchronized Konzept)

=> MESI-Protokoll mit Snooping

- Caches enthalten Wert, Tag, und 2-bit MESI-Zustand

SMP: MESI Konzept

MESI := modified, exclusive, shared, invalid

- jede Cache-Speicherstelle wird um 2 Statusbits erweitert
- alle Prozessoren überwachen die Zugriffe anderer Prozessoren
- entsprechende Aktualisierung der Statusbits

Zustand:	Bedeutung (grob):
invalid	Wert ist ungültig (z.B. noch nie geladen)
exclusive	gültiger Wert, nur in diesem Cache vorhanden
modified	gültiger Wert, nur in diesem Cache vorhanden, gegenüber Hauptspeicher-Wert verändert
shared	gültiger Wert, in mehreren Caches vorhanden

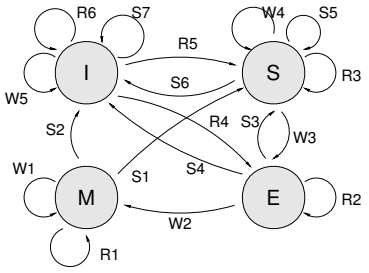
SMP: MESI Zustände

MESI-Zustand	Cache-Eintrag gültig?	Wert im Speicher gültig?	Kopien in anderen Caches?	Zugriff betrifft
M	ja	nein	nein	Cache
E	ja	ja	nein	Cache
S	ja	ja	möglich	Speicher
I	nein	unbekannt	möglich	Speicher

- Cache-Strategie: write-back, kein write-allocate
- Schreibzugriffe auf M führen nicht zu Bus-Transaktionen
- Werte in E stimmen mit Hauptspeicherwerten überein
- Werte in S sind aktuell, Lesezugriff ohne Bus-Transaktion
- Schreibzugriff auf S: lokal S, fremde auf I, Wert abspeichern

- mit write-through Caches: Zustände S/I, kein M/E

SMP: MESI Übergänge



Snoop-Zyklen

M-S	S1	Hit, Speicher schreiben
M-I	S2	Hit, Speicher schreiben
E-S	S3	Hit, aber nicht modifiziert

usw.

Lesezugriffe:

M-M	R1	Cache-Hit, CPU bekommt Daten
E-E	R2	Cache-Hit, CPU bekommt Daten
S-S	R3	Cache-Hit, CPU bekommt Daten
I-E	R4	Miss, Speicher liefert Daten
I-S	R5	Miss, externer Cache liefert Daten
I-I	R6	Miss, Adresse nicht cacheable

Schreibzugriffe:

M-M	W1	Hit, CPU aktualisiert Cache
E-M	W2	Hit, CPU aktualisiert Cache
S-E	W3	Hit (write-back): Cache aktualisiert, Buszyklus markiert fremde Kopien als invalid
S-S	W4	Hit (write-through): Caches und Speicher aktualisiert
I-I	W5	Miss, Speicher schreiben, aber kein write-allocate

SMP: MESI Snooping

Snooping := "schnüffeln"

- alle Prozessoren überwachen alle Bus-Transaktionen
- Zugriffe auf "modified"-Werte werden erkannt:

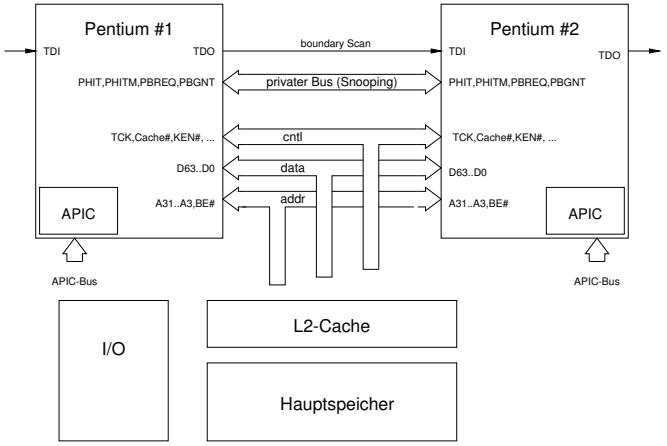
  1. fremde Bus-Transaktion unterbrechen
  2. eigenen (=modified) Wert zurückschreiben
  3. Status auf shared ändern
  4. unterbrochene Bus-Transaktion neu starten

