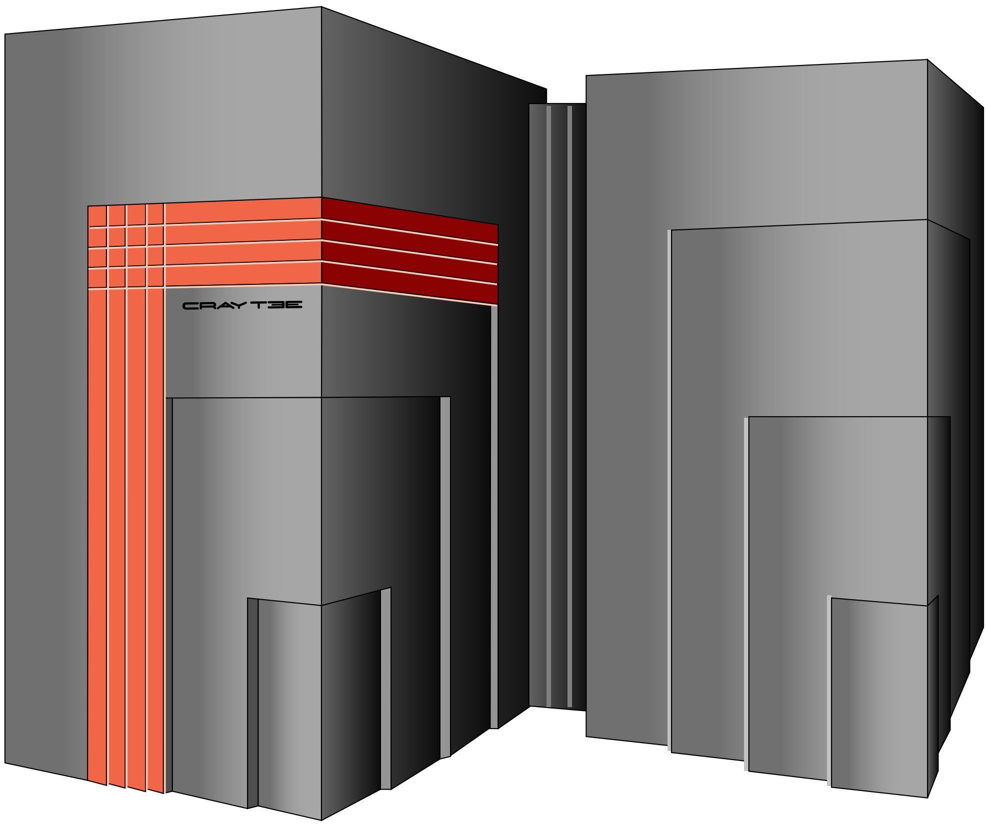
Technische Dokumentation

Cray Research
Massiv-paralleles Prozessorsystem
CRAY T3E

25. 3. 1996



Wilfried Oed

Cray Research GmbH

Riesstraße 25
80992 München
Tel.: 089-14903-0
Fax: 089-14903-149

Inhaltsverzeichnis

1 Das MPP-Konzept von Cray Research

Die spezielle Eigenschaft von Digitalcomputern, daß sie jede andere Maschine mit unstetigen Zuständen nachahmen können, läßt sich auch so beschreiben, daß sie ***universelle Maschinen*** *sind. Die Existenz von Maschinen mit dieser Eigenschaft hat die wichtige Konsequenz, daß es von* ***Geschwindigkeitserwägungen*** *abgesehen unnötig ist, immer neue Maschinen für* ***unterschiedliche Rechenprozesse*** *zu entwickeln.*

Alan Turing 1950

1.1 Das Potential massiv-paralleler Prozessorsysteme

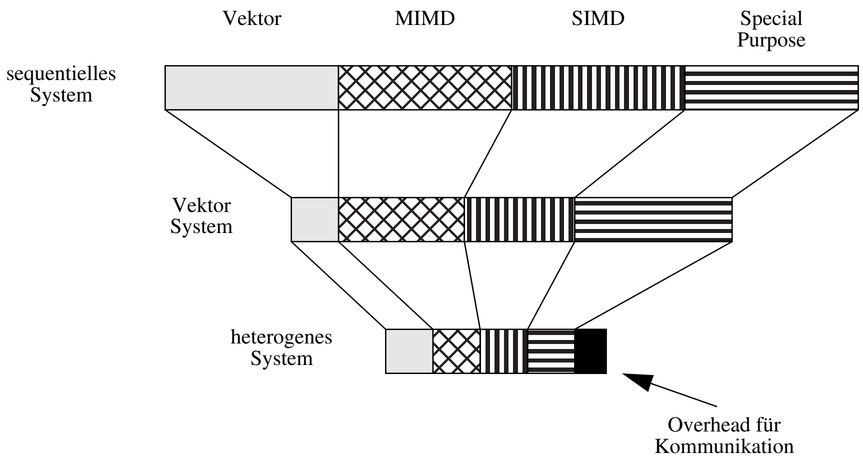
Die Anmerkung des Informatikpioniers Alan Turing ist aktueller denn je zuvor. Während bis vor kurzer Zeit der Mainframe als "universelle Maschine" für die "unterschiedlichsten Rechenprozesse" alle notwendigen Eigenschaften in sich vereinigte, wird dieses Konzept zunehmend in Frage gestellt. Man setzt auf Server-Konzepte, bei denen mehrere, mit speziellen Aufgaben betraute Maschinen in einem leistungsfähigen Netzwerk eng miteinander gekoppelt werden.

Der Gesichtspunkt "Geschwindigkeitsüberlegungen" war speziell im technisch-wissenschaftlichen Anwendungsbereich wesentlich früher von Bedeutung und führte zur Entwicklung der Supercomputer, die auf dem Vektorprinzip beruhen. Bis heute stellen diese Systeme, vor allem dank ihrer ausgereiften Software die Leistungsspitze für ein breites Anwendungsspektrum dar, ja sie sind mittlerweile sogar in die Rolle eines Mainframe geschlüpft.

1.1.1 Heterogene Systeme

Die Verfügbarkeit äußerst leistungsfähiger Mikroprozessoren, zusammen mit der Expansion der Speichergröße ermöglichen die Konzeption unterschiedlichster Maschinen, sowohl für die Abdeckung der Grundanforderungen an File- und Compute-Serving, als auch an absolute Spitzenleistung. Letzteres erscheint besonders vielversprechend über massiv-parallele Prozessorsysteme (MPP) erreichbar zu sein. Damit lassen sich Klassen von Anwendungen in Angriff nehmen, die selbst mit den schnellsten heute verfügbaren Supercomputer Systemen nicht oder nur unzureichend berechenbar sind.

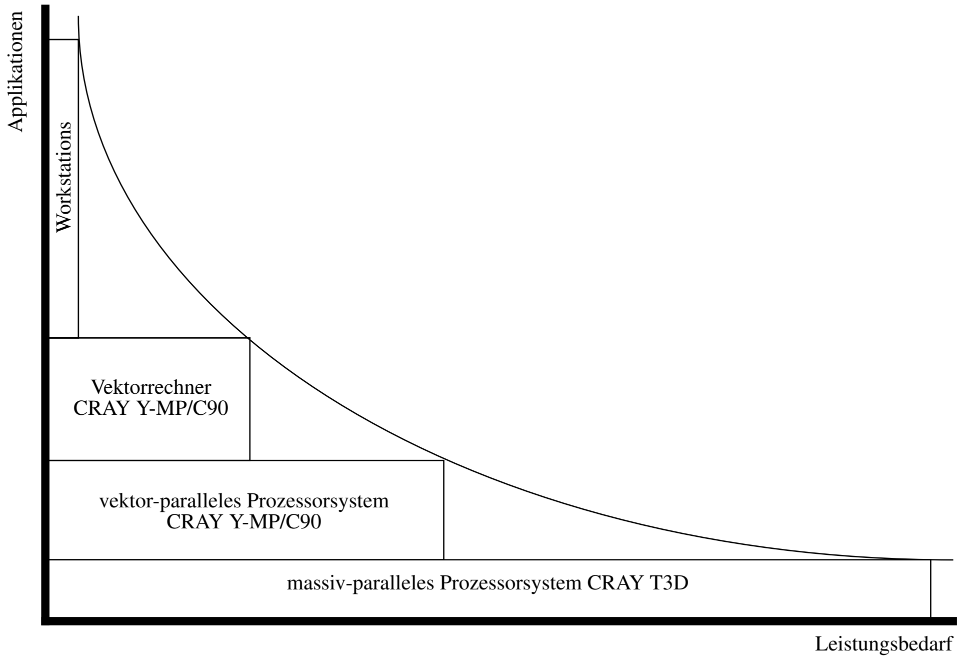
Abbildung 1 Szenario für die Abarbeitung eines Beispiels auf einem skalaren, Vektor sowie heterogenen System; das Potential kann nur bei minimalem Kommunikationsaufwand ausgeschöpft werden



Andererseits ist das technisch-wissenschaftliche Anwendungsspektrum sowohl in seiner Struktur als auch seinen Leistungsanforderungen so vielseitig, daß MPP-Systeme als Ergänzung bzw. Erweiterung "klassischer" Vektorrechner zu sehen sind. Heterogene Systeme bieten zwar die Voraussetzung im Verbund miteinander effizient und leistungsfähig einsetzbar zu sein, aber eben nur dann, wenn die jeweilige Maschine in dem für sie geeigneten Bereich eingesetzt wird. In Abbildung 1 ist beispielsweise dargestellt, wie sich unterschiedliche Spektren verhalten, wenn sie von einem sequentiellen Prozessorsystem, auf ein Vektorsystem oder schließlich auf ein heterogenes System übertragen werden. In dieser einfachen Annahme wird zwar ein prinzipieller Kommunikationsaufwand berücksichtigt, die Quantifizierung dessen ist für die Praxis jedoch eminent wichtig.

Qualitativ läßt sich dieser Sachverhalt so veranschaulichen, daß eine große Zahl von Applikationen mit kleiner bis mittlerer Leistungsanforderung typischerweise auf Workstations berechnet wird, bei steigender Leistungsanforderung auf Vektorrechner, inklusive der Ausnutzung moderater Parallelität übergegangen wird und Anwendungen allerhöchster Leistungsanforderung auf massiv-parallelen Systemen behandelt werden (Abbildung 2).

Abbildung 2 Applikationsspektrum im Vergleich zu Leistungsbedarf



1.1.2 Skalierbarkeit

Eine prinzipielle Hürde zum erfolgreichen Einsatz massiv-paralleler Prozessorsysteme ist in Amdahl's Law formuliert, d.h. die erwartete Leistungssteigerung (Speedup) ist praktisch nur bei vollständiger Parallelisierbarkeit des Programms zu erreichen. Ein Ausweg aus diesem Dilemma wird unter dem vielschichtigen Begriff "Skalierbarkeit" subsumiert, der sich durch Beschränkungen unterschiedlicher Art charakterisieren läßt:

- Konstante Problemgröße, d.h. ein existentes Problem soll durch Parallelisierung in kürzerer Zeit gelöst werden. Dieser Ansatz wird im allgemeinen durch den skalaren Anteil limitiert.
- Ausnutzung des Speicherplatzes, d.h. die Problemgröße wächst mit dem verfügbaren Speicherplatz. Dieser Ansatz führt aber meist zu einem prohibitiven Anwachsen der benötigten Rechenzeit, da die Anzahl der Operationen häufig $O(n^2)$ oder $O(n^3)$ für eine Problemgröße $O(n)$ beträgt.

- Konstante Zeit, d.h. die Problemgröße wird nur soweit angepaßt, daß mehr Prozessoren dazu verwendet werden, ein entsprechend größeres Problem in unveränderter Zeit zu lösen. Dieser Ansatz setzt jedoch voraus, daß der skalare Anteil unverändert bleibt und auch kein zusätzlicher Overhead durch die Parallelisierung entsteht.

Neben der algorithmischen Skalierbarkeit, muß auch die Architektur des Rechnersystems skalierbar sein. Ein Prozessorsystem kann aber nur dann als skalierbar angesehen werden, wenn alle beteiligten Komponenten in entsprechender Weise mitwachsen. Eine Verdoppelung beispielsweise der Anzahl der Prozessoren, muß ebenfalls eine Verdoppelung des Speicherplatzes und vor allem der Bandbreiten mit sich bringen, denn jeglicher Kommunikationsbedarf muß, auf Verweilzeit bezogen, konstant bleiben. Dazu gehört nicht zuletzt die Möglichkeit, die beteiligten Prozessoren schnell zu synchronisieren.

1.2 Strategie von Cray Research

Cray Research hat sich zum Ziel gesetzt, massiv-parallele Prozessorsysteme zu liefern, die ein breites Spektrum technisch-wissenschaftlicher Anwendungen mit Dauerleistungen im TFLOPS-Bereich bearbeiten können. Dieses Projekt wurde formell Mitte 1989 konstituiert. Dem vorangegangen war eine etwa zweijährige Phase extensiver Analyse und Entwicklungsstudien von Architekturen mit verteiltem Speicher.

Durch die Nutzung von Standard-Bausteinen, wie CMOS DRAM Speicher-Chips und RISC Mikroprozessoren kann die Entwicklung dem jeweiligen Stand dieser Technologien folgen. Ausschlaggebend für die tatsächliche Leistungsfähigkeit eines massiv-parallelen Rechnersystems ist neben der Prozessorleistung die effiziente Kommunikation der Prozessorelemente (PEs) untereinander bzw. dem effizienten Datenaustausch zwischen den zugehörigen Speichern sowie die Unterstützung eines flexiblen Programmiermodells. Diesen Anforderungen trägt das System durch ein MIMD-Konzept mit global adressierbarem Speicher, einer hohen Bandbreite für den Datenaustausch sowie extrem schnellen Synchronisationsmechanismen Rechnung. Das Hauptaugenmerk liegt dabei vor allem auf der Verbindung der zugehörigen lokalen Speicher und nicht nur auf der reinen Verbindung zwischen den Prozessoren.

Die Entwicklung muß durch eine richtig balancierte Kombination der Hardware und Software getragen werden. In diesem Punkt bringt Cray Research die Erfahrung von nunmehr 20 Jahren als Marktführer von Supercomputer Systemen ein. In enger Zusammenarbeit mit den Kunden, Anwendern und Software-Häusern erfolgt parallel zur Entwicklung der Hardware die Implementierung von Software und Applikationen.

Die erste System-Generation ist unter dem Produktnamen CRAY T3D mit bis zu 2048 Prozessoren und einer Spitzenleistung von 300 GFLOPS 1994 fertiggestellt und vom Markt äußerst erfolgreich aufgenommen worden.

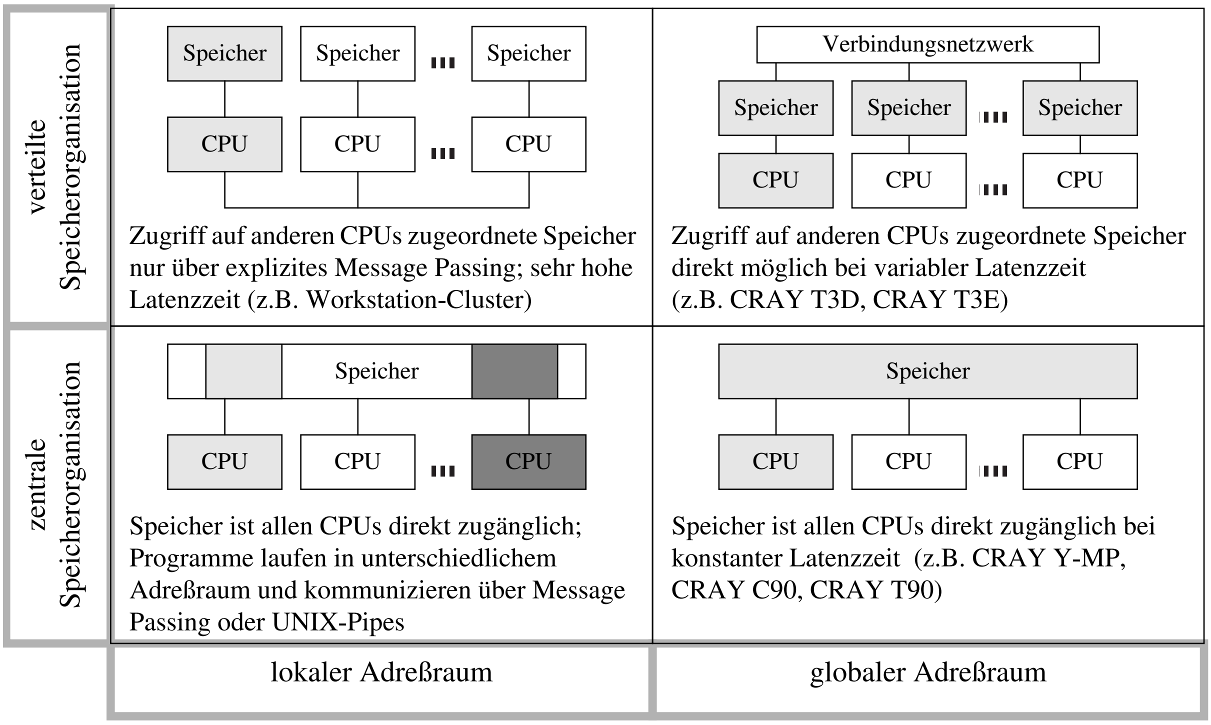
Neue RISC Mikroprozessor-Technologien ermöglichen es heute, ein massiv-paralleles Rechnersystem mit einer Spitzenleistung von 1 TFLOPS zu produzieren. Unter dem Produktnamen CRAY T3E ist die zweite Generation 1996 auf dem Markt verfügbar. Weitere zu erwartende technologische Fortschritte werden etwa 1997/98 zu einem System mit mehreren TFLOPS Spitzenleistung führen, das eine effektive Dauerleistung von 1 TFLOPS realen Anwendungen zur Verfügung stellen wird.

Eine Schlüsselstellung innerhalb dieses Stufenkonzepts nimmt eine stabile Macro-Architektur ein, d.h. die dem Anwender sichtbare Schnittstelle. Konzepte der Programmierung und Optimierung werden mit der jeweils neuen Generation skalieren. Lediglich die zugrunde liegende Micro-Architektur, d.h. die eingesetzten Hardware-Bausteine, wird dem jeweiligen technologischen Stand angepaßt.

1.3 Macro-Architektur der skalierbaren CRAY MPP-Systeme

Das grundlegende Architekturkonzept der massiv-parallelen Prozessorsysteme von Cray Research ist ein skalierbares MIMD (Multiple Instruction Multiple Data) System mit einem physisch verteilten, doch logisch global adressierbaren Speicher. Abbildung 3 zeigt den Zusammenhang zwischen physischer Speicherorganisation und Adreßraum.

Abbildung 3 Relation zwischen Speicherorganisation und Adreßraum



Dieses Konzept vermittelt dem Anwender die Vorteile eines parallelen Rechnersystems mit gemeinsamem Speicher und eignet sich somit für eine breite Klasse von Anwendungen sowie für die Parallelisierung von Konstrukten mit kleiner Granularität.

1.3.1 Prozessor und Addressierung

Eine wesentliche Frage beim Entwurf eines MPP-Systems stellt die Auswahl des Prozessors dar. Aufgrund der momentanen schnellen Weiterentwicklung der RISC-Prozessoren ist eine Festlegung auf einen bestimmten Prozessortyp oder gar die Entwicklung eines proprietären Prozessors für die gesamte Architekturlinie nicht sinnvoll. Cray Research hat sich daher entschlossen, einen kommerziell verfügbaren Micro-Prozessortyp einzusetzen, der folgende Grundeigenschaften besitzen muß:

- 64 Bit Arithmetik (Integer und Floating-Point)
- höchste Spitzenleistung zum jeweiligen Zeitpunkt
- sehr hohe Taktfrequenz, auch außerhalb des Prozessors
- Verfügbarkeit zum Zeitpunkt des Entwurfs der jeweiligen Maschinengeneration

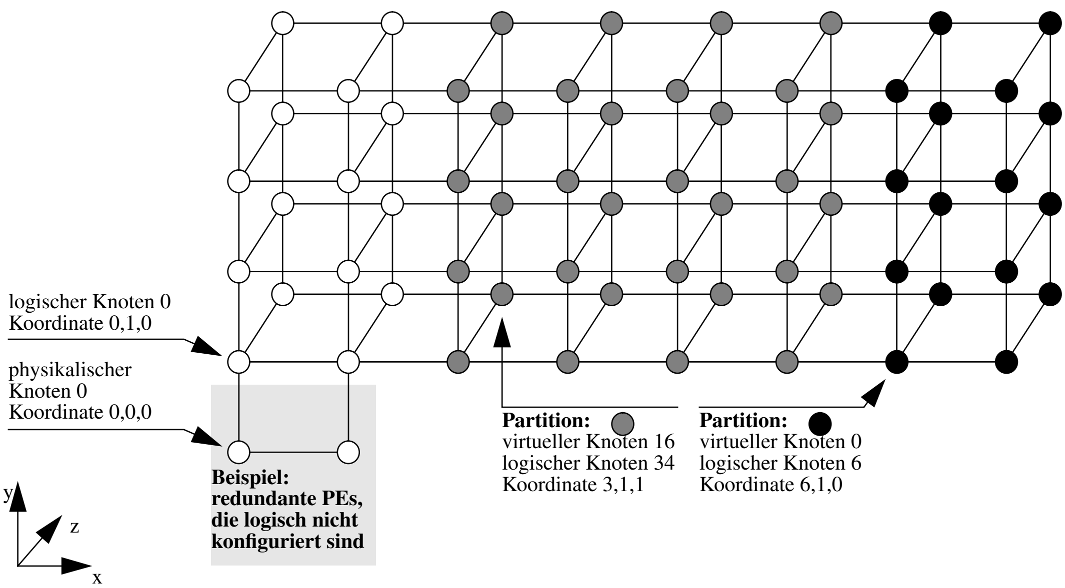
Diese Vorgehensweise stellt sicher, daß die Systementwicklung im vorgesehenen Zeitrahmen erfolgen kann. Der Prozessor selbst ist eingebettet in eine sogenannte Support-Hardware, die die vielen Einzelprozessoren zu einem homogenen massiv-parallelen Prozessorsystem verbinden.