- erfordert spezielle Snoop-Logik im Prozessor
- garantiert Cache-Kohärenz aller Prozessoren
- optimale Performance

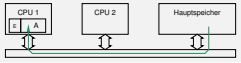
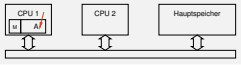
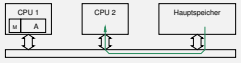
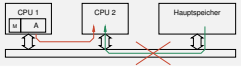
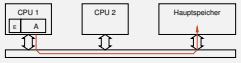
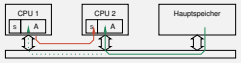
Beispiel: siehe nächste Folie

[PC-Hardwarebuch]

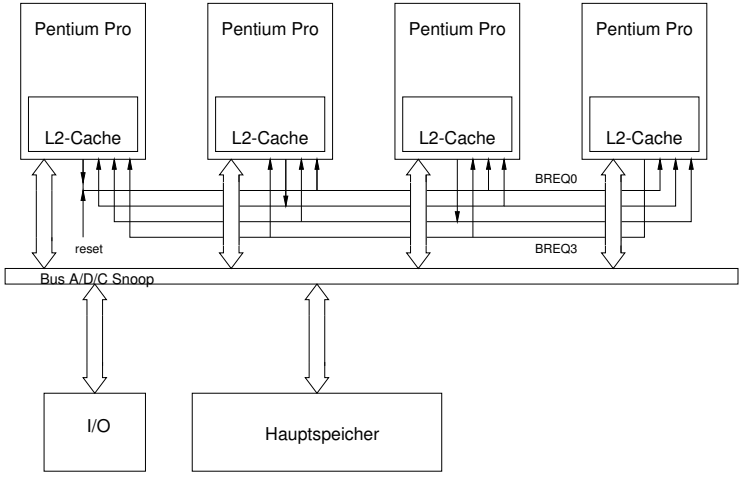
SMP: Pentium



SMP: MESI Snooping: Beispiel

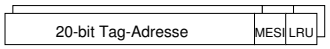
	MESI-Status Wert A:	CPU1	CPU2
	CPU 1 lädt Wert A	E	I
	CPU 1 modifiziert A	M	I
	CPU 2 lädt A (aber Wert modified)	-	-
	CPU 1 SNOOP! (CPU2 read gestoppt)	- -	- -
	CPU 1 schreibt A (CPU2 read gestoppt)	E	I
	CPU 2 lädt A (CPU2 read restart, A shared)	S	S

SMP: Pentium Pro



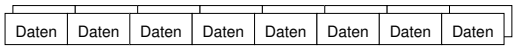
SMP: MESI Pentium

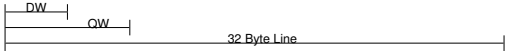
D-Cache Tags:



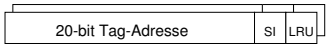
2 MESI Bits
1 LRU Bit

Daten-Cache Werte:

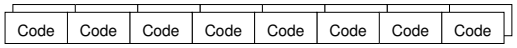




I-Cache Tags:



1 SI Bit
1 LRU Bit



- 32-Byte Cache-Lines
- D-Cache untersützt MESI, I-Cache nur SI
- externe Signale zeigen MESI-Übergänge an [PC-Hardwarebuch]

SMP: Interrupts

spezielle Interrupt-Behandlung in SMP-Rechner notwendig:

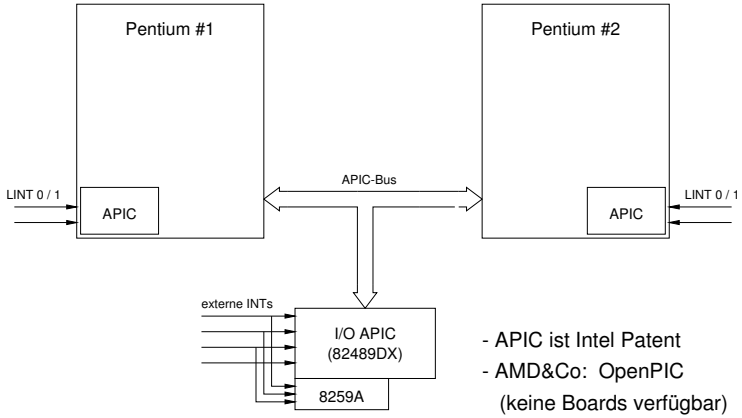
- welcher Prozessor soll einen Interrupt bearbeiten?

- statisch, z.B. immer der erste Prozessor
- der am wenigsten ausgelastete
- round-robin, oder ähnliche Strategien

Interrupt-Maskierung

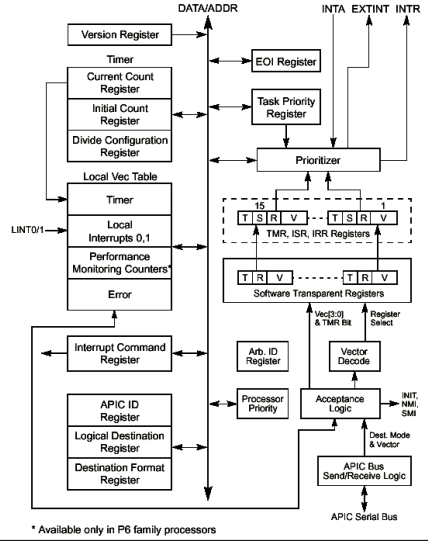
- externe Folie

SMP: Pentium APIC



- APIC ist Intel Patent
- AMD&Co: OpenPIC
  (keine Boards verfügbar)

SMP: Pentium APIC



x86: locked atomic operations

7.1. LOCKED ATOMIC OPERATIONS

The 32-bit Intel Architecture processors support locked atomic operations on locations in system memory. These operations are typically used to manage shared data structures (such as semaphores, segment descriptors, system segments, or page tables) in which two or more processors may try simultaneously to modify the same field or flag. The processor uses three interdependent mechanisms for carrying out locked atomic operations:

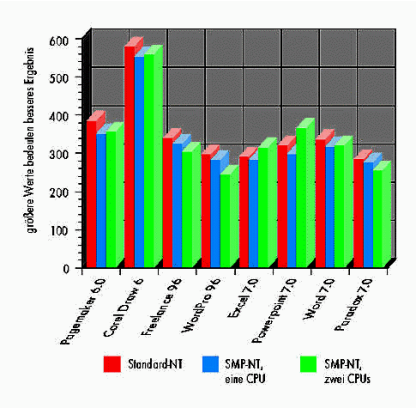
- Guaranteed atomic operations.
- Bus locking, using the LOCK# signal and the LOCK instruction prefix.
- Cache coherency protocols that insure that atomic operations can be carried out on cached data structures (cache lock). This mechanism is present in the P6 family processors.

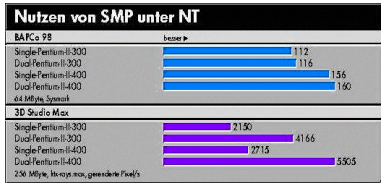
These mechanisms are interdependent in the following ways. Certain basic memory transactions (such as reading or writing a byte in system memory) are always guaranteed to be handled atomically. That is, once started, the processor guarantees that the operation will be completed before another processor or bus agent is allowed access to the memory location. The processor also supports bus locking for performing selected memory operations (such as a read-modify-write operation in a shared area of memory) that typically need to be handled atomically, but are not automatically handled this way. Because frequently used memory locations are often cached in a processor's L1 or L2 caches, atomic operations can often be carried out inside a processor's caches without asserting the bus lock. Here the processor's cache coherency protocols insure that other processors that are caching the same memory locations are managed properly while atomic operations are performed on cached memory locations.