Der wichtigste Gesichtspunkt hierbei ist die effiziente Adressierung des globalen Adreßraums, d.h. jede Speicherzelle des Gesamtsystems kann unmittelbar angesprochen werden. Eine globale Adresse in einem MPP-System mit physisch verteiltem Speicher besteht prinzipiell aus der PE-Nummer sowie dem Offset in den lokalen Speicher. Da ein MPP-System nicht stets einer Anwendung vollständig zur Verfügung gestellt werden soll, muß bei der Adressierung die Partitionierung des Systems berücksichtigt werden. Daraus ergeben sich drei Ebenen der PE-Adressierung:

- Virtuelle PE-Nummer, das was die Anwendung sieht, unabhängig von der Lage der Partition
- Logische PE-Nummer, das was das Betriebssystem sieht, d.h. alle benutzten PEs
- Physische PE-Nummer, das was die Hardware sieht, d.h. alle tatsächlich vorhandenen PEs inklusive der redundanten PEs.

Die Umsetzung von virtueller, zu logischer und schließlich zu physischer Adresse erfolgt aus Effizienzgründen in Hardware. Abbildung 4 stellt die verschiedenen Adreßbereiche in einem 3D Torus dar (Torus-Verbindungen sind nicht eingezeichnet).

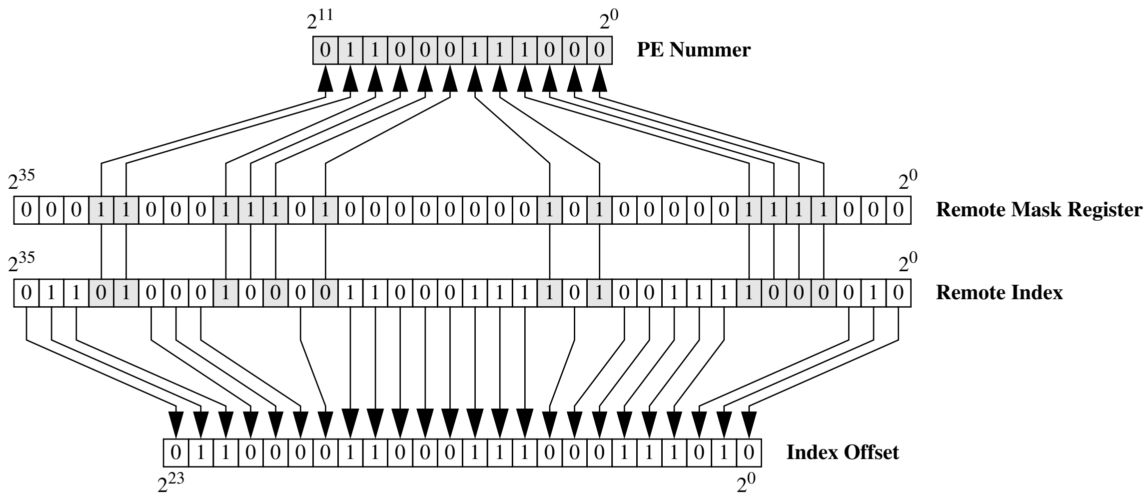
Abbildung 4 Adreßbereiche in einem partitionierten System



Zur flexiblen Adressierung des gesamten Adreßraums ist außerdem das Konzept der "Adreß-Zentrifuge" entwickelt worden. Darunter versteht man die Möglichkeit, bei fortlaufender Adressierung (Strides) sich nicht auf den lokalen Speicher beschränken zu müssen. Dazu wird die jeweilige Adresse (Remote Index) über ein Masken-Register automatisch in PE-Nummer und lokalen Offset separiert. Abbildung 5 zeigt das Prinzip der Adreß-Zentrifuge und darunter drei Beispiele, wie bei fortlaufender Indexerhöhung in Abhängigkeit des Masken-Registers unterschiedliche Datenblöcke angesprochen werden:

- Linke Spalte: es werden je acht zusammenhängende Worte aus einem PE angesprochen, danach erfolgt der Wechsel zum nächsten PE
- Mittlere Spalte: aus vier PEs werden nacheinander jeweils die gleichen Adressen angesprochen, d.h. es wird durch die PEs adressiert
- Rechte Spalte: Adreßpaare werden aus Prozessorpaaren angesprochen, wie sie beispielsweise bei einer Blockzerlegung eines Feldes entstehen.

Abbildung 5 Adreß-Zentrifuge und Beispiele für Wirkungsweise



Remote Index Bits 2^4 - 2^0	Remote Index Bit Formate					
	2^4, 2^3 = PE Nummer 2^2, 2^1, 2^0 = Index Offset		2^1, 2^0 = PE Nummer 2^4, 2^3, 2^2 = Index Offset		2^3, 2^1 = PE Nummer 2^4, 2^2, 2^0 = Index Offset	
	PE Nummer	Index Offset	PE Nummer	Index Offset	PE Nummer	Index Offset
00000	0	0	0	0	0	0
00001	0	1	1	0	0	1
00010	0	2	2	0	1	0
00011	0	3	3	0	1	1
00100	0	4	0	1	0	2
00101	0	5	1	1	0	3
00110	0	6	2	1	1	2
00111	0	7	3	1	1	3
01000	1	0	0	2	2	0
01001	1	1	1	2	2	1
01010	1	2	2	2	3	0
01011	1	3	3	2	3	1
01100	1	4	0	3	2	2
01101	1	5	1	3	2	3
01110	1	6	2	3	3	2
01111	1	7	3	3	3	3

1.3.2 Interprozessor-Verbindungsnetzwerk

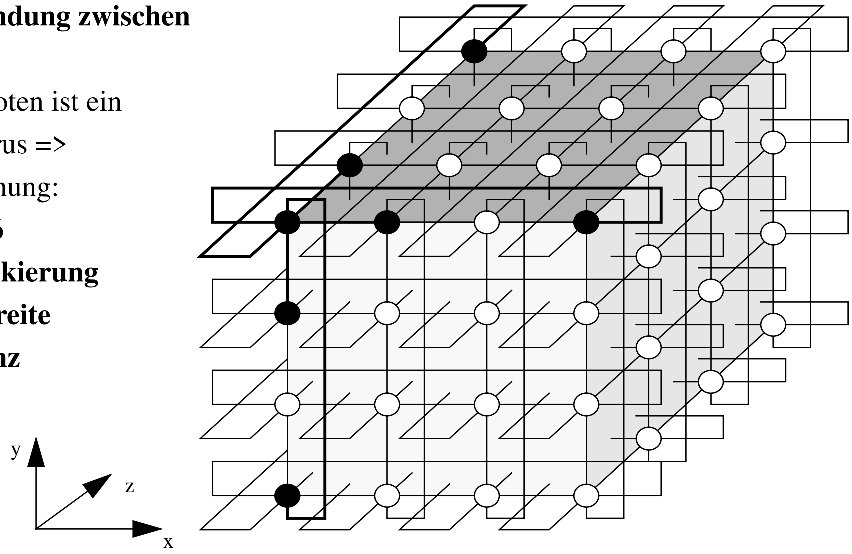
Ein effizientes Verbindungsnetzwerk mit geringen Latenzzeiten stellt eine wesentliche Voraussetzung für ein leistungsfähiges MPP-System dar. In vielen Fällen läßt sich Parallelität nur auf feingranularer Ebene nutzen und erfordert dementsprechend viel Kommunikation.

Eine Klasse von Netzwerken stellen die *k*-ary *n*-cubes dar, d.h. ein Verbindungsnetzwerk der Dimension *n* und *k* Knoten in jeder Dimension. Je höher die Dimension, desto mehr Knoten sind direkt miteinander verbunden. Allerdings lassen sich bei Netzwerken sehr hoher Dimension, wie beispielsweise einem Hypercube, einem binären *n*-cube, nur sehr geringe Bandbreiten zwischen benachbarten Knoten realisieren. Darüberhinaus bleiben bei einem Hypercube viele Verbindungen ungenutzt, falls das System nicht voll ausgebaut ist. Daher werden heute bevorzugt Gittertopologien eingesetzt.

Im Cray Research MPP-System ist die Netzwerktopologie ein dreidimensionaler Torus, d.h. ein dreidimensionales, ringförmig geschlossenes Gitter (Abbildung 6). Dadurch werden kurze Verbindungswege sowie eine hohe Bandbreite (Bisection Bandwidth) gewährleistet.

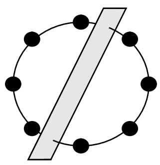
Abbildung 6 3D-Torus Netzwerktopologie

Kurze Verbindung zwischen den Knoten
z.B. 1024 Knoten ist ein
16 x 8 x 8 Torus =>
größte Entfernung:
8 + 4 + 4 = 16
Geringe Blockierung
Hohe Bandbreite
Fehlertoleranz



Unter Bisection versteht man, ein gegebenes Netzwerk bei minimaler Anzahl von Schnitten durch die Verbindungsleitungen in zwei Hälften zu teilen. Die Bisection-Bandbreite ist die Bandbreite durch die Schnittebene und stellt ein Maß für die Kommunikation durch das System hindurch dar. Je größer die Bisection-Bandbreite, desto geringer ist die Wahrscheinlichkeit von Blockierungen im Netz bei allgemeiner Kommunikation. Abbildung 7 zeigt Bisection B sowie deren Berechnung für einige Topologien. Die Bisection-Bandwidth ergibt sich durch Multiplikation der Übertragungskapazität (Bytes/sec) einer Leitung mit der Bisection. Sind die Leitungen bidirektional ausgelegt, verdoppelt sich die Bisection-Bandwidth.

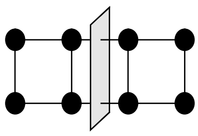
Abbildung 7 Bisection für Ring-, 2D Gitter-, 2D-Torus- und 3D Torus-Topologie



Die Teilung eines Rings erfordert, unabhängig von der Systemgröße, stets einen Schnitt durch zwei Verbindungsleitungen. Damit ergibt sich eine konstante Bisection von:

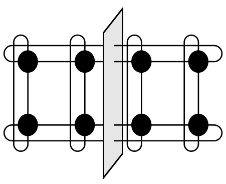
$B_{Ring} = 2$

Eine Ring-Topologie skaliert demnach nicht mit der Systemgröße.



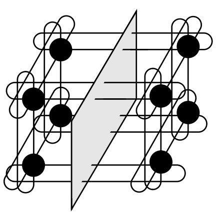
Die Teilung eines 2D Gitters der Dimension (`m x n`) mit minimaler Schnittfläche erfolgt durch die Ebene der größeren der beiden Dimensionen. Damit ergibt sich eine Bisection von:

$B_{2D\ Gitter} = \min(m,n)$



Die Teilung eines 2D Torus der Dimension (`m x n`) mit minimaler Schnittfläche erfolgt wie beim 2D Gitter. Aufgrund der Torusverbindung verdoppelt sich jedoch die Bisection und ergibt zu:

$B_{2D\ Torus} = 2 * \min(m,n)$



Die Teilung eines 3D Torus der Dimension (`l x m x n`) mit minimaler Schnittfläche erfolgt durch die Ebene der maximalen Dimension. Die Torusverbindung verdoppelt auch hier im Vergleich zu einem 3D Gitter die Bisection, die sich ergibt zu:

$B_{3D\ Torus} = 2 * [l+m+n - \max(l,m,n) - \min(l,m,n)] * \min(l,m,n)$

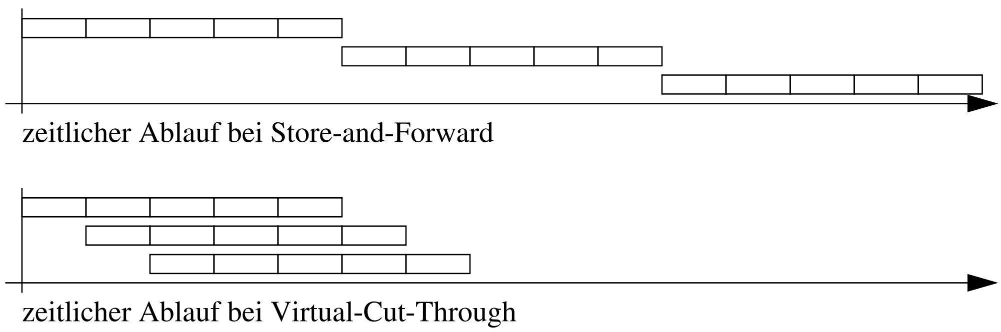
Ein 3D Torus hat demnach von allen hier vorgestellten Topologien die größte Bisection.

Jedes PE arbeitet unabhängig vom Netzwerk auf den Daten, die sich in seinem lokalen Speicher befinden. Gleichzeitig können Daten und Messages unabhängig über die separaten Kommunikationspfade durch den Knoten bidirektional in jeder der drei Koordinatenrichtungen (x, y, z) transportiert werden.

Durch die bidirektionale Auslegung der Kommunikationspfade läßt sich die häufig vorhandene Lokalität ausnutzen. So ist die Wahrscheinlichkeit sehr hoch, daß ein Knoten A, der eine Botschaft an einen Knoten B schickt, auch eine Antwort von Knoten B erhält. In unidirektional gerichtenen Netzen, z.B. Ring-Topologien, erfordert eine Frage/Antwort-Sequenz stets einen vollen Umlauf in der entsprechenden Dimension.

Das Routing-Schema stellt eine Kombination zwischen "Virtual-Cut-Through" und "Wormhole-Routing" dar. Virtual-Cut-Through ist eine Methode, die dem Store-and-Forward Prinzip entspricht, allerdings bei Ankunft der ersten Informationseinheit, einem sogenannten "PHIT" (physical transfer unit), diese sofort an den nächsten Knoten weiterleitet, falls der Pfad dorthin frei ist. Nur im Konfliktfalle werden die ankommenden Daten im Knoten zwischengespeichert (Abbildung 8).

Abbildung 8 Vergleich zwischen Store-and-Forward und Virtual-Cut-Through bei der Übertragung von drei Paketen, die jeweils aus fünf PHITs bestehen



Die Größe der Zwischenspeicher (Puffer) ist ausreichend für kleine Pakete, bei größeren werden mehrere Puffer benötigt, was dann dem Wormhole-Routing entspricht. Allerdings werden die Pfade nie vollständig geschaltet oder gar für einen langen Zeitraum blockiert. Diese Schemata, zusammen mit der Verwendung virtueller Kanäle verleihen dem Netz einen extrem hohen Durchsatz.

1.3.3 Synchronisationsmechanismen

Die PEs der Cray Research MPP-Systeme verfügen über effiziente Mechanismen zur Synchronisation auf verschiedenen Granularitätsebenen und unterstützen sowohl Kontroll- als auch Datenparallelität:

- Eine wichtige Funktion für MIMD-Systeme ist die Barrier-Synchronisation, d.h. alle Prozessoren müssen an einem Synchronisationspunkt angekommen sein. Neben der logischen UND-Verknüpfung (Barriere) ist auch die ODER-Verknüpfung möglich. Letztere, Eureka-Modus genannte Betriebsart, eignet sich beispielsweise bei parallelem Suchen. Sobald einer der Prozessoren das gesuchte Objekt gefunden hat, kann er dies den anderen mitteilen und diese können dann die Suche ihrerseits sofort beenden.
- Fetch-and-Increment führt zum automatischen Erhöhen eines Registers oder einer Speicherzelle beim Lesen. Diese Funktion ist sowohl zentral als auch verteilt zu nutzen, letzteres vermeidet mögliche "Hot-Spots". Diese atomare Funktionalität unterstützt beispielsweise das Verteilen der Iterationen parallel ausführbarer Schleifen.

- Atomic-Swap Operation tauscht den Inhalt spezieller lokaler Register mit dem Inhalt einer Speicherzelle, die auch in einem entfernten Speicher liegen kann. Verwendungsmöglichkeiten sind beispielsweise "full/empty" oder die Implementierung von Locks für kritische Regionen. Full/empty bezeichnet die Prüfung, ob eine Speicherzelle einen gültigen Wert (full) enthält oder nicht (empty).
- Eine unabhängig arbeitende Messaging-Facility ermöglicht einem Prozessor eine Nachricht zu schicken, ihn zu unterbrechen und zu veranlassen eine Antwort zurückzuschicken. Dieser Mechanismus ermöglicht effizientes Message Passing, da die Messages in vordefinierte Warteschlangen eingereiht werden und der Prozessor weitgehend entlastet ist.

Diese Funktionen garantieren zusammen mit dem dreidimensionalen, ringförmig geschlossenen Verbindungsnetzwerk eine nur geringfügig erhöhte Latenzzeit beim Zugriff auf entfernt liegende Daten gegenüber dem Zugriff auf lokale Daten. Durch die unabhängig vom Prozessor arbeitende Hardware für Datentransfers und Synchronisation lassen sich viele Operationen überlappt durchführen und Latenzzeiten größtenteils verstecken.

2 Massiv-paralleles Prozessorsystem CRAY T3E

Das CRAY T3E System stellt die zweite Generation der Cray Research massiv-parallelen Prozessorsysteme dar. Zielsetzung dieses Systems ist, unter Beibehaltung der Macro-Architektur (u.a. MIMD Konzept, globaler Adreßraum, 3D-Torus Verbindungsnetzwerk) des CRAY T3D Systems ein skalierbares System mit einer Spitzenleistung von 1 TFLOPS zu liefern.

Weiterhin schlagen sich Erkenntnisse, die bei der Entwicklung des CRAY T3D Systems sowie der zugehörigen Software-Umgebung gewonnen wurden im Entwurf des CRAY T3E Systems nieder. Diese Änderungen lassen sich stichpunktartig folgendermaßen zusammenfassen:

- DEC Alpha EV5 (21164 Implementierung) Microprozessor mit einer Spitzenleistung von 600 MFLOPS
- Speichergrößen von 64 MBytes, 128 MBytes, 256 MBytes oder 512 MBytes pro PE
- 3D-Torus mit Transferraten von ca. 500 MB/s bidirektional in jede der drei Koordinatenrichtungen ergeben eine Durchsatzleistung von 3 GB/s in einem Netzwerkknoten
- Ein Prozessor pro Netzwerkknoten
- Satz externer Register (E-Register) über die asynchroner Datentransfer erfolgt; ersetzt die Prefetch Queue (PFQ) und Block Transfer Engine (BLT) aus dem CRAY T3D System
- Cache-Kohärenz Protokoll durch Hardware
- Barrier/Eureka Netz ist im Komunikationsnetzwerk integriert
- Leichtere Skalierbarkeit (in Einheiten von 8 PEs)
- CRAY T3E System kann ohne Front-End betrieben werden (Self-Hosting); für je 16 PEs ist ein zusätzlicher OS-Knoten vorgesehen
- Luftgekühlte Systeme werden verfügbar (kleine Chassis aufrüstbar bis 128 PEs)

In den folgenden Unterabschnitten wird auf die Komponenten des CRAY T3E Systems näher eingegangen.

2.1 Technologie

Im Gegensatz zum CRAY T3D System wird im CRAY T3E System CMOS Technologie, auch für die Cray Research eigene "Support Hardware" eingesetzt. Neben dem Prozessor sind das die Netzwerk-Router sowie die Zugriffskontrolleinheiten für die 8 Bänke des lokalen Speichers, für den zunächst 16 MBit DRAM Chips verwendet werden.

2.2 Architektur

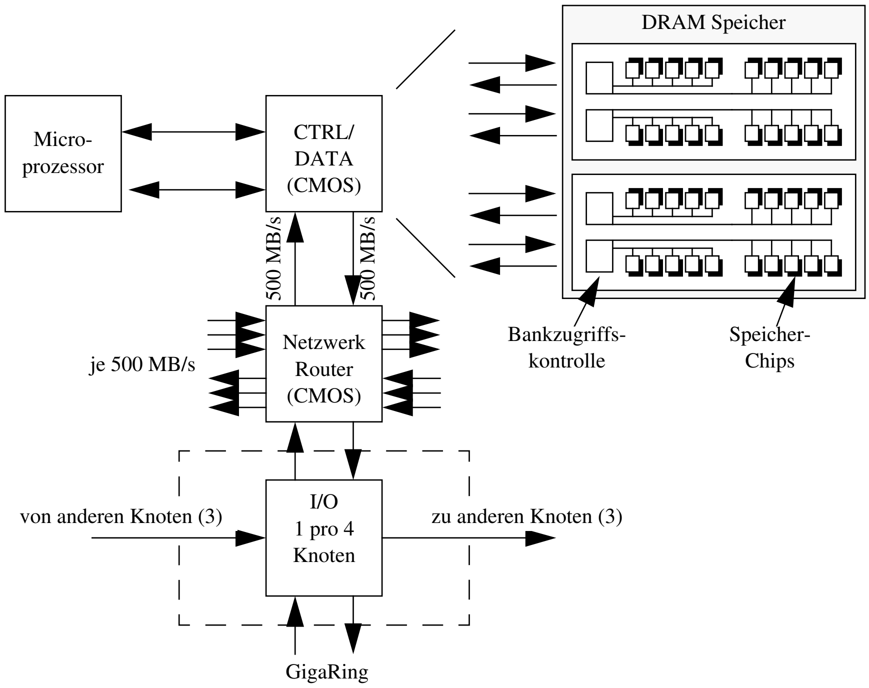
Das massiv-parallele Prozessorsystem CRAY T3E ist ein skalierbares MIMD (Multiple Instruction Multiple Data) System mit einem physisch verteilten, doch logisch global adressierbaren Speicher. Der Zugriff auf den entfernten Speicher erfolgt durch ein Netzwerk, das asynchron Datenpakete zwischen den Prozessoren verschickt.

Das CRAY T3E System ist von 16 PEs bis zu 2048 PEs skalierbar. Die Skalierung erfolgt in konstanten Inkrementen, die vom Chassis-Typ abhängen, es ist keine jeweilige Verdoppelung nötig.

Zur Unterstützung des eigenständigen Betriebs sind separate OS-Knoten vorgesehen, die identisch zu den anderen in das 3D-Torus Netzwerk integriert sind. OS-Knoten können auch als Redundanzknoten konfiguriert werden, um im Falle eines Ausfalls eines Rechenknotens diesen zu ersetzen.