Note that the mechanisms for handling locked atomic operations have evolved as the complexity of Intel Architecture processors has evolved. As such, more recent Intel Architecture processors (such as the P6 family processors) provide a more refined locking mechanism than earlier Intel Architecture processors, as is described in the following sections.

- notwendig für Multiprozessorsysteme

SMP: Windows NT Benchmarks





- fast kein Gewinn für die BAPCo
- Verwaltungsoverhead ~ 10%
- 3D-Studio Max doppelte Perf.

SMP: x86 Memory Type Range Registers

Table 9-6. MTRR Memory Types and Their Properties

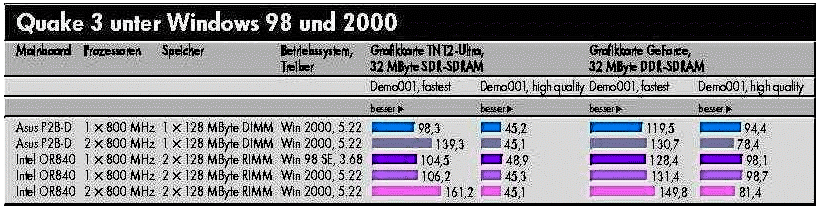
Mnemonic	Encoding in MTRR	Cacheable in L1 and L2 Caches	Writeback Cacheable	Allows Speculative Reads	Memory Ordering Model
Uncacheable (UC)	0	No	No	No	Strong Ordering
Write Combining (WC)	1	No	No	Yes	Weak Ordering
Write-through (WT)	4	Yes	No	Yes	Speculative Processor Ordering
Write-protected (WP)	5	Yes for reads, no for writes	No	Yes	Speculative Processor Ordering
Writeback (WB)	6	Yes	Yes	Yes	Speculative Processor Ordering
Reserved Encodings*	2, 3, 7 through 255				

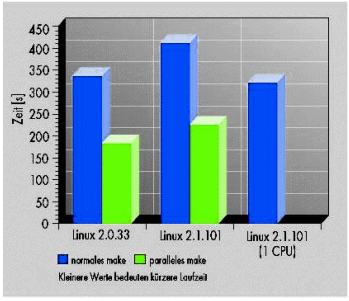
NOTE:

* Using these encoding result in a general-protection exception (#GP) being generated.

- Register (Pentium+) zur Einstellung des Cache-Verhaltens
- Vorsicht mit aggressiven Optimierungen. . .

SMP: Quake3, Linux make





- Nutzen nur für geeignete Apps.
- evtl. seltsame Effekte (Quake)
- beträchtlicher OS-Overhead (in Win2K, Linux 2.4 besser)
- gut für Server-Aufgaben siehe Compaq "Piranha"

ASCI: Motivation

"Accelerated Strategic Computing Initiative", DOE seit ~1996

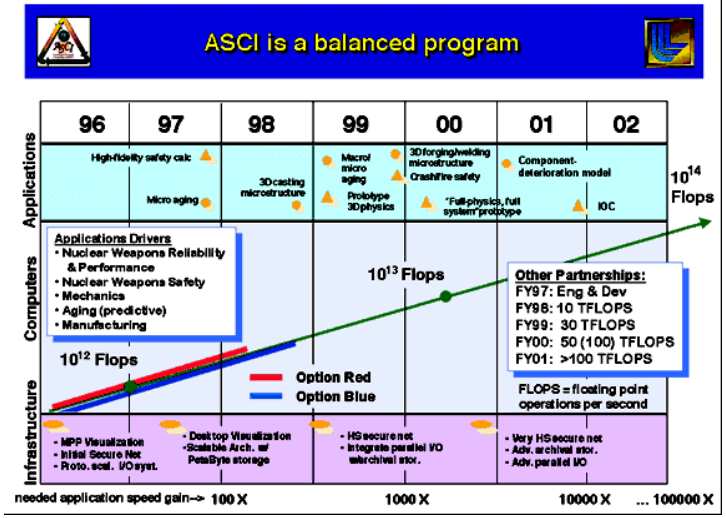
- Überalterung der Kernwaffenbestände
- Simulation notwendig wegen Teststopp-Verträgen ...
- und außerdem die "grand challenge" Anwendungen (QM, Wettervorhersage, finite-elements, ...)

=> Realisierung mehrerer Prototypen-Rechner für 1 TFlop
=> Bau eines 100 TFlops Rechners bis ca. 2002

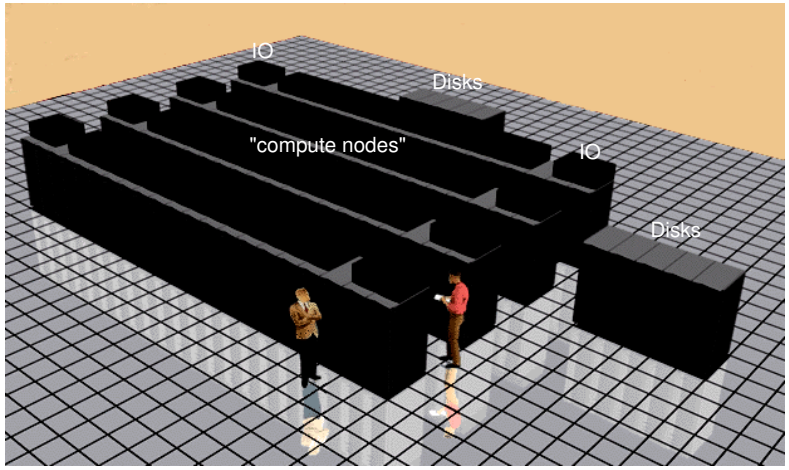
- "option red" Intel, Sandia NL
  9400 Prozessoren (PentiumPro/200), PC-Standardkomponenten
- "pacific blue" IBM, LLNL
- "mountain blue" SGI, Los Alamos NL

[www.sandia.gov/ASCI/]

ASCI: Roadmap



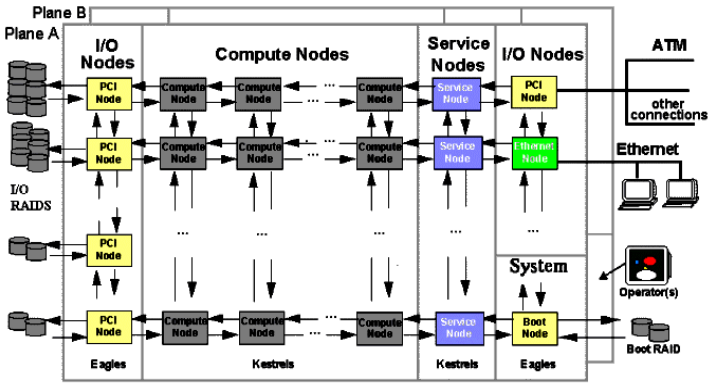
ASCI red: (Intel 1997)



9216 P6-CPUs 594 GB RAM 1 TB Disk 1.0GB/s I/O 1.8 TFLOPS

ASCI red: Photo

ASCI red: Architektur



Topologie: Netz mit
38 * 32 * 2 compute nodes

Figure 5: Logical System Block Diagram for the ASCI Option Red Supercomputer. This system uses a split-plane mesh topology and has 4 partitions: System, Service, I/O and Compute. Two different kinds of node boards are used and described in the text: the Eagle node and the Kestrel node. The operators console (the SPS station) is connected to an independent ethernet network that ties together patch support boards on each card cage.