Im Gegensatz zum CRAY T3D System, bei dem sich zwei PEs einen Netzknoten teilen, hat im CRAY T3E System jedes PE einen eigenen Netzknoten. Für Ein-/Ausgabe steht für je 4 Rechenknoten ein separater I/O-Knoten zur Verfügung, der über das Kommunikationsnetzwerk erreicht wird.

Abbildung 9 CRAY T3E Knotenorganisation



2.3 Prozessor

Als Prozessor wird die nächste Generation des DEC Alpha Microprozessors (EV5 - 21164 Implementierung) verwendet. Gegenüber dem im CRAY T3D System eingesetzten EV4 Prozessor kann EV5 einerseits mit einer höheren Taktfrequenz betrieben werden, andererseits wurden architektonische Maßnahmen ergriffen, die die Flexibilität des Prozessors erhöhen. Wesentliche Eigenschaften des in CMOS Technologie ausgelegten Prozessors sind:

- 300 MHz Taktfrequenz (3.3 nsec)
- IEEE 64 Bit Arithmetik
- Super-skalarer Prozessor mit einer Instruktionsrate von bis zu 4 pro Zyklus (2 Gleitkomma, 2 Integer/Logic), entsprechen einer maximalen Leistung von 1200 MIPS und 600 MFLOPS
- 4 unabhängige, parallel betreibbare Pipelines (2 Gleitkomma, 2 Integer/Logic); die Gleitkommaeinheiten sind partiell spezialisiert, mit einer für Gleitkommaaddition, der anderen für Gleitkommamultiplikation
- Division wird auf 64 Bit durchgeführt
- 43 Bit virtueller Adreßraum und 40 Bit realer Adreßraum
- Speicherbandbreite vom/zum Prozessor beträgt ca. 1200 MB/s (externer Datenbus)

- Auf dem Prozessor-Chip sind 3 Caches vorhanden:
  - 8 KBytes Daten-Cache (D-Cache), der in 256 Cache-Lines zu je 4 Worten (64-Bit) unterteilt ist
  - 8 KBytes Instruktions-Cache (I-Cache), Unterteilung wie im D-Cache
  - 96 KBytes Secondary Cache (S-Cache), der in 3 assoziativen Sätzen zu je 512 Cache-Lines unterteilt ist und sowohl den D-Cache als auch den I-Cache bedient.

Die Instruktionsinitiierung ist flexibler geworden mit bis zu vier gleichzeitig absetzbaren Instruktionen, u.a. zwei Lade- und eine Speicherinstruktion im gleichen Takt. Instruktionen werden in "Slots" eingereiht, die den entsprechenden Funktionseinheiten zugeordnet sind. Die Slots gewährleisten auch die Einhaltung der Reihenfolge von Instruktionen soweit dies nötig ist.

Lade-Operationen, deren Operanden nicht im D-Cache liegen, werden in einem "Missed-Address File" (MAF) zusammengeführt. Im MAF können bis zu 24 zusammenhängende Adressen (jeweils innerhalb einer Cache-Line) oder 6 nicht zusammenhängende untergebracht werden. Dieser Mechanismus verhindert ein vorzeitiges Anhalten des Prozessors (processor stall) bei fehlenden Operanden. Zunächst wird versucht, die fehlenden Operanden aus dem S-Cache zu laden, was mit einer Transferrate von zwei 64 Bit Worten pro Takt (4.8 GB/s) erfolgt. Sind die benötigten Daten auch nicht im S-Cache, werden sie über den 128 Bit breiten Datenbus aus dem Speicher angefordert.

Schreib-Operationen werden zunächst in einem Write-Buffer gesammelt. Der Write-Buffer kann ebenfalls bis zu 24 zusammenhängende, bzw. 6 nicht zusammenhängende Adressen aufnehmen.

Abbildung 10 DEC Alpha EV5 Prozessor-Blockschaltbild

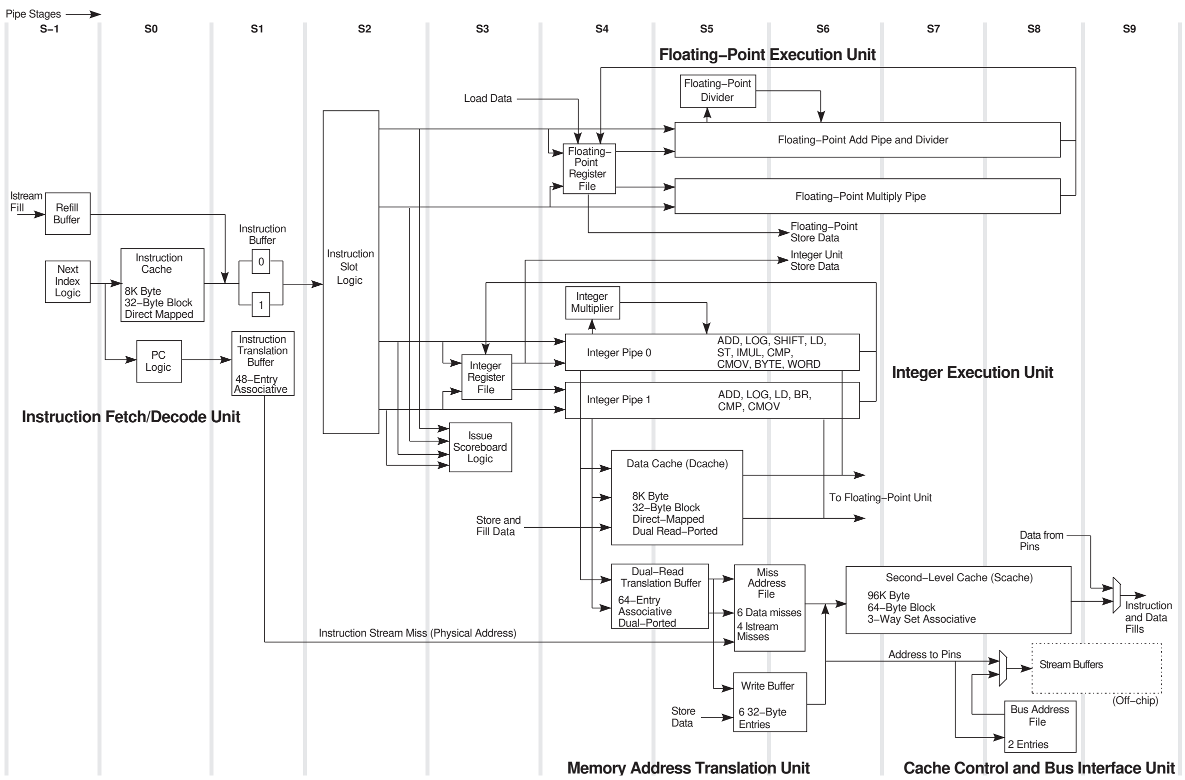


Abbildung 10 zeigt das Blockschaltbild des DEC EV5 Prozessors zusammen mit der Zeiteinteilung der Pipeline-Stufen.

2.4 Speicherorganisation

Obwohl das CRAY T3E System über einen globalen Adreßraum (global shared memory space) verfügt, ergeben sich aufgrund der physikalischen Verteilung (distributed memory), für die Anwendung transparente Unterschiede in der Hardware-Realisierung der Adressierung. Die beiden wesentlichen Komponenten bestehen aus dem lokalen sowie dem entfernten Speicher:

- Der lokale Speicher bezeichnet die über lokale Adressierung zugänglichen Speichereinheiten bestehend aus:
  - dem lokalen DRAM Speicher, der Programme und ihre zugehörigen Daten enthält,
  - einem Registersatz in schneller SRAM Technologie (E-Register) als Schnittstelle für lokalen und entfernten Speicherzugriff,
  - einem zweiten Registersatz in schneller SRAM Technologie für spezielle Funktionen (memory mapped registers).
- Der entfernte Speicher bezeichnet die über entfernte Adressierung auf anderen PEs zugänglichen, d.h. dort lokalen, Speichereinheiten. Das sind der DRAM Speicher und die E-Register.

Der lokale Adreßraum eines jeden Prozessors ist in zwei Bereiche unterteilt, was anhand des obersten Bits der realen Adresse unterschieden wird:

- Cached Bereich, der direkt vom Prozessor über seine eigenen Zugriffsmethoden inklusive D-Cache und S-Cache verwaltet wird. Dieser Bereich ist ausschließlich vom lokalen DRAM Speicher aus zugänglich. Lokaler Zugriff wird von der Hardware zwischen Speicher und Cache automatisch kohärent gehalten. Dazu wird das im Prozessor vorhandene "Flush Protokoll" verwendet. Sobald ein externer Zugriff auf den lokalen Speicher erfolgt, wird die entsprechende Cache-Line in beiden Daten-Caches (D-Cache und S-Cache) ungültig markiert.
- Non-Cached Bereich, der von der externen Cray Research eigenen "Support Hardware" verwaltet wird. Zugriffe auf diesen Bereich, z.B. E-Register, erfolgen ohne Beeinträchtigung des Cache direkt in die Register des Microprozessors.

2.4.1 Lokaler DRAM Speicher

Der lokale Speicher ist in acht Bänken unterteilt. Die Verschränkung (Interleaving) erfolgt auf der Ebene von 64 Bit Worten, so daß auf Cache-Line Ebene ein quasi-paralleler Zugriff gegeben ist. Eine Speicherbank besteht aus fünf oder zehn 16 MBit DRAM Speicher-Chips, womit der lokale Speicher insgesamt aus 40 (64 MB) oder 80 (128 MB) Chips besteht; größere Speicher entstehen durch Stapelung. Hinzu kommen vier CMOS Bank-Control-Chips.

DRAM Speicher-Refresh wird von der Hardware durchgeführt, die Refresh-Rate läßt sich durch Software einstellen.

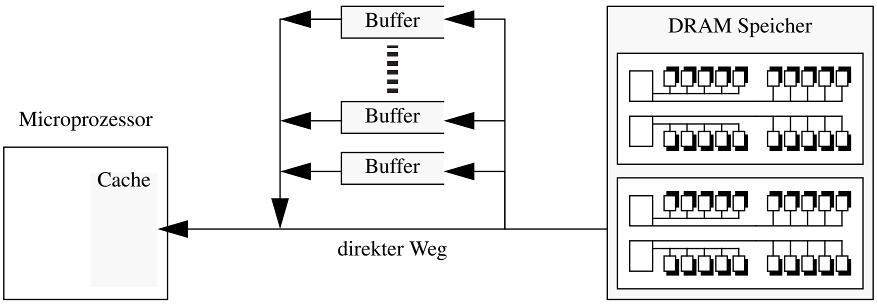
2.4.2 Stream-Buffers

Zur Unterstützung des schnellen Speicherzugriffs wird in Workstations der On-Chip-Cache typischerweise durch einen mehrere MBytes großen externen Cache erweitert. Um seinen Zweck zu erfüllen, muß dieser externe Cache in schneller SRAM Technologie realisiert werden. Für massiv-parallele Prozessorsysteme sind große externe Caches nicht praktikabel, da sie für jedes PE realisiert werden müßten und deshalb Platz- sowie Kühlprobleme aufwerfen.

Anstelle eines externen Cache, werden im CRAY T3E System externe Stream-Buffer eingesetzt. Dieser Mechanismus benötigt wesentlich weniger Hardware und zeigt speziell für Vektor- und Feldzugriffe aufgrund der höheren Assoziativität deutlich bessere Leistung als ein externer Daten-Cache.

Sobald die Microprozessor-externe Hardware den Zugriff auf einen Datenstrom erkennt, erfolgt ein Read-ahead auf den DRAM Speicher in einen der sechs Stream-Buffer. Ein Strom ist hierbei einfach der Zugriff auf hintereinanderliegende Datenbereiche im Speicher. Ein weiterer Vorteil gegenüber einem Cache besteht darin, daß die zeitliche Reihenfolge der Zugriffe unerheblich ist. Abbildung 11 zeigt das Konzept, das die hohe externe Bandbreite zum DRAM Speicher nutzt, die sich durch die Speicherverschränkung ergibt.

Abbildung 11 Konzept der Stream-Buffer



Das "Erkennen" eines Datenstroms läßt sich über ein STREAM_CTRL Register steuern. Sobald mehr Zugriffe, als in dem Register angegeben auf hintereinanderliegende Adressen erfolgt sind, wird ein Strom im nächsten freien Stream-Buffer allokiert und das Read-ahead beginnt. Weitere Anforderungen seitens des Prozessors werden unmittelbar aus dem Stream-Buffer befriedigt.

2.4.3 E-Register und Zugriff auf den entfernten Speicher

Der Zugriff auf den entfernten Speicher, d.h. den lokalen Speicher eines anderen PEs, wird über einen externen Registersatz, die sogenannten E-Register behandelt. E-Register sind zwar Teil des Adreßbereichs (memory-mapped), befinden sich jedoch in schnellen SRAM Chips, um somit die höchstmögliche Bandbreite und kürzeste Latenzzeit zum Prozessor zu erreichen. Der E-Registersatz wird unterteilt in einen mit Zugriff auf Anwenderebene (512 Einträge) und einen für privilegierten Zugang (128 Einträge). Jedes E-Register hat zwei Status-Bits assoziiert, über die Funktionen wie full/empty oder Fehler markiert werden.

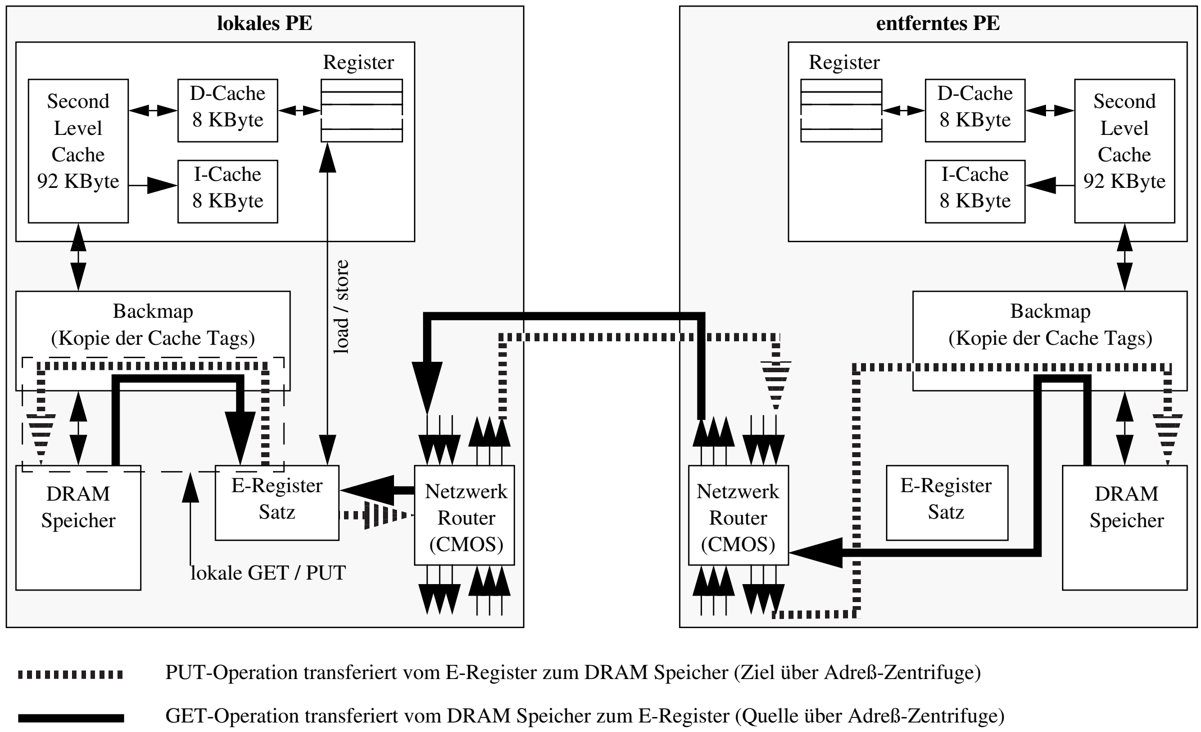
Zur Unterscheidung wird das Lesen (bzw. Schreiben) eines E-Register aus dem DRAM Speicher (lokal oder entfernt) mit GET (bzw. PUT), das Lesen (bzw. Schreiben) eines Prozessor-Registers aus einem E-Register mit Load (bzw. Store) bezeichnet. Load/Store Operationen gehen am Cache vorbei.

Der Zugriff auf den entfernten Speicher erfolgt grundsätzlich über den indirekten Weg der E-Register. Soll beispielsweise ein Wort aus einem entfernten Speicher in das Register des lokalen Prozessors geladen werden, so sind folgende Schritte nötig (Abbildung 12):

- Schreiben (store) der Maske für die Adreß-Zentrifuge und der Basisadresse des entfernten Speichers in ein lokales E-Register; dieser Schritt kann entfallen, sofern noch gültige Masken von früheren Zugriffen her exisitieren.
- Absetzen einer GET-Operation, deren Index mit Zeiger auf die zugehörige Maske in ein E-Register bewirkt, daß die externe Harwdware über die Adreß-Zentrifuge ein Paket in das Netzwerk absetzt, um das geforderte Wort aus dem entfernten Speicher in ein lokales E-Register zu transportieren.

- Lesen (load) des lokalen E-Register; sollte das angeforderte Wort noch nicht eingetroffen sein (Status-Bits), wartet der Prozessor; soll nicht gewartet werden kann alternativ ein Test abgesetzt werden.

Abbildung 12 GET/PUT-Operationen der E-Register



GET-Operationen können spekulativ abgesetzt werden, um somit Latenzzeiten zu verstecken. Dabei erfolgt keine Unterbrechung oder Fehlermeldung, falls außerhalb des zulässigen Adreßraums gegriffen wird. Erst wenn versucht wird, das entsprechende E-Register tatsächlich zu lesen erfolgt eine Fehlermeldung (E-Register Status-Bits zeigen Fehler).

GET/PUT-Operationen stellen automatisch die Kohärenz zwischen dem Prozessor-Cache und seinem zugehörigen lokalen Speicher sicher. Dazu wird eine externe Kopie der Cache-Tags überprüft, ohne den Prozessor zu stören. Sollte sich das fragliche Wort im Cache befinden, wird dieser im Falle einer PUT-Operation invalidiert. Im Falle einer GET-Operation wird die entsprechende Cache-Line aus dem S-Cache in den Speicher geschrieben (write-back protocol).

E-Register Transfers können auf einzelnen Worten oder Vektoren von acht Worten erfolgen (PUTV / GETV). Bei entsprechender Maske für die Adreß-Zentrifuge führt dies in der verallgemeinerten Form zu regelmäßigen Gather/Scatter Operationen (Strides) auch auf verschiedene PEs.

Schließlich sei noch auf spezielle Funktionen hingewiesen, die über den E-Register Mechanismus implementiert sind. Diese erweitern einerseits die Funktionalität des Prozessors, andererseits stellen sie die für ein massiv-paralleles Prozessorsystem wesentlichen Synchronisationsmechanismen bereit:

- Pop- und Leading-Zero-Count, d.h. die Anzahl der in einem Wort gesetzten Bits bzw. die Anzahl führender Null-Bits. Das Absetzen einer entsprechenden "Store-Operation" bewirkt, daß anstelle des Inhalts des Prozessor-Registers, dessen Pop- und Leading-Zero-Count in das ausgewiesene E-Register geschrieben werden (eine Operation liefert beide Werte). Durch eine anschließende Load-Operation kann dieses Ergebnis in den Prozessor zur Weiterverarbeitung geladen werden.

- Atomare Speicheroperationen, die zwischen lokalem oder entferntem DRAM Speicher und dem spezifizierten lokalen E-Register wirken:
  - Lade Inhalt einer Speicherzelle, wobei diese anschließend erhöht ist (um 1 oder um den vorherigen Inhalt des E-Registers).
  - Tausche den Inhalt einer Speicherzelle mit dem des E-Registers.
  - Vergleiche den Inhalt eines E-Registers mit dem einer Speicherzelle und tausche gegebenenfalls den Inhalt eines zweiten E-Registers mit dem der Speicherzelle.
  - Lade den Inhalt einer Speicherzelle in das E-Register und setze (bzw. lösche) die entsprechenden Bit-Stellen der Speicherzelle, des ursprünglichen E-Register Inhalts.

2.5 Messaging Facility

Das CRAY T3E System enthält eine generelle Message Passing Facility, die im Vergleich zur Messaging Facility des CRAY T3D Systems wesentlich mächtiger geworden ist. So sind beispielsweise Messages nicht mehr auf Cache-Lines beschränkt. Weiterhin können sowohl von Anwender- als Systemseite Message-Queues eingerichtet werden, die auf jeder beliebigen Adresse innerhalb des zulässigen Adreßraums beginnen und jeweils bis zu 64 MBytes lang sein können. Ein Interrupt erfolgt erst bei Überschreiten einer festsetzbaren Schranke, um asynchronen Datentransfer zu ermöglichen.

Messages werden vor der eigentlichen Übertragung in acht E-Registern gesammelt, von wo aus sie in die entfernte Message-Queue geschickt werden (SEND-Operation, Variante einer PUT-Operation). Es wird anschließend eine entsprechende Bestätigung durch Setzen der Status-Bits der beteiligten E-Register zurückgeschickt.

2.6 Interprozessor-Verbindungsnetzwerk

Das Verbindungsnetzwerk des CRAY T3E Systems ist wie im CRAY T3D System als ein 3D-Torus aufgebaut und verbindet alle Knoten für:

- Datenkommunikation
- Barrier/Eureka Synchronisation
- System Konfiguration
- Wartung und Diagnose

Zur Performance Analyse enthält das Netzwerk Monitore.

Die Einbettung sämtlicher Funktionen in ein Netzwerk anstelle zwei separater wie im CRAY T3D erhöht die Flexibilität hinsichtlich Partitionierbarkeit als auch Aufrüstbarkeit des Systems. Die Hardware unterstützt nunmehr unmittelbar Partitionen beliebiger Größe.

2.6.1 Netzwerk-Routing

Ein Netzwerk-Router ist ein Crossbar-Switch, der Verbindungen zu einem PE-Port, I/O-Port sowie den sechs Netzwerkkanälen herstellt. Die Netzwerkkanäle verbinden benachbarte Knoten bidirektional in jeder der drei Dimensionsrichtungen. Auf diesem Netz koexistieren drei virtuelle Netzwerke:

- Ein azyklisches Netzwerk, das deterministisches, dimensionsorientiertes Routing durchführt. Ein einzelner initialer, ungeordneter Weg kann zunächst in eine der drei Richtungen +x, +y, +z ausgeführt werden sowie ein finaler ungeordneter in -z. Die zusätzliche Verwendung virtueller Kanäle garantiert, daß keine Deadlocks in diesem Netz auftreten.
- Ein Netzwerk mit adaptivem Routing, bei dem jede Richtung unter der Bedingung zugelassen ist, daß jeder Schritt näher an den Zielknoten führt.
- Ein blockierungsfreies Wartungsnetz.

Die Gesamtheit dieser Methoden gewährleistet ein fehlertolerantes Netzwerk, das als azyklisches virtuelles Netzwerk bezeichnet wird.

Das Routing der Datenpakete erfolgt durch ein Vergleich des Routing-Tags im Paket und einer Knotenadresse, die im Knoten enthalten ist. Im Gegensatz zum CRAY T3D System, das mit Deltas durch fortlaufendes Dekrementieren arbeitet, werden im CRAY T3E absolute Adressen verwendet. In jedem Knoten befindet sich eine Tabelle, die folgende Information enthält:

- Physikalische Adresse des Zielknotens
- Der deterministische Weg zum Zielknoten
- Hinweis, ob adaptives Routing gestattet ist.

Als Verweis in diese Tabelle dient die logische PE Nummer des Zielknotens. Logische PE Nummer ist das, was das System als Ganzes sieht (OS), im Gegensatz zum Anwender, der mit virtuellen PE Nummern arbeitet.

2.6.2 Barrier/Eureka Synchronisation

Eine wichtige Funktion für MIMD-Systeme ist die Barrier-Synchronisation, d.h. alle Prozessoren müssen an einem Synchronisationspunkt angekommen sein. Neben der logischen UND-Verknüpfung (Barrier) ist auch die ODER-Verknüpfung möglich. Letztere, Eureka-Modus genannte Betriebsart, eignet sich beispielsweise bei parallelem Suchen. Sobald einer der Prozessoren das gesuchte Objekt gefunden hat, kann er dies den anderen mitteilen und diese können dann die Suche ihrerseits sofort beenden. Diese Funktionalität ist im CRAY T3D System über ein separates, baumförmig zusammenlaufendes Netzwerk realisiert.

Im CRAY T3E System ist diese Funktionalität erweitert worden. Das Barrier-Netz besteht aus mehreren virtuellen Netzen, die über das allgemeine Kommunikationsnetz laufen, was zu mehr Flexibilität vor allem bei der Partitionierung der Maschine führt. Das virtuelle Synchronisationsnetz verfügt über die höchste Priorität, so daß Barrier/Eureka Synchronisation Vorrang vor Datentransfer hat.

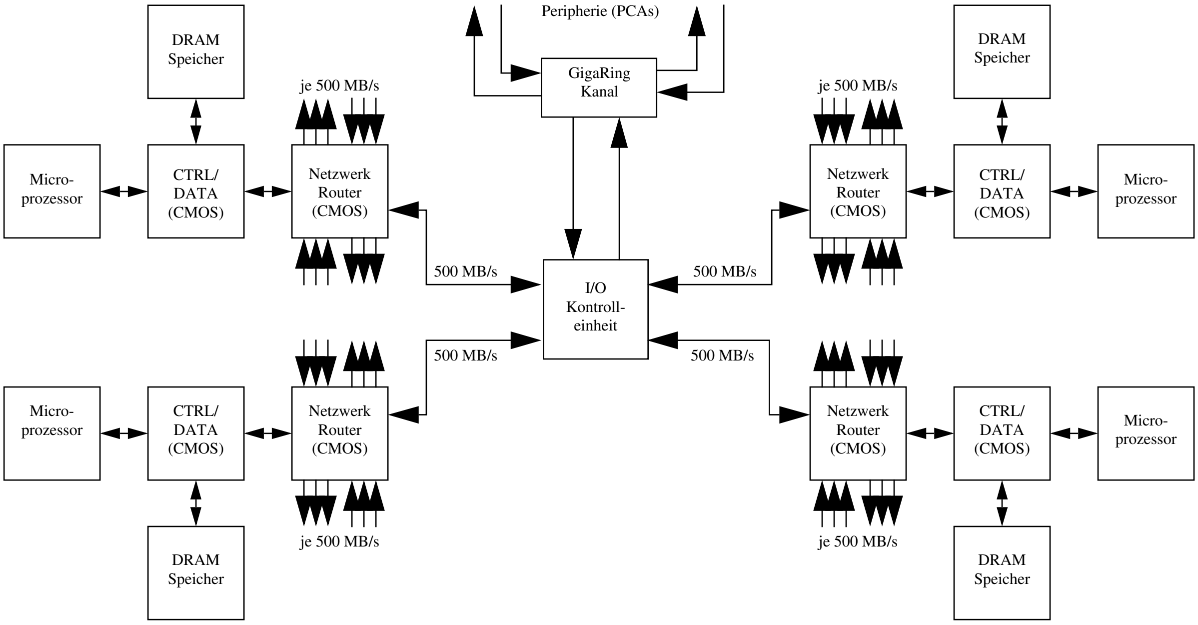
Das Barrier-Netz muß im CRAY T3E System nicht symmetrisch konfiguriert sein. Es werden Knoten im Netz festgelegt, die für einen Teilbaum des Netzes zuständig sind und zunächst auf ihrer Ebene den Barriere/Eureka Zustand sammeln bevor sie ihn an die nächste Ebene weiterleiten. Diese Teilbäume liegen in unmittelbarer Nachbarschaft, wodurch die Netzbelastung minimiert wird.

Jeder Knoten verfügt über 32 Barrier-Kontexte, wovon jeder zwei Barrieren vorsieht, die unabhängig voneinander in Subnetze konfigurierbar sind, auch mit unterschiedlicher Tiefe. Subnetze, die über den gleichen virtuellen Barrier-Kanal laufen, müssen disjunkte Knotenmengen enthalten.

2.7 Ein-/Ausgabe

Im Gegensatz zum CRAY T3D System, enthält das CRAY T3E System deutlich mehr I/O-Knoten. Dadurch wird einerseits eine wesentlich höhere I/O-Bandbreite erreicht, gleichzeitig wird die Gefahr von punktueller Überlastung vermieden.

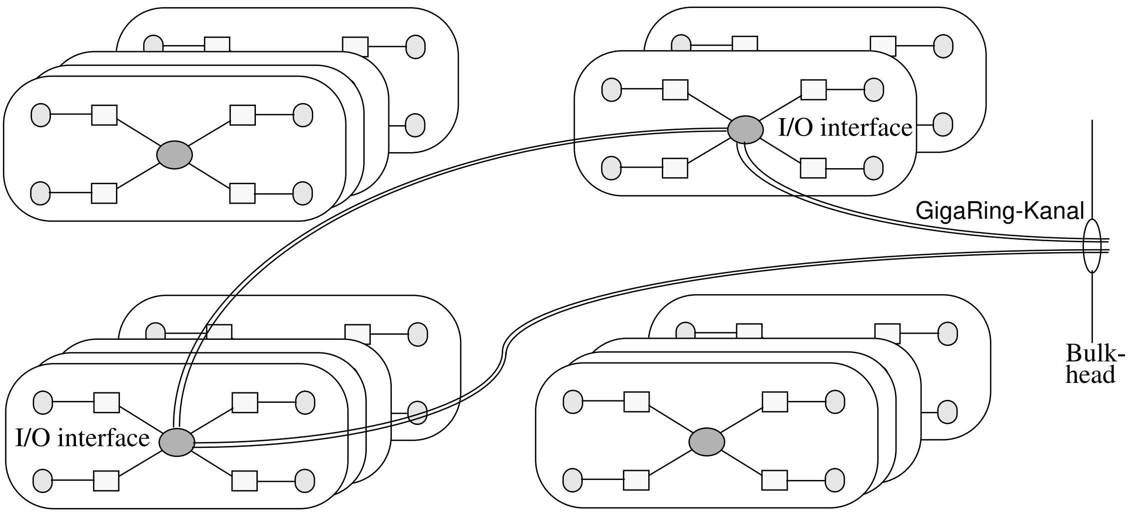
Abbildung 13 Knotenorganisation auf einem Printed Circuit Board



Jedes Printed Circuit Board enthält neben vier PEs eine I/O-Kontrolleinheit, die durch die vier, auf dem gleichen Printed Circuit Board vorhandenen Netzwerk-Router zugänglich ist (vgl. Abbildung 9 und Abbildung 13). Jede I/O-Kontrolleinheit ist im Netz global adressierbar, d.h. von jedem PE aus ansprechbar. Eine Kontrolleinheit im lokalen Bereich zeichnet sich nur durch unmittelbare Nähe, nicht jedoch durch spezielle Funktionalität aus.

Je 2 I/O-Kontrolleinheiten werden zu einem GigaRing-Kanal gekoppelt (Abbildung 14). Die tatsächlich nach außen anschließbare Anzahl an GigaRing-Kanälen (Bulkheads) hängt vom Chassis-Typ ab.

Abbildung 14 I/O-Kontrolleinheiten stellen eine Kopplung in den GigaRing-Kanal her

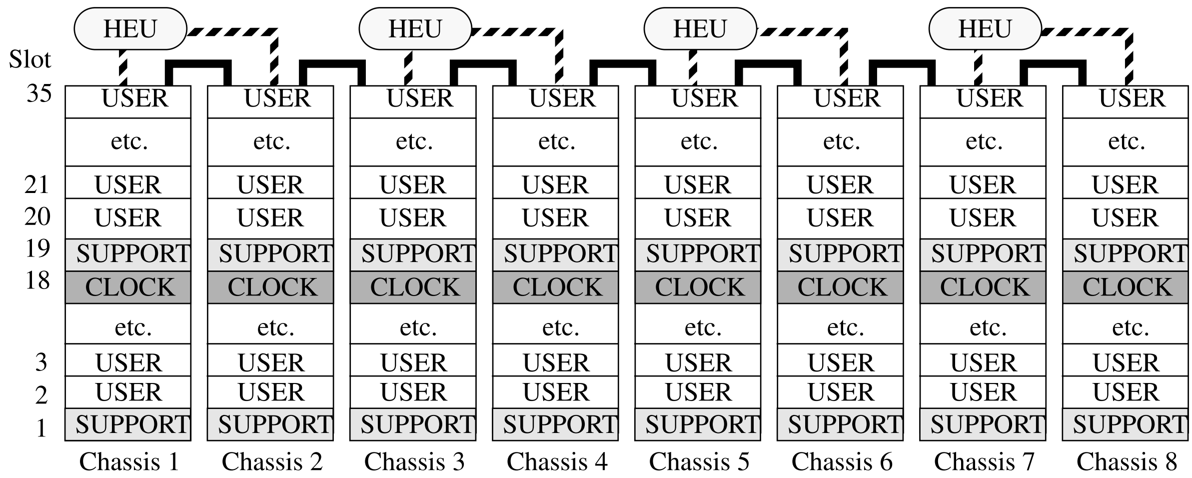


2.8 CRAY T3E Konfigurationen

Es werden rein luftgekühlte (kein Fluorinert Primärkreislauf wie im CRAY T3D MCA System) sowie flüssiggekühlte (Fluorinert für Primärkreislauf, Wasser für Sekundärkreislauf mit Heat Exchange Unit HEU) Systeme angeboten. Die wesentlichen Eckdaten sind:

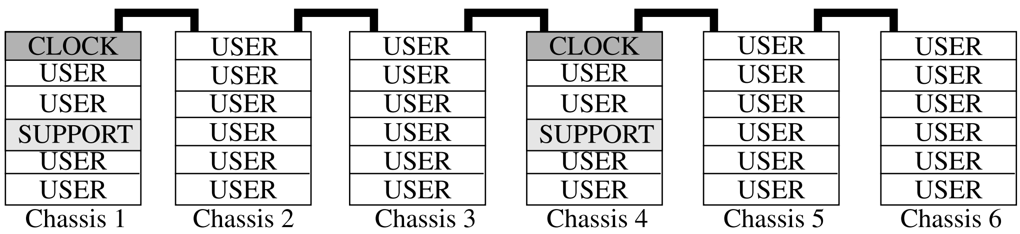
- Flüssiggekühlte Systeme (64 bis 2048 Anwender PEs):
  - 2 Printed Circuit Boards (je 4 PEs und 1 I/O-Kontrolleinheit) pro Modul
  - Aufrüstbarkeit in Schritten von einem Modul (8 PEs)
  - 1 bis 8 Chassis der jeweiligen Größe (in cm) von ca. B x T x H = 120 x 180 x 180
  - je Chassis bis zu 256 Anwender-PEs (USER), 8 oder 16 System-PEs (SUPPORT), 1 Clock-Modul sowie 16 GigaRing-Kanäle

Abbildung 15 Maximale Ausbaustufe (2048 User-PEs) der flüssiggekühlten CRAY T3E System



- Luftgekühlte Systeme (16 bis 128 Anwender-PEs):
  - 1 Printed Circuit Board (4 PEs und 1 I/O-Kontrolleinheit) pro Modul
  - Aufrüstbarkeit in Schritten von einem Modul (4 PEs)
  - 1 bis 6 Chassis der jeweiligen Größe (in cm) von ca. B x T x H = 60 x 150 x 175
  - je Chassis 20 (plus 1 Clock-Modul) oder 24 PEs sowie 3 GigaRing-Kanäle

Abbildung 16 Maximale Ausbaustufe (128 User-PEs) der luftgekühlten CRAY T3E Systeme



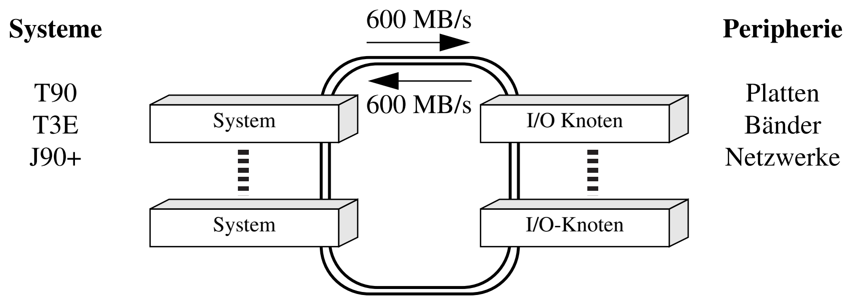
3 GigaRing I/O-Subsystem

Die Verbreiterung der Produktpalette der Cray Research Systeme hat dazu geführt, die Anbindung peripherer Geräte neu zu gestalten, um die verschiedenen Systeme einfach und eng miteinander zu koppeln. Bislang hatte jedes System zunächst sein eigenes, zwischen den Systemen meist inkompatibles I/O-Subsystem. Verbindungen zwischen den Systemen waren nur über Standard-Netzprotokolle wie beispielsweise FDDI oder HIPPI möglich. Weitere Forderungen für ein neues Verbindungskonzept sind einfache und preiswerte Skalierbarkeit, sehr hohe Übertragungsbandbreiten sowie Ausfallsicherheit.

3.1 Grundkonzept

Das Grundkonzept des neuen I/O-Subsystems ist eine Ring-Topologie, die auf dem IEEE SCI-Standard (Scalable Coherent Interface) basiert und mit GigaRing bezeichnet wird. Die vorgenommenen Abweichungen dienen der Verbesserung der Übertragungsraten sowie der Konfigurationsflexibilität zur Erhöhung der Ausfallsicherheit. Wie Abbildung 17 zeigt, können unterschiedliche Cray Research Systeme in den Ring eingebunden werden und periphere Geräte über einen I/O-Knoten ansprechen. Der GigaRing ist bidirektional ausgelegt und hat eine Bandbreite von 600 MB/s in jeder Richtung. Es können sowohl mehrere unabhängige Ringe als auch Teilringe gebildet werden

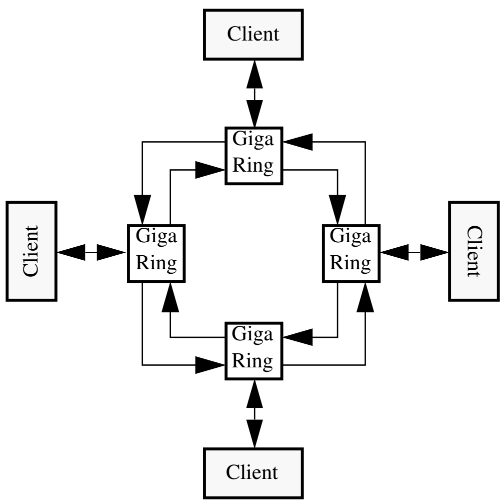
Abbildung 17 Prinzipielle System- und Peripherie-Einbindung



3.2 Aufbau der Kanalverbindungen

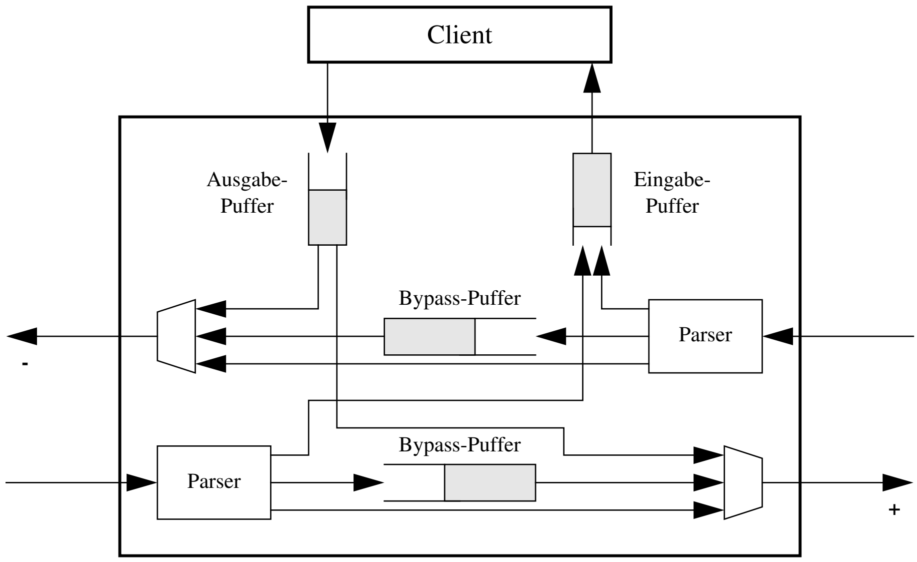
Sowohl Systeme als auch I/O-Knoten stellen in dem Ringkonzept Clients dar, die sich gegenseitig Datenpakete zuschicken. Abbildung 18 zeigt einen Ring mit vier Clients.

Abbildung 18 GigaRing Kanalverbindung



Zur Anpassung der unterschiedlichen Sende- und Empfangsgeschwindigkeiten, enthält ein GigaRing Knoten Ein- und Ausgabe-Puffer. Der GigaRing Knoten prüft im Parser beim Eintreffen eines Pakets die Header-Information und stellt sie in einen Eingabe-Puffer, falls sie für den zugehörigen Client bestimmt ist, andernfalls wird das Paket einfach weitergereicht. Daten, die der Client in den Ring schikken will werden entsprechend in einen Ausgabe-Puffer eingereiht (Abbildung 19).

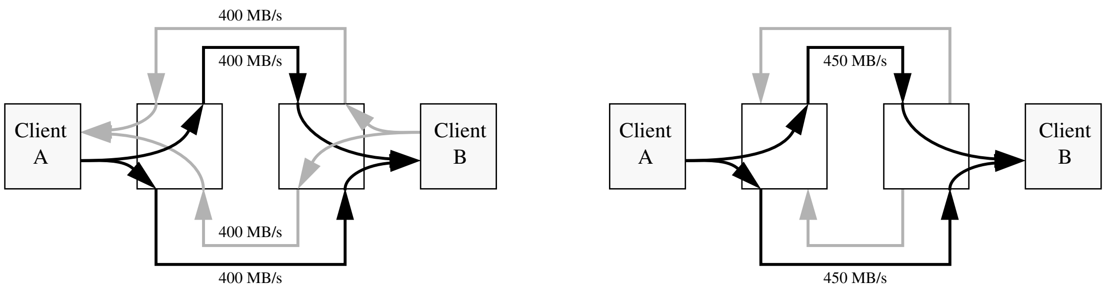
Abbildung 19 Prinzipieller Aufbau eines GigaRing Knotens



Protokolle verhandeln den fairen Zugriff zum Ring. Virtuelle Kanäle ermöglichen die Entkopplung von Strömen und gewährleisten, daß keine Deadlock-Situationen entstehen.

Abbildung 19 verdeutlicht außerdem, daß ein Client gleichzeitig sowohl in + als auch in - Richtung senden bzw. aus beiden Richtungen empfangen kann. Dadurch ergibt sich eine theoretische Bandbreite von 2400 MB/s (reine Hardware-Leistung) innerhalb eines Rings. Durch Protokoll-Overhead reduziert sich die erreichbare, auf Nutzdaten (Payload) bezogene Leistung auf 1600 MB/s (2 x 800 MB/s) im voll-duplex Modus (Abbildung 20 - links), bzw. 900 MB/s im halb-duplex Modus (Abbildung 20 - rechts).

Abbildung 20 Voll-duplex bzw. halb-duplex Betrieb am Beispiel zweier Clients A und B



3.3 Zugriffssicherheit

Ein weiterer Aspekt des Konzepts ist die Zugriffssicherheit. Ein Knoten vertraut entweder prinzipiell allen anderen im Ring oder nicht. Entsprechend dieser Einstellung verfügt jeder Knoten über eine Tabelle, in der vertrauenswürdige (bzw. nicht vertrauenswürdige) Knoten eingetragen sind. Pakete von nicht vertrauenswürdigen Knoten werden vollständig ignoriert.