ASCI red: I/O-Node

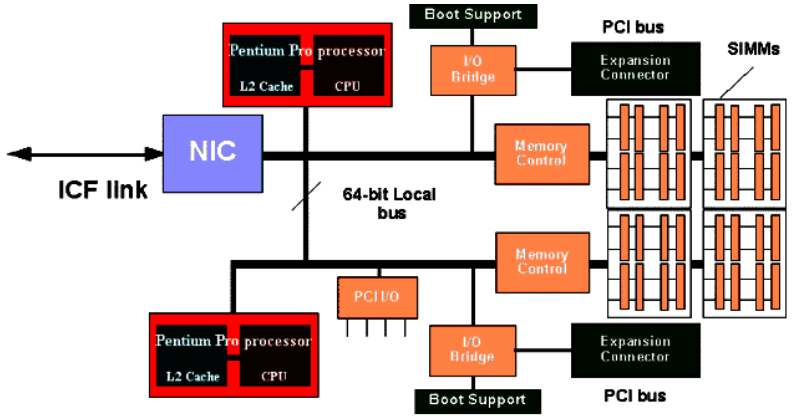
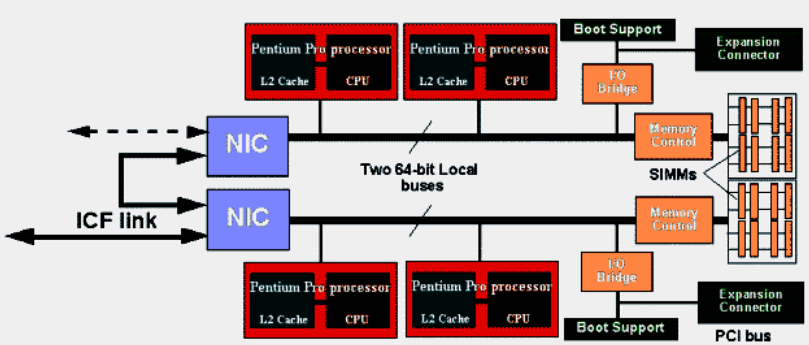


Figure 2: The ASCI Option Red Supercomputer I/O and system Node (Eagle Board). The NIC connects to the MRC on the backplane through the ICF Link.

ASCI red: "compute node"



"Kestrel"-Board:
2 Nodes mit je 2 PentiumPro/200, i82453 Chipsatz, 128 MB / node

Figure 3: The ASCI Option Red supercomputer Kestrel Board. This board includes two compute nodes chained together through their NIC's. One of the NIC's connects to the MRC on the backplane through the ICF Link.

ASCI red: "interconnection node"

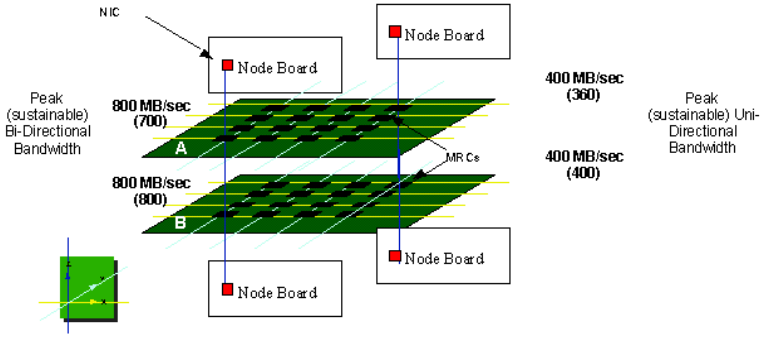


Figure 4: ASCI Option Red Supercomputer 2 Plane Interconnection Facility (ICF). Bandwidth figures are given for NIC-MRC and MRC-MRC communication. Bi-directional bandwidths are given on the left side of the figure while directional bandwidths are given on the right side. In both cases, sustainable (as opposed to peak) numbers are given in parentheses.

ASCI red: Performance

- 200 MHZ PentiumPro: 200 MFLOPs peak
- 9200x: 1.8 TFLOPs peak

- Weltrekord am 07.12.1996: 1 TFLOP erreicht
  - handoptimierter Assemblercode
  - handoptimierter Algorithmus (LRU blocked, pivoting)
  - Maschine 80% vollständig => ca. 140 MFLOPs/node
  - 75% der Maximalleistung erreicht (!)

- Speicherlimitierte Programme < 20 MFLOPs / node
- Compilierte Programme 20 .. 80 MFLOPs / node

- 640 Disks, 1540 Netzteile, 616 ICF-Backplanes . . .
- MTBF > 50 hours (bzw. 97% nodes aktiv für > 4 Wochen)

[Intel ITJ Q1/98]

Leerseite

Leerseite

Leerseite