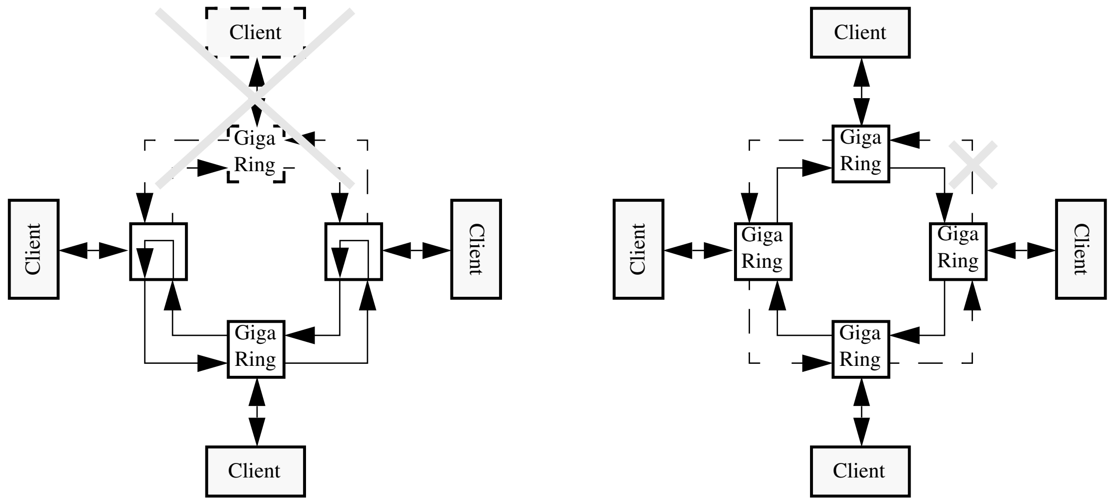
3.4 Ausfallsicherheit

Das bidirektionale Ringkonzept gewährleistet eine sehr hohe Ausfallsicherheit. Die beiden prinzipiellen Möglichkeiten sind (Abbildung 21):

- Ring-Faltung, wo bei Ausfall eines Knotens die beiden Nachbarn ihre Verbindungen intern kurzschließen und umlenken. Es bleibt die volle Bandbreite erhalten.
- Ring-Maskierung, bei der eine Ringrichtung nicht weiter verwendet wird. Es bleibt die volle Funktionalität erhalten, die Bandbreite halbiert sich.

Diese Mechanismen erfolgen durch Software und lassen sich im laufenden Betrieb durchführen. Defekte GigaRing-Verbindungen lassen sich nach der Isolierung unmittelbar austauschen (hot swap).

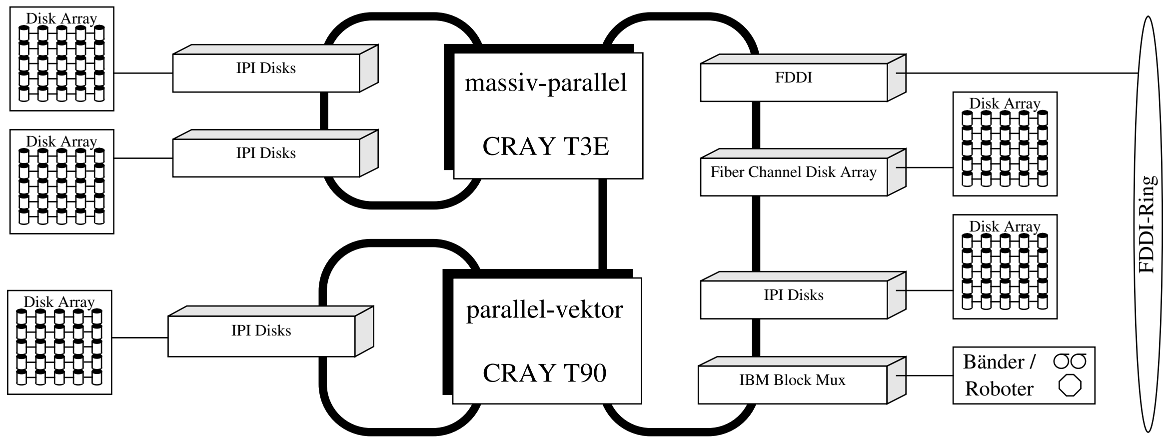
Abbildung 21 Ring-Faltung (links) und Maskierung (rechts)



3.5 Konfigurationen

Neue Produkte, wie das massiv-parallele Prozessorsystem CRAY T3E oder das Parallel-Vektorsystem CRAY T90 werden mit entsprechenden I/O-Knoten ausgerüstet. Abbildung 22 verdeutlicht die gewonnene Flexibilität.

Abbildung 22 Beispiel für flexible Kopplung unterschiedlicher Systeme



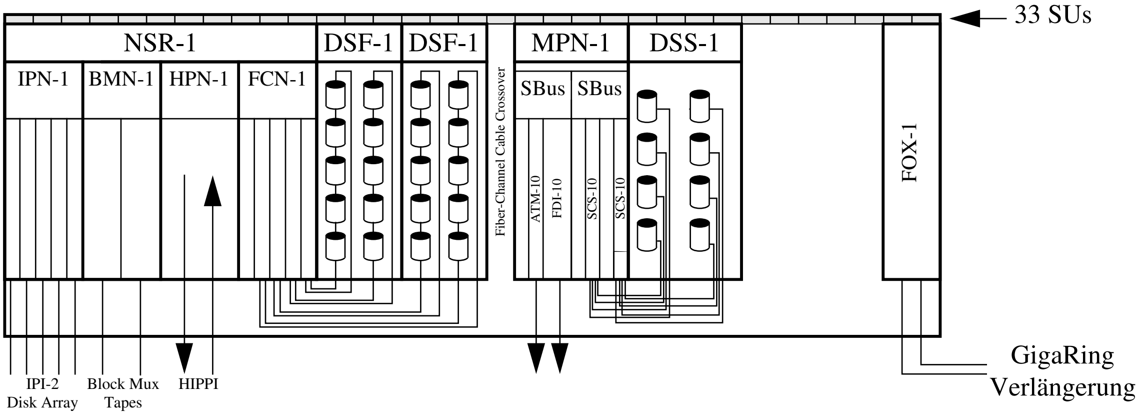
Die Unterbringung der I/O-Knoten für Kanal-Adapter, Fiber-Channel Disks sowie Platten mit SCSI-Schnittstelle erfolgt in Standard-Gehäusen. Ein Gehäuse bietet Platz für 33 SUs (Space Unit, 1 SU = 1.75"), die entsprechend aufgefüllt werden können. Die verwendete CMOS-Technologie erfordert einen geringen Strombedarf, die Kühlung erfolgt ausschließlich durch Luft. Für Entfernungen bis zu 11m werden Kupferkabel verwendet, über einen speziellen Adapter können Glasfaserkabel angeschlossen werden, die eine Entfernungen bis zu 200m zulassen.

Tabelle 1 faßt die verfügbaren I/O-Knoten sowie notwendigen Einschübe (Sub-Racks) zusammen und in Abbildung 23 ist eine Beispielkonfiguration dargestellt.

Tabelle 1 Gehäuse und Adaptertypen

Bezeichnung	Platzbedarf	Funktion
PC-10	HxBxT = 1.8x0.7x1.0 m	Peripheral Cabinet mit Stromversorgung und 33 SUs
NSR-1	11 SUs in PC-10	I/O-Node Subrack, bietet Platz für 4 I/O-Knoten
IPN-1	1 Steckplatz in NSR-1	I/O-Node zum Anschluß von IPI-2 Platten, entweder 5 separate Kanäle oder 1 RAID-3 Disk Array
BMN-1	1 Steckplatz in NSR-1	2 Kanäle für Block-Mux Bandgeräte oder -roboter; 1.5MB/s High-Speed oder 4.5 MB/s Streaming Mode
ESN-1	1 Steckplatz in NSR-1	4 ESCON Kanäle zu je 17MB/s
HPN-1	1 Steckplatz in NSR-1	2 100 MB/s HIPPI-Kanäle (Input und Output) - 32 Bit
HPN-2	1 Steckplatz in NSR-1	1 200 MB/s HIPPI-Kanal (Input und Output) - 64 Bit
MPN-1	4 SUs in PC-10	Multi-Purpose I/O-Node mit 2 SBus (je 30 MB/s bei 32Bit, 45 MB/s bei 64 Bit) mit je 4 SBus-Slots
FDI-10	2 SBus-Slots in MPN-1	FDDI-Anschluß; 9 MB/s
ATM-10	1 SBus-Slot in MPN-1	OC3 ATM-Anschluß; 17 MB/s
ETN-10	1 SBus-Slot in MPN-1	Ethernet-Anschluß; 1 MB/s
SCS-10	1 SBus-Slot in MPN-1	SCSI-2 F/W Schnittstelle; 17 MB/s
DSF-1	3 SUs in PC-10 plus 1 SU pro PC-10 für Kabel	Disk-Subrack für 10 Fiber-Channel Disks (DD-308) sowie deren Stromversorgung
FCN-1	1 Steckplatz in NSR-1	I/O-Node zum Anschluß von 5 Fiber-Channel Loops 100 MB/s Transferrate pro Channel Loop; unterstützt RAID-3 oder RAID-5 Disk-Arrays, die seriell entlang des Loops konfiguriert sind
DD-308	1 Platz in DSF-1	Fiber-Channel Disk; 9.4 GB Kapazität; 8-12 MB/s
DSS-1	4 SUs in PC-10	Disk-Subrack für 8 SCSI-2 Platten (DD-314, DD-318) sowie deren Stromversorgung
DD-314	1 Platz in DSS-1	SCSI-2 Platte; 4.3 GB Kapazität; 4.8-7.4 MB/s
DD-318	1 Platz in DSS-1	SCSI-2 Platte; 9.4 GB Kapazität; 8.0-12.0 MB/s
FOX-1	2 SUs in PC-10	Fiber-Optic Extender zur Verlängerung des GigaRing auf 200 m; wird auf beiden Seiten benötigt

Abbildung 23 Beispielkonfiguration in einem PC-10 Gehäuse

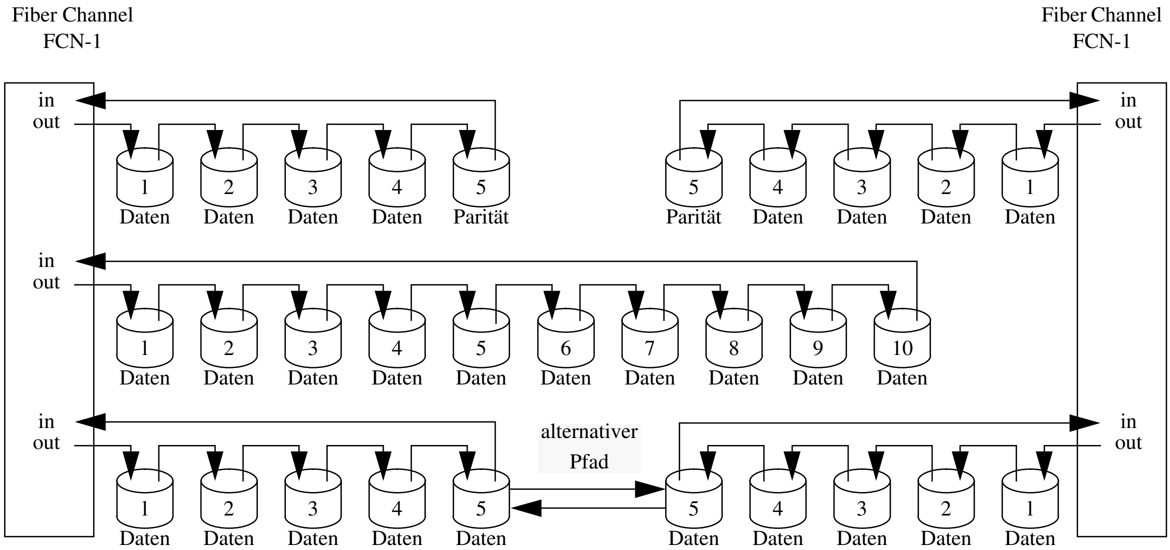


3.6 DD-308 Fiber Channel Platten und Disk Array

Die Fiber Channel Platten sind zur Zeit in der Entwicklung und werden für das GigaRing I/O-Subsystem zur Verfügung stehen. Die Kapazität eines Laufwerks beträgt 8 GB, die Transferrate beträgt bis zu 10 MB/s. Die Laufwerke werden über eine Serial-Channel-Loop mit 100 MB/s an den FCN-1 angeschlossen. Ein FCN-1 unterstützt bis zu 5 Channel-Loops. An einer Channel-Loop lassen sich bis zu 40 Laufwerke hintereinander anschließen. Die Laufwerke besitzen einen Datenpuffer und können aufgrund der hohen Geschwindigkeit der Channel-Loop quasi-simultan übertragen.

Abbildung 24 zeigt die Schaltung als RAID-3 oder RAID-5 Disk Array (mit fünftem Laufwerk für Paritätskontrolle), als Einzellaufwerke sowie die Möglichkeit der Schaltung alternativer Pfade zur Erhöhung der Ausfallsicherheit.

Abbildung 24 Fiber Channel Disk Array



4 Betriebssystem UNICOS/mk

Das Cray Research Betriebssystem UNICOS basiert auf UNIX System V von UNIX System Laboratories (USL) und stellt eine ausgereifte und portable Software-Umgebung für den interaktiven Betrieb und für den lokalen sowie remote Batch-Zugang zur Verfügung. 1985 eingeführt, stellte UNICOS das erste UNIX basierte Betriebssystem für Supercomputer dar. UNICOS enthält Erweiterungen aus Fourth Berkeley Software Distribution (4.3 BSD) sowie die Investition hunderter Mannjahre in Verbesserungen, um optimale Leistung und Funktionalität in einer Supercomputer Produktionsumgebung zu erlangen. UNICOS erfüllt vollständig den X/Open XPG4 Base 95 Profile, einer Obermenge der POSIX 1003.1 und POSIX 1003.2 Standards.

UNICOS ist ein Betriebssystem, das die Vorteile von UNIX, wie beispielsweise Portabilität sowie einer mächtigen Benutzeroberfläche vereinigt mit produktionsorientierten Notwendigkeiten wie Höchstleistung-I/O, Multiprozessor-Unterstützung, ANSI/IBM Magnetbandverarbeitung, Checkpoint-Restart, Betriebsmittelkontrollen, Accounting, Job-Scheduling und Batch-Verarbeitung. Weiterhin wurde besonderer Wert auf Datensicherheit, Zugriffsschutz sowie Netzwerkintegration gelegt.

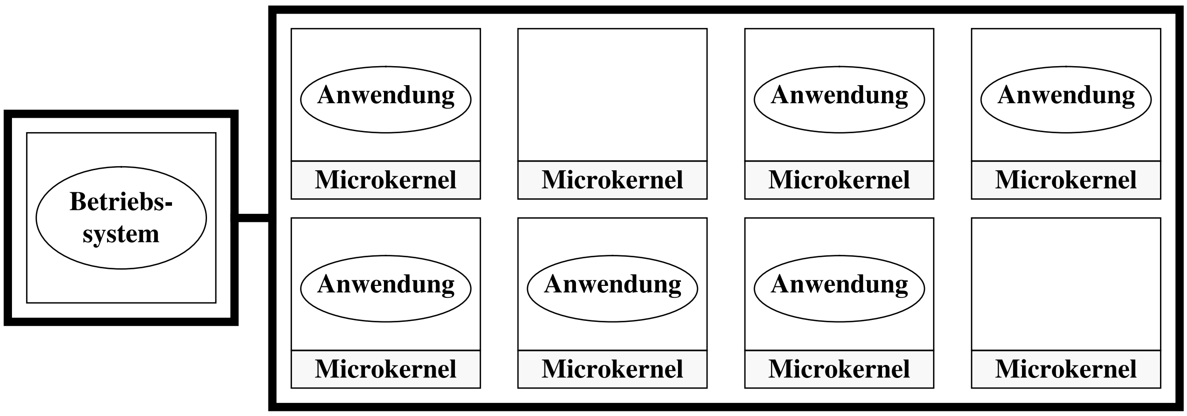
4.1 MPP-Betriebssystemmodelle

Die UNICOS Funktionalität und Stabilität wird auch für die Cray Research MPP Systeme gefordert. Die hohe Prozessorzahl der MPP-Architektur sowie der verteilte Speicher stellen jedoch eine grundlegend andere Plattform als die moderat parallelen Vektorprozessorsysteme dar. Für den Einsatz auf den Cray Research MPP-Systemen wurden drei unterschiedliche Betriebssystemmodelle analysiert, die im folgenden kurz skizziert werden.

4.1.1 Attached Prozessor Modell

Das Attached Prozessor Modell besteht aus einem separaten Rechnersystem oder einem dedizierten Knoten innerhalb des MPP-Systems auf dem alle Betriebssystemdienste für die PEs abgehandelt werden. Die PEs selbst verfügen nur über einen rudimentären Microkernel, der die Schnittstelle zwischen der Anwendung auf dem PE und dem zentralen Betriebssystem bildet (Abbildung 25).

Abbildung 25 Attached Prozessor Modell (separates System)



Die wesentlichen Vorteile eines solchen Modells sind:

- Unmittelbar einheitliche Sicht (Single System Image) aller Ressourcen
- Einfaches Prozeßmodell mit nur einem Prozeßraum

- Einfache Systemstruktur, die stabil und leicht wartbar sein sollte
- Einfaches Systemmanagement vor allem in einer heterogenen Umgebung (z.B. nur eine IP-Adresse)
- Gute Umgebung für rechenintensive Anwendungen mit nur wenig Systeminteraktion und großen I/O-Anforderungen

Die wesentlichen Nachteile sind:

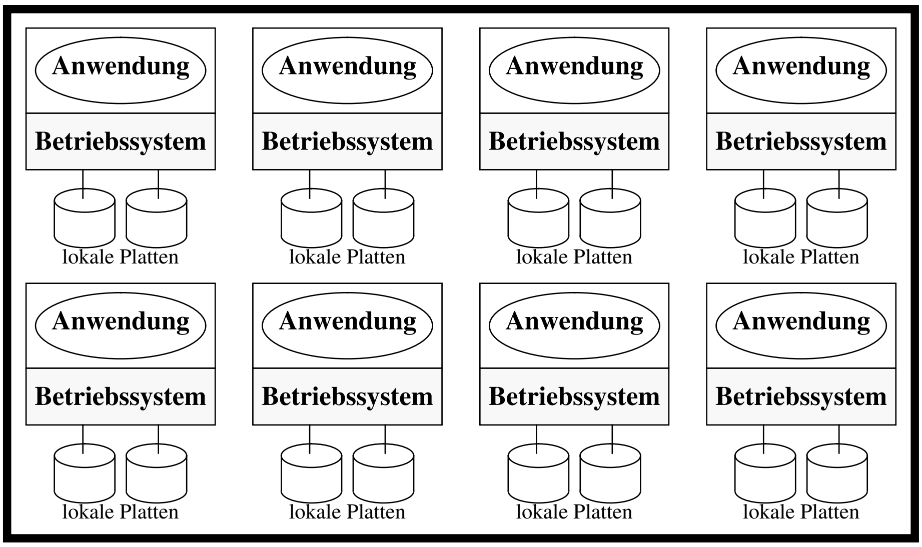
- Skalierbarkeit hängt in starkem Maße von den Anforderungen der PEs sowie der Leistungsfähigkeit des Betriebssystemrechners ab; der Betriebssystem-Server wird schnell zum Engpaß
- PE-lokaler I/O (`/tmp` Stil) ist relativ langsam, vor allem bei kleinen Anforderungen
- Nur dedizierter Betrieb der PEs, da Multiprogramming vom lokalen Microkernel in der Regel nicht unterstützt wird
- Die Abhängigkeit von einem Betriebssystemrechner stellt einen single Point of Failure dar

Ein solches Modell wird in verschiedenen MPP-Systemen verwendet, so auch im CRAY T3D. Beim CRAY T3D sind einige der oben dargestellten Nachteile nicht so gravierend, da der Server ein leistungsfähiges CRAY Y-MP oder CRAY C90 System mit starken I/O-Kanälen und gegebenenfalls mehreren Prozessoren ist.

4.1.2 Workstation-Farm Modell

In einer Workstation-Farm, einem Workstation-Cluster mit großer Prozessorzahl und typischerweise einem schnellen Kommunikationsnetz, wird auf jedem PE ein vollständiges Betriebssystem zur Verfügung gestellt. Meist sind an jedem PE lokale Plattenlaufwerke angeschlossen (Abbildung 26).

Abbildung 26 Workstation-Farm Modell



Die wesentlichen Vorteile eines solchen Modells sind:

- Kurze Latenzzeit eines einzelnen Prozesses zum Betriebssystem
- PE-lokaler I/O (`/tmp` Stil) ist schnell, da direkter Weg zur lokalen Platte
- Hohe Ausfallsicherheit des Gesamtsystems (graceful degrade)

Die wesentlichen Nachteile sind:

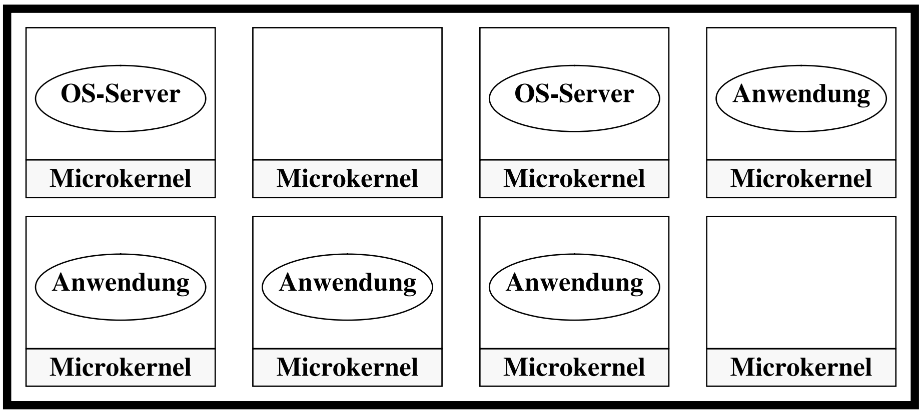
- Hoher Speicherplatzbedarf, da das Betriebssystem repliziert wird
- Schwierig zu administrieren, da jeder Prozessor ein eigenständiges Rechnersystem darstellt
- Schwieriges Prozeß-Management für Anwendungen mit großer PE-Anzahl; führt zu schlechter Skalierung bei mehr als 10 PEs
- Globales Scheduling erfordert zusätzliche Mechanismen
- Kein Single System Image; gemeinsame Filesysteme nur über NFS oder ähnliches

Dieses Modell wird, wie der Name schon impliziert, bevorzugt in MPP-Systemen eingesetzt, die auf Workstation-Plattformen basieren. Der Hauptvorteil für den Hersteller besteht darin, größtenteils auf Vorhandenes zurückgreifen zu können.

4.1.3 Verteiltes Betriebssystemmodell

Beim verteilten Modell werden die Betriebssystemdienste auf mehrere PEs verteilt. Der naheliegendste Weg ist die Dienste in einzelne Server aufzuspalten und diese Server zu verteilen (serverized system). Häufig benötigte Server können gegebenenfalls repliziert werden. Die Kommunikation der PEs mit den Servern erfolgt wie im Attached Prozessor Modell über einen lokalen Microkernel (Abbildung 27).

Abbildung 27 Verteiltes Betriebssystemmodell



Die wesentlichen Vorteile eines solchen Modells sind:

- Emulation des Attached Prozessor Modells (Single System Image)
- Geringer Speicherplatzbedarf, da keine Replikation eines vollständigen Betriebssystems
- Zentrale Administration
- Direkter Zugriff auf alle Filesysteme
- Gute Skalierbarkeit

Die wesentlichen Nachteile sind:

- Sehr schnelle Kommunikations-Hardware mit kurzer Latenzzeit unabdingbar; daher nur auf eng gekoppelten oder Shared Memory Systemen effizient
- PE-lokaler I/O (`/tmp` Stil) nicht so schnell wie im Workstation-Farm Modell

4.2 UNICOS/mk Konzepte

Da das massiv-parallele Prozessorsystem CRAY T3E eigenständig betrieben werden soll, wurde ein verteiltes Betriebssystemmodell als das geeignetste angesehen. Einerseits sind die nötigen Hardware-Voraussetzungen gegeben, andererseits soll das System auch für hunderte von PEs gut skalieren.

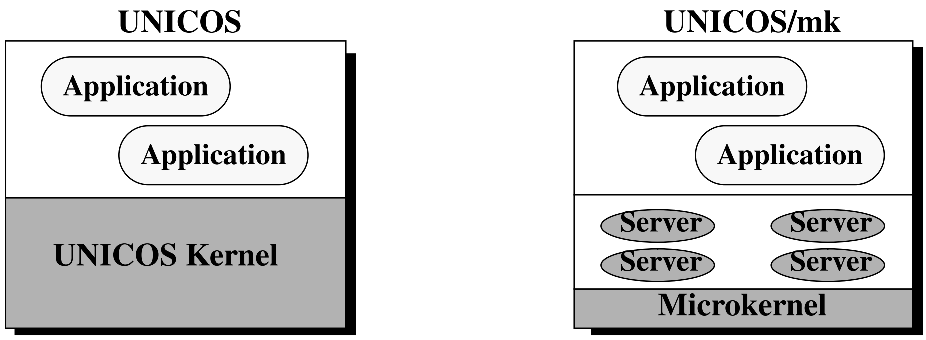
Das neue, auf einer Microkernel Technologie basierende Betriebssystem wird zur Unterscheidung des UNICOS Betriebssystems der Parallel-Vektor-Systeme mit UNICOS/mk bezeichnet und erstmalig auf dem massiv-parallelen Prozessorsystem CRAY T3E eingesetzt.

Für UNICOS/mk wurden folgende Entwicklungsziele gesetzt:

- Einheitliche Sicht des Gesamtsystem (Single System Image)
  - Erhaltung der derzeitigen UNICOS Benutzerschnittstelle (z.B. Login, Prozeßverhalten)
  - Einheitliche Schnittstelle aus Sicht einer Anwendung (System Call Ebene)
  - Einheitliche Sicht für den Systemadministrator (z.B. Resourcen-Vergabe, Accounting)
- Wiederverwendung des vorhandenen UNICOS Kernel Codes
- Erhaltung der Leistungsfähigkeit von UNICOS
- Verbesserung der Zuverlässigkeit und Wartbarkeit

Ein kompakter Microkernel läuft auf jedem Knoten und implementiert nur die wichtigsten Basisfunktionen wie Speicherverwaltung, Interprozeßkommunikation, Basis-I/O (File System Assistant) und die Hardware-abhängigen Funktionen. Alle anderen Betriebssystemfunktionen sind modular auf sogenannte Server verteilt. Durch die Umstrukturierung wird aus einem monolithischen, wenn auch durch Multithreading intern parallelen UNICOS System ein modulares und skalierbares System UNICOS/mk. Abbildung 28 stellt die Abstraktionsebenen beider Systeme gegenüber.

Abbildung 28 Vergleich UNICOS zu UNICOS/mk



4.3 Das CHORUS Modell

Cray Research hat sich nach der Untersuchung mehrerer Alternativen für den Microkernel von CHORUS Systemés entschieden. In einer Prototypportierung wurde sowohl die Machbarkeit untersucht als auch die Leistungsfähigkeit des CHORUS Microkernels verifiziert.

Der CHORUS Microkernel stellt die grundlegende Hardware-Abstraktion, die Speicherverwaltung, das Thread-Scheduling und die Interprozeßkommunikation zur Verfügung und implementiert die folgenden Konzepte:

- Actor-Element für Resourcen-Vergabe
- Thread-Element für CPU-Scheduling
- Port-Zugriff auf Interprozeßkommunikation
- Region-Speicherbereich im Adreßraum eines Actors

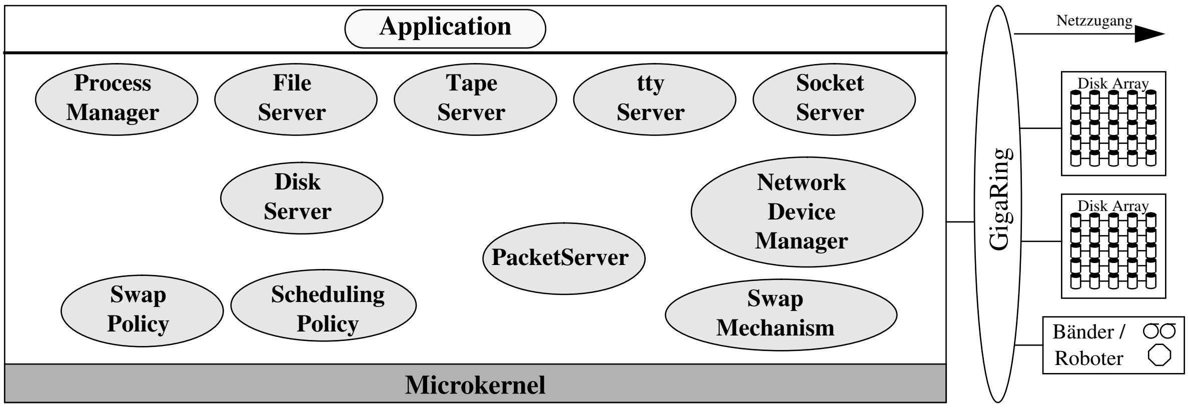
Aufbauend auf diesen Konzepten wird ein UNICOS Prozeß als ein Actor mit mehrfachen Threads und mehreren Regions implementiert. Ein Multitasking-Prozeß besitzt mehrere Threads. Die restliche UNICOS Funktionalität wird durch eine Reihe von Servern zur Verfügung gestellt. Diese Server sind aus der Sicht des Microkernels Actors, die Threads, Regions und Ports enthalten.

Server können entweder im Adreßraum des Anwenders (User Space) oder des Supervisors (Kernel Space) ablaufen. User Space bietet besseren Schutz der Server und damit mehr Sicherheit, Kernel Space bietet höhere Leistung.

4.4 UNICOS/mk Server

Um existierenden UNICOS Code in einem Server zu verwenden, ist eine Anpassung (Wrapper-Technologie in CHORUS) erforderlich, der die Server-spezifischen Interprozeßnachrichten analysiert und eine UNICOS Kernel ähnliche Umgebung aufbaut.

Abbildung 29 Einige Komponenten des Betriebssytems UNICOS/mk



Die Kommunikation zwischen den Servern erfolgt durch Remote Procedure Calls (RPC). Es gibt zwei Arten von RPC, Full Weight RPC (FWRPC) zwischen Servern in verschiedenen Adreßräumen und Light Weight RPC (LWRPC) zwischen Servern im gleichen Adreßraum. FWRPC erfordert Kontext-Switching und ein Kopieren der Daten. LWRPC erlaubt die Weitergabe von Pointern und ist daher effizienter.

Von einem Server können je nach Bedarf mehrere Kopien (z.B. mehrere File-Server für unterschiedliche Filesysteme) aktiv sein. Das Betriebssystem läßt sich so auf mehrere Knoten verteilen und den Anforderungen gemäß skalieren.

4.5 UNICOS/mk Architektur

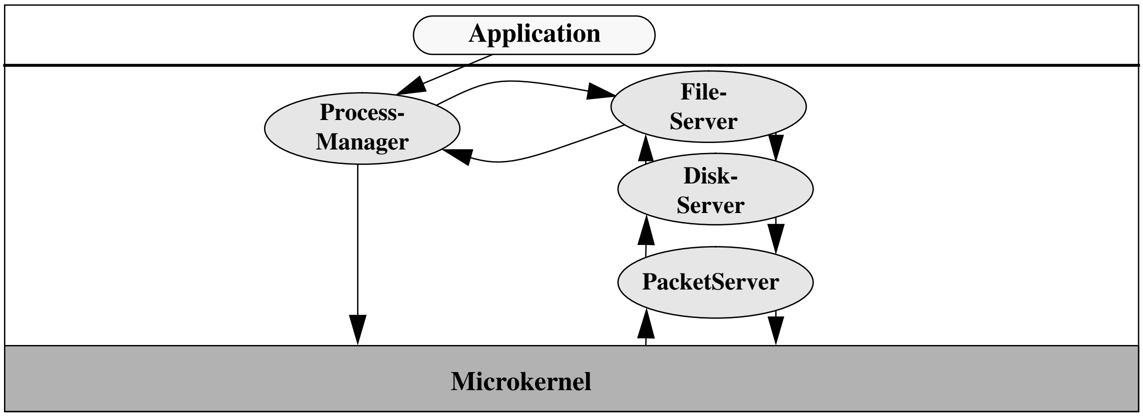
Der UNICOS/mk Microkernel ist im wesentlichen der original CHORUS Microkernel und ist in C++ geschrieben. Statt virtueller Speicherverwaltung und Paging wird im UNICOS/mk Swapping implementiert.

Der UNICOS/mk Process-Manager basiert auf dem CHORUS Process-Manager, ist jedoch stark modifiziert. Der Process-Manager stellt das UNICOS System Call Interface zur Verfügung, implementiert die Prozeß-bezogenen Funktionen (fork, exec, etc.) und leitet I/O-Aufrufe und File-bezogene Aufrufe (open, read, etc.) an den File-Server weiter.

Abbildung 30 zeigt den Ablauf eines `read` Systemaufrufs, wobei die einzelnen Server auf verschiedenen OS-PEs liegen können:

- Der File-Server stellt die File-bezogene Schnittstelle für den Process-Manager und leitet Aufrufe, die sich auf entfernte Files beziehen, an andere File-Server weiter. Der File-Server besteht zum größten Teil aus dem jetzigen UNICOS Code für das NC1 Filesystem.
- Der Disk-Server besteht aus dem Disk-Driver Code aus dem jetzigen UNICOS und führt im Auftrag des File-Servers Disk-Transfers durch.
- Der Packet-Server übernimmt die Kommunikation mit dem I/O- Subsystem. Er sendet und empfängt IOS-Packets.

Abbildung 30 Beispiel für Systemaufruf: read



4.6 UNICOS/mk Single System Image

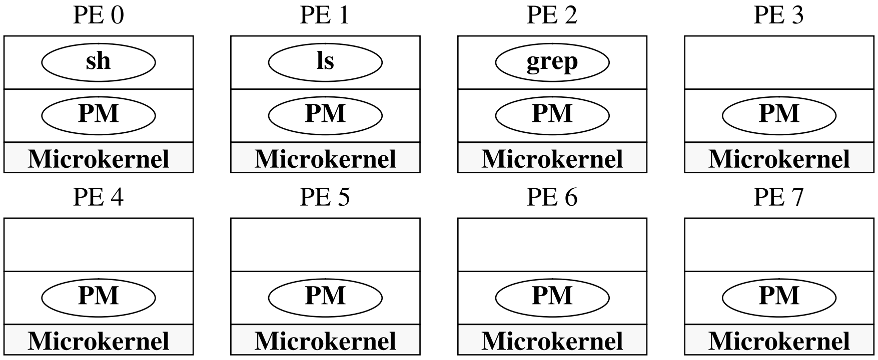
Single System Image bezeichnet die Möglichkeit, ein verteiltes Betriebssystem dem Benutzer, der Anwendung sowie dem Systemverwalter gegenüber wie ein monolithisches System in Erscheinung treten zu lassen. UNICOS/mk ist ein solches System und handelt die Komplexität eines verteilten Systems intern ab. Die folgenden Unterabschnitte erläutern anhand eines Beispiels die jeweilige Sichtweise.

4.6.1 Die Benutzersicht

Beim Einwählen in das System (Login-Prozeß), wird eine Shell gestartet unter der der Benutzer eigene Prozesse (Anwendungen) starten kann. Die Shell sowie alle weiteren Prozesse können dabei auf unterschiedlichen PEs des MPP-Systems ablaufen. Eine parallele Anwendung stellt unter diesem Gesichtspunkt lediglich eine Sonderform dar.

Der einfache Fall eines UNIX-Kommandos der Form `'ls | grep x'` könnte wie in Abbildung 31 dargestellt ablaufen. 'ls' läuft auf einem PE und 'grep' auf einem anderen und die Ausgabe würde auf dem Bildschirm wie bei einem Einzelprozessorsystem erscheinen. Ebenso wäre das weitere Verhalten, z.B. bei einem CTRL-C würden alle beteiligten Prozesse auf den jeweiligen PEs terminiert.

Abbildung 31 Komando: `ls | grep x`

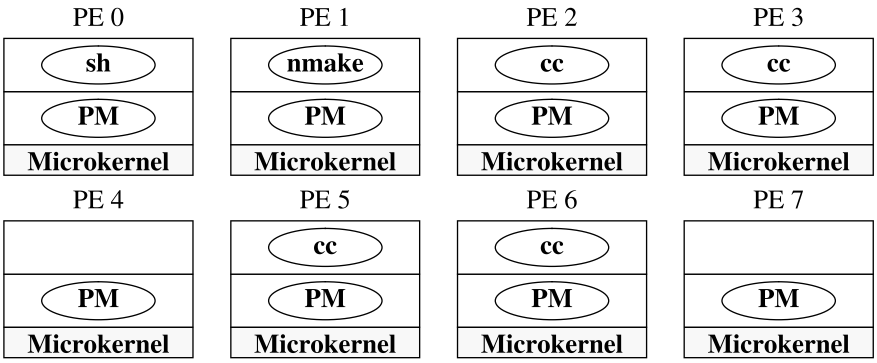


4.6.2 Die Applikationssicht

Aus Sicht einer Anwendung verhält sich das UNICOS/mk ebenfalls wie das monolithische UNICOS System. Wie schon in Abbildung 31 dargestellt, können durch `fork` neue Prozesse erzeugt werden, die z.B. über Pipes miteinander kommunizieren, es kann auf Dateien zugegriffen, Signale können geschickt werden, unabhängig davon wo die Prozesse laufen und wieviele es sind.

Darüberhinaus können Anwendungen auch transparent von der Vielzahl der PEs Gebrauch machen. Als Beispiel sei '`nmake`' genannt, das mehrere im Makefile enthaltene Compilationen parallel startet (Abbildung 32).

Abbildung 32 `nmake` in UNICOS/mk



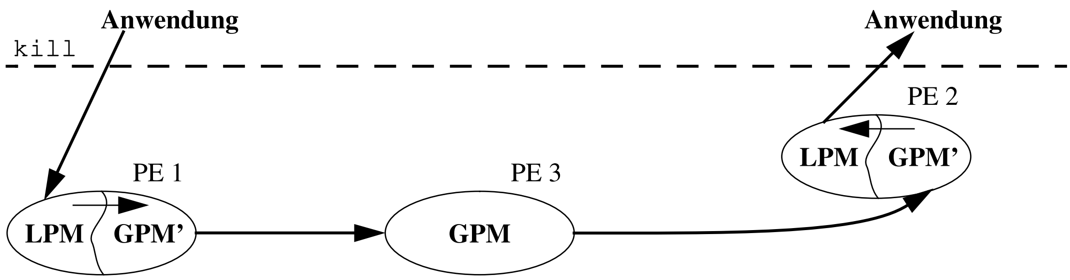
4.6.3 Die Systemsicht

Single System Image dient dazu, dem Anwender und den Anwendungen gegenüber die Sichtweise zu vereinfachen. Es ist die Aufgabe des Betriebssystems die inherente Komplexität zu verschleiern.

Dieses einheitliche Bild wird durch die Prozeß-Manager hergestellt. Ein zentraler 'Global Process Manager' (GPM) behält dabei die Übersicht über das Gesamtsystem. Jeder lokale Process Manager (LPM) verfügt über eine Schnittstelle (GPM') zum globalen Process Manager.

Aus Sicht des Microkernels arbeiten die PEs unabhängig voneinander. So wird beispielsweise ein '`exec`' System-Call vom Prozeß-Manager in einen '`remote exec`' System-Call umgewandelt, d.h. der neue Prozeß wird auf einem anderen PE gestartet. Der lokale Prozeß-Manager behält aber jegliche Information darüber und so würde beispielweise ein späteres Abbrechen dieses erzeugten Prozesses (`kill` Signal) ebnso über die Prozeß-Manager-Ebene abgehandelt (Abbildung 33).

Abbildung 33 Beispiel: `kill` Signal

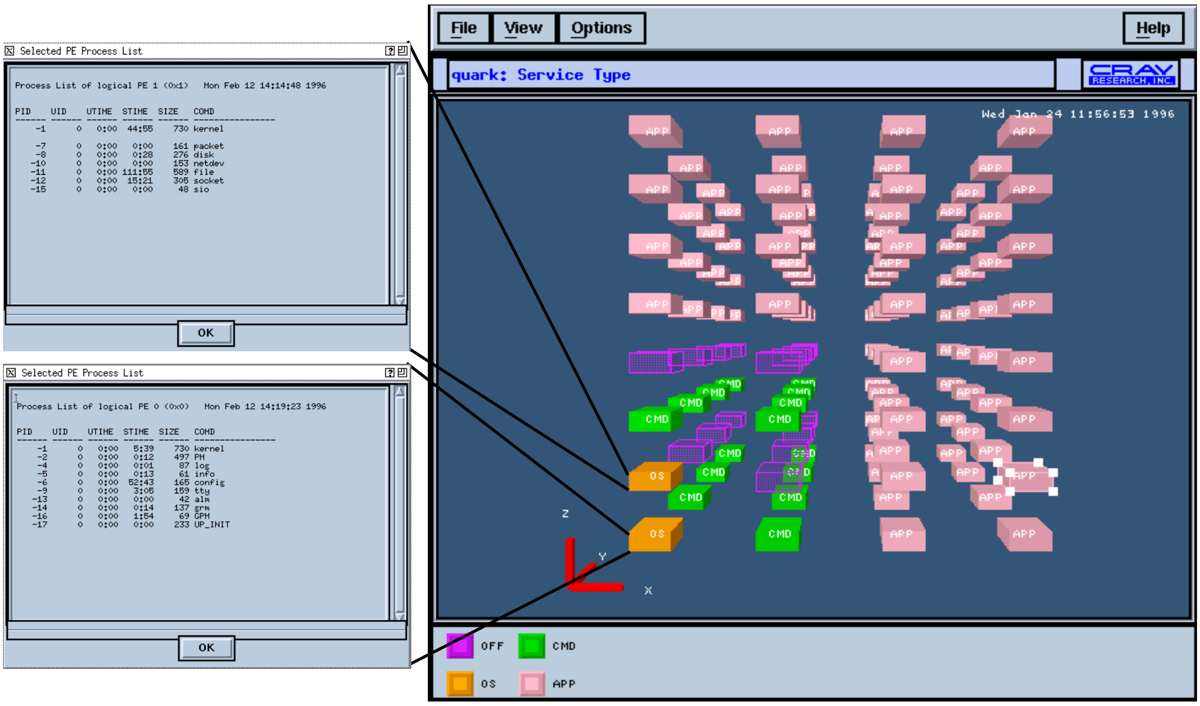


4.7 PE-Klassen des MPP-Systems

Aus Hardware-Sicht gibt es keine Unterscheidung zwischen den tatsächlich vorhandenen PEs. Sie sind allesamt im Torus eingebettet und verfügen in der Regel über identische Prozessoren mit identischem lokalen Speicherau. Die Gesamtheit aller installierter PEs wird aus Sicht von UNICOS/mk zunächst in vier Gruppen (Abbildung 34) aufgeteilt:

- Kommando-PEs zur Ausführung von Benutzer-Kommandos (z.B. Login Shell, Compiler, UNICOS Kommandos) und Single PE Anwendungen
- Anwender-PEs zur Ausführung parallelisierter Anwendungen
- OS-PEs zur Ausführung der Server-Prozesse ohne zusätzliche Benutzer-Kommandos
- Redundante PEs die gleichsam als 'Hot-Spares' im System vorhanden sind, um die Aufgaben ausfallender PEs zu übernehmen.

Abbildung 34 PE-Klassen sowie Einzelaktivitäten der beiden OS-PEs mit `xmppview` Monitor



Das 'Mischungsverhältnis' ist konfigurierbar und wird im wesentlichen von der Größe des Gesamtsystems sowie den Anforderungen bestimmt. Beispielsweise werden für Zeitpunkte mit viel interaktiver Entwicklungsarbeit mehr Kommando-PEs und weniger Anwender-PEs konfiguriert, und zu Zeiten hoher Produktionslast nur ein oder zwei Kommando-PEs bereitgestellt. Im CRAY T3E System werden Kommando-PEs und Anwender-PEs aus dem Pool der 'Compute-PEs', OS-PEs und redundante PEs werden aus dem Pool der 'Support-PEs' genommen.

4.8 UNICOS/mk Scheduling

Die Zuteilung der PEs wird vom globalen Resource-Manager (GRM), einem der UNICOS/mk Server vorgenommen. Ein wesentliches Konzept ist dabei das 'Attribut', einem administrativen Auswahlkriterium, das Präferenzen oder Anforderungen spezifiziert. Folgende drei Attribut-Typen sind vorgesehen:

- Resourcen-Attribut, die Charakteristik eines jeden PEs. Dies beinhaltet Faktoren wie die minimale oder maximale PE Anzahl (vgl. unten), Speichergröße, Taktfrequenz, Bediener-Typ (z.B. batch, interaktiv, rsh), UID und GID. Die Spezifikation eines Resource-Attributs limitiert die Prozesse, die auf dem entsprechenden PE ausgeführt werden können. Dieses Attribut wird vom Systemadministrator vergeben.
- Applikations-Attribut, Anforderungen oder Präferenzen der Anwendung. Dies kann vom Benutzer explizit oder implizit über den Benutzer-Eintrag in der UDB (User Data Base), bzw. über die NQS-Klasse gesetzt werden. Es beinhaltet Faktoren wie Speicherplatzbedarf, Anzahl PEs und Bediener-Typ einer Anwendung.
- Präzedenz-Attribut, das vom globalen Resource-Manager anhand der beiden anderen berechnet wird. Es beinhaltet Faktoren wie Speicherplatz-Zuteilung (best fit), Zusammenhängigkeit sowie optimale PE-Zuteilung innerhalb einer Region.

Ein wesentliches Scheduling-Kriterium betrifft die Anzahl der minimalen bzw. maximalen PE Anzahl. Die Angabe des Minimums bezeichnet die Anzahl der PEs die minimal zugewiesen werden müssen, um den Prozeß ausführbar zu machen, entsprechend darf das Maximum nicht überschritten werden.

Resource-Attribute ermöglichen die Aufteilung der vorhanden PEs in Gruppen. Beispielsweise könnte ein 128 PE System tagsüber in folgende drei Gruppen unterteilt sein:

- PEs 0 - 7: minimale und maximale Anzahl 1; beliebiger Bedien-Typ
- PEs 8 - 31: minimal 2 PEs maximal 8 PEs; beliebiger Bedien-Typ
- PEs 32 - 127: minimal 8 PEs maximal 40 PEs; nur Batch

Wahrend der Nachtschicht könnte dann zu folgender Aufteilung gewechselt werden:

- PE 0 : minimale und maximale Anzahl 1; beliebiger Bedien-Typ nur root
- PEs 1 - 3: minimale und maximale Anzahl 1; beliebiger Bedien-Typ
- PEs 4 - 127: minimal 8 PEs maximal 124 PEs; nur Batch

Ein an das System submittiertes Programm mit einer Anforderung von 4 PEs, würde demnach vom globalen Resource-Manager während der Tag-Attribute der Gruppe 8-31 PEs zugewiesen, die anderen beiden kämen nicht in Frage. Innerhalb dieser Gruppe würden dann die 4 freien PEs mit dem höchsten Gewicht der Präzedenz-Attribute zugewiesen. Während der Nachtschicht würde dieses Beispiel garnicht anlaufen, da keine der Gruppen die geforderten Kriterien erfüllt.

4.9 UNICOS/mk Verfügbarkeit

Da UNICOS/mk ein neues Betriebssystem ist, wird die volle Funktionalität des derzeitigen UNICOS Betriebssystems stufenweise erreicht werden.

Im ersten Release werden alle wesentlichen Funktionen vorhanden sein, um einen Produktionsbetrieb ohne ein Front-End System zu gewährleisten. Dazu gehören beispielsweise alle wesentlichen UNICOS Kommandos, Netzwerkzugang (TCP/IP), NQS-Batch-Subsystem, System-Verwaltung, Accounting und elementares Scheduling (Space-Sharing paralleler Anwendungen, Time-Sharing von Einzel-PE Anwendungen).

Im nächsten Schritt werden vor allem das Scheduling paralleler Anwendungen flexibler (z.B. Prioritäten-Steuerung, Time-Sharing von Multi-PE Anwendungen und Swapping), die System-Verwaltung erweitert sowie die Sicherheit für den Anwender erhöht (Checkpoint/Restart).

5 Cray Research MPP-Programmierumgebung

Die Einsetzbarkeit massiv-paralleler Prozessorsysteme in einer Produktionsumgebung hängt vor allem von geeigneten Software-Oberflächen sowie leistungsfähigen Programmiermodellen ab. Die Software-Umgebung für die Cray Research MPP-Systeme bietet viele Eigenschaften und Möglichkeiten der ausgereiften Software der Cray Research Vektorrechner, die selbst moderat parallele Systeme sind (CRAY T90 bis zu 32 CPUs). Zur Unterstützung der massiv-parallelen Komponente gibt es Erweiterungen im Bereich der Sprachen, Bibliotheken sowie Analysewerkzeuge.

Der Entwicklungsaufwand orientiert sich dabei an einem Spektrum von Supercomputer-Anwendungen, die mit überlegener Leistung auf dem MPP-System ausführbar sein sollen. Ein vorrangiges Ziel dabei ist, diese Umgebung über die bevorstehenden Generationen hinweg aufwärts kompatibel zu halten.

5.1 Programmiersprachen und -modelle

Für technisch-wissenschaftliche Anwendungen ist Fortran weiterhin die wichtigste Programmiersprache, insbesondere da mit dem Fortran 90 Standard bislang vermißte Sprachelemente hinzugekommen sind. Für das CRAY T3D System wird ein Fortran 77 Compiler angeboten, für das CRAY T3E System ein Fortran 90 Compiler im vollen Umfang des Standards.

C sowie C++ Compiler stehen als Single PE Compiler zur Verfügung, die Kommunikation erfolgt über Message-Passing.

5.1.1 Cray Research Adaptive Fortran (CRAFT)

Um MIMD-Parallelität in einem Fortran Programm ausdrücken zu können, werden zusätzliche, über den Sprach-Standard hinausgehende Mechanismen benötigt. Diese reichen von explizitem Message Passing über spezielle Sprachkonstrukte bis zur Steuerung durch Direktiven. Wünschenswert wären einheitliche Erweiterungen, allerdings müssen die Konzepte im Interesse optimaler Leistungsausbeute Kerneigenschaften der Zielarchitektur berücksichtigen. Ein sehr wichtiges Merkmal ist dabei die physische Speicherorganisation sowie die logische Adressierbarkeit des Zielsystems.

Moderat parallele Systeme, wie beispielsweise die CRAY T90 Systeme mit bis zu 32 extrem leistungsfähigen Vektorprozessoren verfügen über eine zentrale Speicherorganisation (Shared Memory) mit globaler Adressierung und werden daher in der Regel nicht mit Message Passing programmiert. Insbesondere der explizite Austausch von Daten ist nicht nötig, da alle Prozessoren auf jedes Datenelement direkt und mit konstanter Latenzzeit zugreifen können.

Cray Research besitzt eine über zehnjährige Erfahrung mit Programmiermodellen auf Shared Memory Systemen. Macro-, Micro- und vor allem Autotasking werden mit großem Erfolg in komplexen Produktionsprogrammen angewandt. Diese Modelle unterstützen MIMD und SPMD Parallelisierung auf Unterprogrammebene sowie Datenparallelismus auf Schleifenebene.

Auf die massiv-parallelen Rechnersysteme wären diese Modelle prinzipiell direkt übertragbar, da es sich ebenfalls um Systeme mit globalem Adreßraum handelt. Aufgrund der hohen Prozessorzahl ist eine zentrale Speicherorganisation mit vertretbarem Aufwand jedoch nicht mehr realisierbar. Als Konsequenz daraus muß ein auf Höchstleistung zielendes Programmiermodell für eine derartige Maschine Möglichkeiten zur Steuerung der Datenverteilung (Distribution) sowie relativen Zuordnung (Alignment) auf die einzelnen lokalen Speicher bieten, da sich die Latenzzeiten beim Zugriff auf den lokalen Speicher gegenüber dem auf einen entfernten Speicher stark unterscheiden.

Das Cray Research Adaptive Fortran (CRAFT) Programmiermodell stellt eine über Direktiven gesteuerte Erweiterung zu Fortran dar (Fortran 77 für CRAY T3D, HPF_CRAFT für CRAY T3E). Es handelt sich um ein Work-Sharing Modell, d.h. alle PEs verfügen über jeweils eine Kopie des Programms und beteiligen sich entweder aktiv an der Problemlösung oder verharren in einem Wartezustand, beispielsweise außerhalb einer parallelen Region oder auf einer Barriere. Die Funktionen zur Datenverteilung wurden teilweise aus dem an der Rice University entwickelten Fortran-D sowie Vienna Fortran übernommen, die Kontrollmechanismen aus dem Cray Research Auto-, Micro- und Macrotasking. Die wesentlichen Konzepte lassen sich folgendermaßen skizzieren:

- Alle Variablen sind standardmäßig privat, d.h. jedes PE hat eine eigene Kopie in seinem lokalen Speicher, andere PEs haben keinen unmittelbaren Zugriff. Globale Variablen werden explizit über Direktiven (`SHARED`) gekennzeichnet und sind nur in einer Instanz vorhanden. Kopien werden nicht explizit angelegt, allerdings wird der Compiler aus Optimierungsgründen entfernte Daten in lokalen Puffern vorhalten, ähnlich wie ein Compiler Daten in Registern hält. Bei Feldern läßt sich dann noch die Verteilung der Elemente auf die lokalen Speicher spezifizieren. Dabei können Aufteilungen in Blöcken, Spalten, Zeilen sowie Strides vorgenommen werden. Die relative Positionierung von Feldern (Alignment) erfolgt implizit, da das jeweils erste Element einer Zerlegung auf dem logischen PE 0 zu liegen kommt. Abbildung 35 zeigt eine Deklaration mit blockweiser (Feld A) und spaltenweiser (Feld B) Verteilung der Elemente auf 4 PEs.

Abbildung 35 Beispiel für eine `SHARED` Deklaration und Verteilung

```
      DIMENSION A(8,8)
CDIR$ SHARED A(:BLOCK,:BLOCK)

 1,1 1,2 1,3 1,4|1,5 1,6 1,7 1,8
 2,1 2,2 2,3 2,4|2,5 2,6 2,7 2,8
 3,1 3,2 3,3 3,4|3,5 3,6 3,7 3,8
 4,1 4,2 4,3 4,4|4,5 4,6 4,7 4,8
 ---------------+---------------
 5,1 5,2 5,3 5,4|5,5 5,6 5,7 5,8
 6,1 6,2 6,3 6,4|6,5 6,6 6,7 6,8
 7,1 7,2 7,3 7,4|7,5 7,6 7,7 7,8
 8,1 8,2 8,3 8,4|8,5 8,6 8,7 8,8
```

```
      DIMENSION B(8,8)
CDIR$ SHARED B(:,:BLOCK)

 1,1 1,2|1,3 1,4|1,5 1,6|1,7 1,8
 2,1 2,2|2,3 2,4|2,5 2,6|2,7 2,8
 3,1 3,2|3,3 3,4|3,5 3,6|3,7 3,8
 4,1 4,2|4,3 4,4|4,5 4,6|4,7 4,8
 5,1 5,2|5,3 5,4|5,5 5,6|5,7 5,8
 6,1 6,2|6,3 6,4|6,5 6,6|6,7 6,8
 7,1 7,2|7,3 7,4|7,5 7,6|7,7 7,8
 8,1 8,2|8,3 8,4|8,5 8,6|8,7 8,8
```

- Die parallele Abarbeitung ist unmittelbarer Bestandteil des Programms, d.h. alle PEs arbeiten gleichzeitig und unabhängig voneinander am gleichen Programm. Regionen, die sequentiell, d.h. auch nur einmal durchlaufen werden dürfen, müssen also solche über die Direktive `MASTER` bzw. Abschluß durch `END MASTER` gekennzeichnet werden.
- Die Parallelisierung auf Schleifenebene wird durch die Direktive `DO SHARED` bewirkt. Dies ermöglicht die Parallelisierung auf kleiner Granularitätsebene aber auch der einfachen Steuerung parallel bearbeitbarer Unterprogramme. Zur Optimierung der Abarbeitung läßt sich steuern, welche Indizes der Schleife auf welchem PE bearbeitet werden sollen. So bewirkt die Direktive `DO SHARED (I,J) ON A(I,J)` die Parallelisierung sowohl über den Index `I` als auch `J` (z.B. geschachtelte Schleifen). Das jeweilige Indexpaar wird auf dem PE abgearbeitet, in dessen lokalen Speicher `A(I,J)` liegt.
- Synchronisationsmechanismen sind einmal am Ende einer parallelen Region (z.B. Schleife) implizit vorhanden, daneben können sie noch explizit spezifiziert werden. Die Direktive `BARRIER` markiert eine Stelle, an der alle PEs angekommen sein müssen, bevor weitergearbeitet werden kann. Für ein Programmiermodell mit globalem Adreßraum ist daneben die Markierung sogenannter kritischer Regionen unerläßlich. Mit `CRITICAL` bzw. `END CRITICAL` werden Code-Sequenzen markiert, die zu einem Zeitpunkt nur von einem PE durchlaufen werden dürfen, um beispielsweise globale Variablen in geordneter Weise zu verändern. In einfachen Fällen ist die Direktive `ATOMIC UP-`

DATE effizienter, d.h. es kann im Prinzip parallel gearbeitet werden, lediglich die Veränderung globaler Variablen bedarf des besonderen Schutzes. In beiden Fällen wird sichergestellt, daß Zugriffskonflikte auf globale Variablen aufgelöst und Cache Coherency zwischen Prozessor und Speicher hergestellt ist.

- Spezielle Intrinsic-Funktionen stellen Grundoperationen zur Verfügung, wie Reduktionsoperationen (z.B. SUM, PRODUCT, MINVAL), Parallel-Prefix (Aufbewahrung aller Zwischenresultate) und Parallel-Scan (maskierbare Parallel-Prefix Funktionen) sowie Funktionen, die Auskunft über die Abbildung von Daten (z.B. BLKKCT, LOWIDX, HIIDX, IS_SHARED) liefern.

Zur Verdeutlichung der Wirkungsweise der Direktiven sei die Bildung eines Skalarprodukts angeführt, obwohl man diese Funktion normalerweise nicht explizit programmieren, sondern auf die entsprechende Intrinsic-Funktion zurückgreifen würde.

Abbildung 36 Mögliche Programmierung eines Skalarprodukts

```
      subroutine sc_prod (a, b, n, s)
      real a(n), b(n), s, s_priv
cdir$ shared a(:block),b(:block),s,n
cdir$ master
      s = 0.0
cdir$ end master
      s_priv = 0.0
cdir$ do shared (i) on a(i)
      do i = 1,n
        s_priv = s_priv + a(i)*b(i)
      end do
cdir$ atomic update
      s = s + s_priv
      return
      end
```

globale Variable s darf nur einmal initialisiert werden

parallele Bearbeitung über Index i auf dem PE mit a(i) lokal; alle Zugriffe erfolgen aus den lokalen Speichern, da b die gleiche Verteilung wie a hat

abschließende Aufsummierung auf s muß geordnet erfolgen

5.1.2 High Performance Fortran (HPF)

Das High Performance Fortran Forum besteht unter der Leitung von K.Kennedy (Rice University) aus Teilnehmern von mehr als 40 Organisationen aus Industrie, Hochschulen und Forschung. Ziel des Forums ist die Erweiterung des Fortran 90 Standards, um Programme mit möglichst hoher Leistung auf MIMD und SIMD Parallelrechnern mit verteilter Speicherorganisation auszuführen.

Die mittlerweile verabschiedete Spezifikation wird für Programmsysteme bevorzugt, die in erster Linie portabel zwischen verschiedenen Plattformen sein sollen.

5.1.3 Vergleich und Implementierung von HPF und CRAFT

Sowohl HPF als auch CRAFT sind Programmiermodelle mit impliziter Parallelisierung und Datenverteilung, allerdings sehr unterschiedlicher Semantik. Abbildung 37 stellt die wesentlichen Aspekte gegenüber.

Cray Research hat eine Vereinbarung mit Portland Group Inc. (PGI) geschlossen, den bei PGI entwikkelten HPF Compiler für alle Cray Research Systeme (massiv-parallel und Parallel-Vektor) zu portieren. Dieser Compiler kann sowohl über PGI als auch Cray Research bezogen werden.

Zusätzlich zur direkten HPF-Implementierung nimmt PGI für das CRAY T3E System eine, mit HPF_CRAFT bezeichnete Implementierung der CRAFT Konstrukte im EXTRINSIC Bereich des HPF Compilers mit auf.

Abbildung 37 Vergleich der wesentlichen Eigenschaften der beiden Programmiermodelle

Cray Research Adaptive Fortran	High Performance Fortran
Sprachelemente	**Sprachelemente**
- Fortran 77 und Fortran 90	- Fortran 90 ist Voraussetzung
+ Automatic Arrays	+ FORALL
+ Array Assignment, WHERE	+ einige Intrinsics
+ Teilmenge Fortran 90 und andere Intrinsics	+ Bibliotheksroutinen
Unterstützung für parallele Ausführung	**Unterstützung für parallele Ausführung**
- Array Anweisungen	- Array Anweisungen
- Fortran 90 Array Intrinsics	- Fortran 90 Array Intrinsics
	- FORALL
- CDIR$ DO SHARED	- !HPF$ INDEPENDENT
- Bibliotheksroutinen	- Bibliotheksroutinen
- Serialisierung durch CDIR$ MASTER	- Serielle Abarbeitung ist Standard
- Synchronisation durch Barrier, Locks, Events und Critical Regions	- Keine expliziten Synchronisationsmechanismen
Ausführungsmodell	**Ausführungsmodell**
- Parallele Ausführung ist Teil des Modells	- Korrekte Ausführung entsprechend der sequentiellen Semantik
- Compiler Direktiven bestimmen die Semantik des Programms	- Compiler Direktiven beeinflussen die Semantik nicht
- Alle Daten sind standardmäßig privat	- Private Daten nicht vorgesehen
- Explizite Deklaration globaler Daten über SHARED Direktiven	- Alle Daten sind standardmäßig global
- Alle Prozessoren führen das Programm aus; gegebenenfalls redundant	- Alle Prozessoren führen jeweils eine Anweisung aus
- Message Passing kann in dem Modell direkt enthalten sein	- Message Passing nur über Ausweichen auf EXTRINSIC Funktionen, in denen alle Daten privat sind
Datenverteilungsmechanismen	**Datenverteilungsmechanismen**
- Relative Positionierung implizit dadurch, daß das erste Element jeweils auf PE 0 liegt	- Relative Positionierung statisch durch ALIGN und dynamisch durch REALIGN Direktiven
- Verteilung auf Unterprogrammebene	- Verteilung auf Unterprogrammebene und innerhalb durch REDISTRIBUTE
-w:BLOCK(m)	-BLOCK
-BLOCK(1)	-CYCLIC
-Kollabiert	-Kollabiert
	-Replizierung
- Kopien lokaler Daten beim Verlassen einer seriellen Region durch CDIR$ END MASTER COPY $var_1,var_2,...,var_n$	- n/a
- Allgemeine Vorlagen durch GEOMETRY	- Allgemeine Vorlagen durch TEMPLATE
- Umverteilung, falls nötig beim Wechsel in ein anderes Unterprogramm	- Umverteilung, falls nötig beim Wechsel in ein anderes Unterprogramm (keine dynamische Änderung mehr möglich)
Prozessortopologie	**Prozessortopologie**
- Kein explizites logisches Modell; PEs werden von 0 bis n-1 durchnummeriert	- Logisches Modell unabhängig von realer Hardware definierbar

5.2 Message Passing Toolkit

Für CRAY T3D sind die nachfolgend aufgeführten Produkte separat verfügbar, MPI wird vom Edinburgh Parallel Computing Centre (EPCC) unterstützt. Der Cray Research Message Passing Toolkit führt hoch optimierte Versionen der Message Passing Bibliotheken PVM und MPI für alle Cray Research Systeme zusammen. Alle Message Passing Konstrukte sind für Fortran und C/C++ verfügbar.

5.2.1 Parallel Virtual Machine (PVM)

PVM ist prinzipiell in heterogenen Umgebungen einsetzbar, im Interesse optimaler Leistungsausbeute bei der Kommunikation innerhalb des MPP-Systems wurden jedoch wesentliche Modifikationen vorgenommen unter gleichzeitiger Beibehaltung der Anwenderschnittstelle. Existierende, mit Hilfe von PVM parallelisierte Programme lassen sich daher leicht auf Cray Research MPP-Systeme portieren.

Die PVM Implementierung erlaubt auch die Kommunikation verteilter Applikationen auf anderen Cray Research Systemen. Von dort können weitere, im Netz befindliche Systeme angesprochen werden.

5.2.2 Message Passing Interface (MPI)

MPI wird zunehmend als ein de facto Message Passing Standard angesehen. Cray Research ist aktiv im MPI Forum beteiligt, das derzeit MPI-2 definiert. MPI-2 wird neben dynamischem Prozeß-Management einseitige Punkt-zu-Punkt Kommunikation enthalten. Letztere stellen eine portable Syntax der Cray Research Shared Memory Routinen dar.

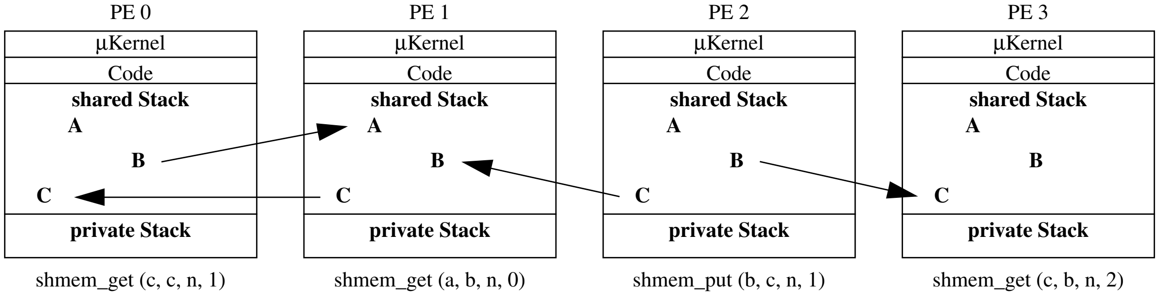
5.2.3 Shared Memory Routinen

Weiterhin sind Routinen verfügbar, die den Datenaustausch unter unmittelbarer Ausnutzung der Shared Memory Hardware ermöglichen. Diese Routinen arbeiten ohne jegliche Zwischenpuffer oder Message Queues direkt zwischen dem lokalen und entfernten Speicher. Daraus resultieren extrem kurze Latenzzeiten (ca. 1 µs auf den MPP-Systemen), was vor allem für den Austausch kleiner Nachrichten vorteilhaft ist. Oft ist es auch konzeptionell einfacher, Speicherbereiche unmittelbar auszutauschen, anstatt über eine Message-Passing Schittstelle (einseitige Verbindung im Gegensatz zu paarweisem Send-Receive). Die Syntax der beiden wichtigsten Konstrukte in der C Version:

```
shmem_get (long *target, long *source, int length, int pe)
shmem_put (long *target, long *source, int length, int pe)
```

`target`	Anfangsadresse im entfernten PE
`source`	Anfangsadresse im lokalen PE
`length`	Anzahl der 64Bit Worte, die transferiert werden
`pe`	entfernte PE Nummer

Abbildung 38 Beispiel für shmem_get und shmem_put



5.3 Analysehilfsmittel

Performance-Analysen werden mit *MPP Apprentice*, dem Autotasking *Atexpert* ähnlichen Werkzeug durchgeführt. Zur Fehlersuche steht der interaktive Debugger *Cray TotalView* zur Verfügung.

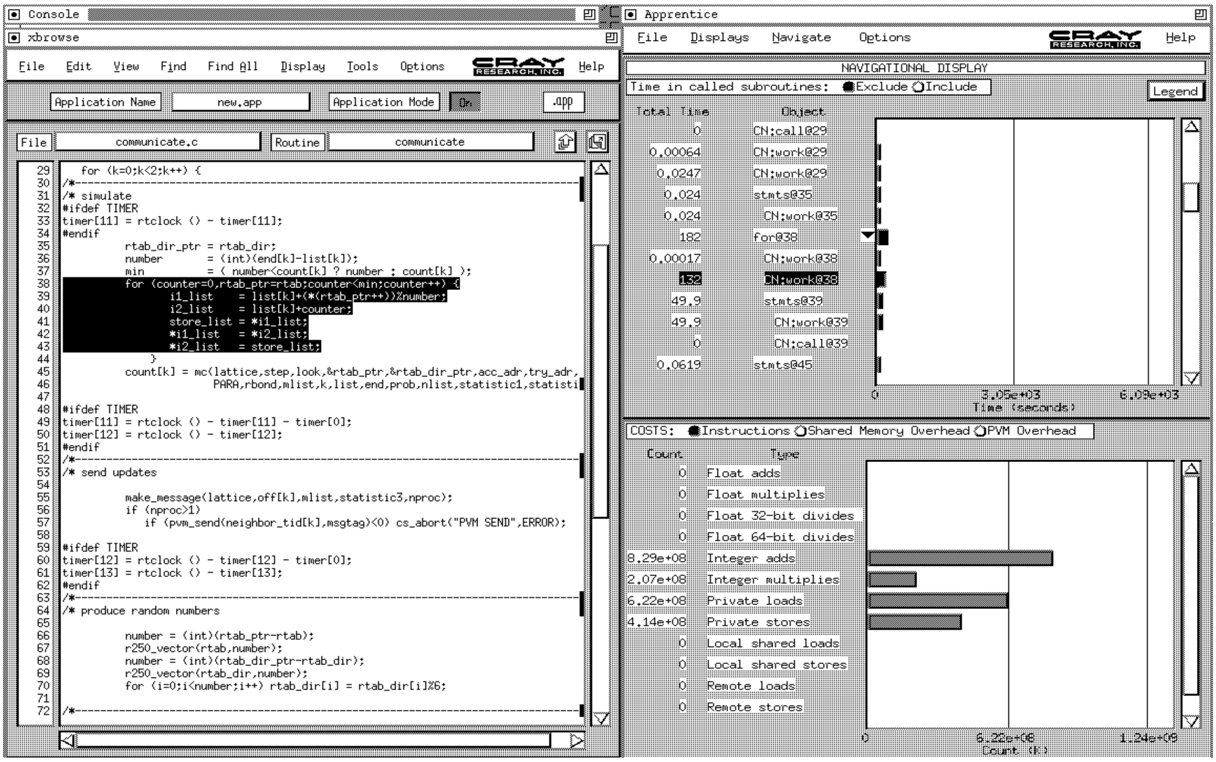
5.3.1 Apprentice

MPP Apprentice vermittelt dem Anwender Informationen über die Leistung einzelner Code-Bereiche oder die Anzeige von Zählern, wie beispielsweise Gleitkommaarithmetik, Speicheroperationen, PVM-Overhead und ähnliches mehr. Über einen Browser kann die gewonnene Information direkt in den Quelltext eingebracht werden. Damit wird man sehr schnell auf die zeitintensiven Routinen oder Problempunkte im Programm geführt.

Durch eine Compiler-Option (`-h apprentice` für C, `-Ta` für Fortran) wird optimierter Code instrumentiert und zur Laufzeit gemessen. Die Analyse erfolgt anschließend über eine X-Window Schnittstelle. In Abbildung 39 wird ein Beispiel gezeigt, auf dessen rechter Seite verschiedene 'Compute Nodes' aufgeführt sind. Ein Compute Node ist eine Code-Sequenz, die gemessen wurde. Bei Auswahl eines Compute Nodes werden im unteren Teilfenster (Costs Display) die Instruktionen dieser Sequenz angezeigt. Gleichzeitig wird im Xbrowse (linkes Fenster) die entsprechende Code-Sequenz durch Unterlegung hervorgehoben.

Es sei noch darauf hingewiesen, daß Apprentice die auf den einzelnen PEs gesammelte Information über alle PEs aufsummiert. Dadurch ist es möglich, diese Messungen mit vertretbarem Aufwand auf hunderten von PEs durchzuführen und auch auszuwerten.

Abbildung 39 Beispiel für Apprentice



Compute Nodes bzw. Routinen werden im Balkendiagramm des 'Navigational Displays' (oberes Teilfenster) durch unterschiedliche Schraffierungen dargestellt. Im einzelnen sind die, im 'Legend' Untermenu aufzeigbaren Zeiterfassungen folgendermaßen zu interpretieren:

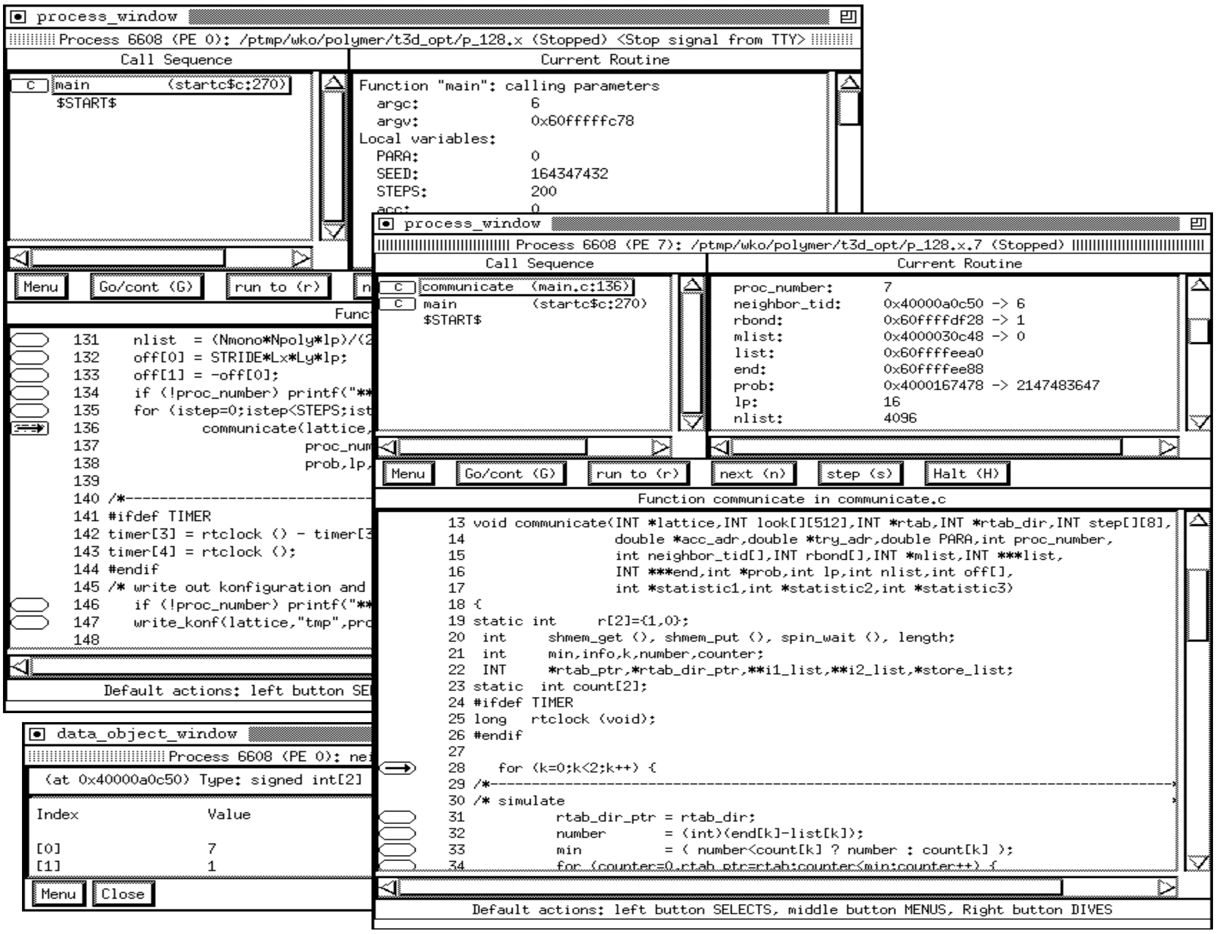
- Overhead Time — Zeit, in der auf andere PEs gewartet wurde (z.B. Barriere)
- Parallel Work Time — CPU Zeit innerhalb der instrumentierten Code-Sequenz
- Called Routine Time — Zeit, die in gerufenen Routinen verbraucht wurde (erfordert Selektierung von ’Include’)
- I/O Time — Zeit für I/O-System-Calls
- Uninstrumented Routine Time — Zeit in nicht instrumentierten Routinen (z.B. Bibliotheksprogramme)

5.3.2 TotalView Debugger

Zur Fehlersuche steht der interaktive Debugger *Cray TotalView* zur Verfügung, mit dem sich sowohl Fortran als auch C Programme direkt auf den einzelnen PEs analysieren lassen. *TotalView* kommt ursprünglich von Bolt, Beranek, and Newman (BBN) und ist auf die CRAY MPP-Systeme angepaßt.

In Abbildung 40 ist ein Beispiel einer Programmanalyse dargestellt. Über einen globalen Breakpoint (auf die Routine "communicate" gesetzt), der aber nur einmal in einem beliebigen PE gesetzt werden muß, kann das Programm auf allen PEs bis dahin laufen. Für jedes PE kann im Bedarfsfall ein eigenes "Process Window" aufgemacht werden (gezeigt sind PE 0 und PE 7), um die PEs speziell zu analysieren, wie beispielsweise schrittweises Weiterarbeiten (in PE 7 ist durch "Step" die nächste Kommandozeile erreicht worden). Das Anzeigen von Variablenwerten erfolgt durch Ansteuern der Variablen direkt im Quelltext. Ein für jedes PE eigenes "Data Object Window" zeigt den aktuellen Wert an.

Abbildung 40 Beispiel einer Programmanalyse mit TotalView



6 Anwendungs-Software

Der langfristige Erfolg massiv-paralleler Prozessorsysteme hängt entscheidend von der Verfügbarkeit stabiler und mit hoher Leistung laufender Anwendungen ab. Cray Research hat zu diesem Zweck verschiedene Initiativen ergriffen, deren Ziel die Portierung und gegebenenfalls auch Neuentwicklung von Anwendungen aus unterschiedlichen Disziplinen ist. Das 1993 etablierte Parallel Applications Technology Program (PATP) vereinigt Partner aus Forschungseinrichtungen, Industrie sowie Software-Häusern. Derzeit sind etwa 65 Applikationen zur Portierung auf Cray Research MPP-Systeme in Arbeit bzw. bereits portiert (Tabelle 2 faßt einige Programmpakete zusammen). Vor allem zeigt sich, daß die Cray Research MPP-Systeme auch auf sehr großen Prozessorzahlen ein ausgezeichnetes Skalierungsverhalten besitzen.

Tabelle 2 Einige Anwendungsprogramme und -pakete für Cray Research MPP-Systeme

Chemie		**Elektronik**	
AMBER	University of California, San Fr.	DAVINCI	Technology Modeling Assoc.
CHARMm	Molecular Simulations, Inc.	FAIM	Vector Technology
DGauss	Cray Research, Inc.	**Umwelt und Klima**	
DGEOM	Jeff Blaney	PARLAM	parallele Version HIRLAM
GAMESS	Iowa State University	IFS	ECMWF 10 Tage Vorhersage
GAUSSIAN	Pittsburgh	OPYC	DKRZ
MNDO93	Universität Zürich	RADM	
Supermolecule	University of Minnesota	GCM	NOAA
T3DMD	Cray Research, Inc.	**Geophysik und Erdölexploration**	
Turbomole	BIOSYM Technologies, Inc.	GEOVECTOR	Comp. Generale de Geophysique
X-PLOR		DISCO	CogniSeis
Strömungsmechanik		ECLIPSE	Intera
FLO67	Princeton University	VIP	Western Atlas Software
FIRE	AVL List GmbH	MORE	Reservoir Simulation Research
FLUENT / UNS		MCTMIG	3DGEO
STAR-CD	Computational Dynamics Ltd.	MIGDMO3	Geophysical Development Corp.
NEKTON	Fluent, Inc.	Falcon	
AIRPLANE	AESOP, Inc.	ProMax	
RAMPANT	Fluent, Inc.	**Bibliotheken**	
Strukturanalyse		LINDA	Scientific Computing Assoc.
LS-DYNA3D	Livermore Software Tech. Corp.	PARMACS	PALLAS GmbH
PAM-CRASH	PAM SYSTEM INTL. S.A.	NAG	Numerical Algorithms Group Inc.
Visualisierung		IMSL	Visual Numerics, Inc.
CAT	Cray Research, Inc.	HARWELL	AEA Technology

7 CRAY T3E Aufträge

Für das CRAY T3E System liegen bereits eine Reihe von Bestellungen vor (LC sind große, flüssiggekühlte Konfigurationen, AC sind kleine bis mittlere luftgekühlte Konfigurationen). Nachfolgend sind einige Kunden aufgeführt:

Europa

Forschungszentrum Jülich (KFA)	CRAY T3E/LC
Max-Planck Gesellschaft RZ Garching (MPIPP)	CRAY T3E/LC
Max-Planck Gesellschaft Göttingen	CRAY T3E/AC
Max-Planck Gesellschaft Berlin	CRAY T3E/AC
Max-Planck Gesellschaft Mainz	CRAY T3E/AC
Max-Planck Gesellschaft Stuttgart	CRAY T3E/AC
Regionales Rechenzentrum Universität Stuttgart (RUS)	CRAY T3E/LC
Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB)	CRAY T3E/LC
Alfred Wegener Institut Bremerhaven	CRAY T3E/AC
TU Berlin (Fakultät für Informatik)	CRAY T3E/AC
Regionales Rechenzentrum Niedersachsen (RRZN)	CRAY T3E/AC
CINECA Bologna	CRAY T3E/AC
Commissariat a l'Energie Atomique, Grenoble (CEA)	CRAY T3E/LC
Commissariat Recherche Nationale Scientifique (IDRIS)	CRAY T3E/LC
Supercomputer Center Helsinki	CRAY T3E/LC
Universität Trondheim	CRAY T3E/AC

USA

Pittsburgh Supercomputer Center	CRAY T3E/LC
North Carolina Supercomputer Center	CRAY T3E/LC
National Oceanic and Atmospheric Administration (NOAA)	CRAY T3E/AC
Electronic Data Systems (EDS)	CRAY T3E/AC
ARPA Arlington, Virginia	CRAY T3E/LC

8 Literaturverzeichnis

[1] Amdahl G.M. *Validity of the single-processor a roach to achieving large scale computing capabilities* Proc. AFIPS 1967, 483-485

[2] Anderson R.J., Snyder L. *A Comparison of Shared and Nonshared Memory Models of Parallel Computation* Proc. IEEE, Vol.79, No.4, April 1991, 480-487

[3] Bailey D.H., Barszcz E., Dagum L., Simon H. *NAS Parallel Benchmark Results* RNR Technical Report RNR-93-016, October 27, 1993

[4] Beguelin A., Dongarra J., Geist A., Sunderam V. *Visualization and Debugging in a Heterogeneuos Environment* IEEE Computer June 1993, 88-95

[5] Behling S.R. *Moving CFD to the T3D: A Progress Report* Proc. Cray User Group Fall 1994, 60-63

[6] Chapman B.M., Mehrotra P., Zima H.P.. *Vienna Fortran - A Fortran Language Extension for Distributed Memory Multiprocessors* Report 91-72, ICASE, Sept.1991

[7] Dally W.J. *Performance Analysis of k-ary n-cube Interconnection Networks* IEEE Trans. on Computers June 1990, 775-785

[8] Dally W.J. *Virtual-Channel Flow Control* IEEE Trans. on Parallel and Distributed Systems March 1992, 194-205

[9] Dixon B., Swallow J. *Sorting on the CRAY T3D* Proc. CUG Spring 1995, 24-34

[10] Dobberpuhl D., et.al. *A 200MHz 64 Bit Dual Issue CMOS Microprocessor* Digital Technical Journal vol.4, no.4, 1992

[11] Ewing A., Henty D. *HPF on the Cray T3D - A Comparison with Craft T3D* Technical Report EPCC-TR95-05 Oct.1995

[12] Felperin S.A., Gravano L., Pifarre G.D., Sanz J. *Routing Techniques for Massively Parallel Communication* Proc. IEEE, Vol.79, No.4, April 1991, 488-503

[13] Feng T. *A Survey of Interconnection Networks* IEEE Computer Dec. 1981, 12-27

[14] Fox G. *Solving Problems on Concurrent Processors* Prentice Hall 1988

[15] Fox G., Hiranandani S., Kennedy K., Koelbel C., Kremer U., Tseng C., Wu M. *FORTRAN D Language Specification* Rice Univ. April 1991

[16] Gustafson J.L. *Reevaluating Amdahl's Law* Comm. ACM, May 1988

[17] Hansen C., de Verdiere G.C., Troutman R. *Binary-Swap Volumetric Rendering on the T3D* Proc. CUG Spring 1995, 61-69

[18] Henty D. *Craft Performance Optimisation T3D* Technical Report EPCC-TR95-04 Sept.1995

[19] Hill M.D. *What is Scalability?* Computer Architecture News Dec.1990, 18-21

[20] Hockney R. *Performance parameters and benchmarking of supercomuters* Parallel Computing 17 (1991), 1111-1130

[21] Kermani P., Kleinrock L. *Virtual cut-through: A new computer communication switching technique* Computer Networks, vol.3, 1979, 267-286

[22] Kessler R.E., Schwarzmeier J.L. *CRAY T3D: A new Dimension for Cray Research* Proc.COMPCON 1993, 176-182

[23] Levesque J.M. *Using High Performance Fortran on the CRAY T3D* Proc. Cray User Group Fall 1994, 418-422

[24] Loveman D.B. *High Performance Fortran* IEEE Parallel & Distributed Technology, Feb 1993, 25-42

[25] Ni L.M., McKinley P.K. *A Survey of Wormhole Routing Techniques in Direct Networks* IEEE Computer Feb. 1993, 62-76

[26] Numrich R.W., Springer P.L., Peterson J.C. *Measuremen of Communication Rates on the CRAY T3D Interprocessor Network* Proc. HPCN Europe '94, Munich, April 18-20, Springer Verlag

[27] Nussbaum D., Agarwal A. *Scalability of Parallel Machines* Comm. ACM March 1991, 57-61

[28] Oed W. *Das Cray Research massiv-parallele Prozessorsystem CRAY T3D* PIK 2/1994 96-104

[29] Palacharla S., Kessler R.E. *Evaluating Stream Buffers as a Secondary Cache Replacement* Proc. 21st Annual Intl. Symposium on Computer Architecture, April 1994

[30] Pase D.M., MacDonald T., Meltzer A. *MPP Fortran Programming Model* Cray Research Feb. 1993

[31] Rice University *High Performance Fortran Language Specification* Mai 1993

[32] Sawley M-C. *The CRAY T3D at EPFL: First Activities* SPEEDUP Dec. 1994, 59-64

[33] Siegel H.J. *Interconnection Networks for Large -Scale Parallel Processing* McGraw Hill 1990

[34] Singh J.P., Hennessy J.L., Gupta A. *Scaling Parallel Programs for Multiprocessors: Methodology and Examples* IEEE Computer July 1993, 42-50

[35] Sites R.L. *Alpha Architecture Reference Manual* Digital Press 1992

[36] Stenström P. *A Survey of Cache Coherence Schemes for Multiprocessors* IEEE Computer June 1990, 12-24