folge in der Seitenkacheltabelle des Betriebsystemkerns ein- und ausgetragen werden. Diese Funktionen können bei jedem Knoten ausgeführt werden, der ein Speichersegment eines Nachbarknotens bei sich zur Verfügung stellen möchte.

Zu verschiedenen Gelegenheiten muß mit einem Nachbarknoten Information über gemeinsame Speichersegmente ausgetauscht werden. Die Kommunikation zwischen den Shared-Memory-Moduln direkter Nachbarn wird über den Nachrichtenmechanismus (Abschnitt 2.1.5) abgewikkelt. Die Notwendigkeit zur Kommunikation tritt zum Beispiel auf, wenn ein Speichersegment bei einem Prozeß eines Nachbarknotens eingeblendet werden soll (Anfordern der Nummer der ersten Seite) oder wenn ein Speichersegment zerstört wird (Markieren des Segments beim Nachbarn als zerstört). Die Benutzerschnittstelle ist ausführlich in [Kar91] beschrieben und soll hier nicht weiter ausgeführt werden.

3 Zusammenfassung

Obwohl in diesem Artikel die am eingesetzten Betriebssystem vorgenommenen Erweiterungen bei weitem nicht erschöpfend dargestellt werden konnten, so wurden doch die wesentlichen Elemente angesprochen. Es wurde das Applikationskonzept eingeführt, sowie auf die Bedeutung des Kommunikationsspeichers eingegangen und die wichtigsten Kommunikationsmechanismen vorgestellt.

4 Literaturverzeichnis

Bericht92 M. Dal Cin, *Rechnerbaukasten für funktionsorientierte Multiprozessoren mit Fehlertoleranzeigenschaften*, in: Arbeits- und Ergebnisbericht des SFB 182 Multiprozessor- und Netzwerkkonfigurationen 1990 - 1992, Seite 114 ff

Kar91 F. Kardel, W. Stukenbrock, T. Thiel, S. Turowski, *Anleitung zur Benutzung des MEMSY-Programmiermodells für Anwender*, interner Bericht, IMMD 4, Univ. Erlangen-Nürnberg, Oktober 1991

MOT89 Karl J. Rusnock, *Multiprocessor SYSTEM V/88 Release 3 Design Specification*, MOTOROLA Confidential Proprietary, May 1989

Rotzoll92 B. Rotzoll, *Dynamischer Multiprogrammbetrieb von Parallelrechnern*, SFB 182-Herbsttagung, 1992

2.1.5 Nachrichtenmechanismus

Der Nachrichtenmechanismus ist ein Kommunikationsmodul, welches direkt auf dem Unterbrechungsmechanismus aufbaut. Durch ihn werden Nachrichten realisiert, die sowohl durch Module des Kerns (zum Beispiel Semaphore und Signale), als auch, über eine Schnittstelle, von Anwendungen verwendet werden können. Die Basis bildet eine Menge von Nachrichtenpuffern, die im knoteneigenen Kommunikationsspeicher angelegt wurden. Diese werden für alle Nachrichten zu Knoten verwendet, die ebenfalls Zugriff auf diesen Kommunikationsspeicher haben. Eine spezielle Verwaltung der Puffer sorgt dafür, daß gesendete Puffer unter anderem getrennt nach Empfängerknoten verwaltet werden, was ihre Behandlung bei Knotenausfällen erleichtert. Darüberhinaus werden empfangene Puffer nicht sofort wieder zurückgegeben, sondern entweder nur auf besondere Anforderung hin, wenn eine bestimmte Anzahl überschritten ist oder wenn selbst eine Nachricht in diese Richtung zu schicken ist, der man die Puffer mitgeben kann.

Der Nachrichtenmechanismus ist analog zum Unterbrechungsmechanismus aufgebaut. Auch er unterscheidet verschiedene Typen und ruft beim Empfang einer Nachricht die registrierte Behandlungsfunktion auf. Eine Reihe von internen Typen, zum Beispiel für die aus dem Programmiermodell bekannten 2-Wort-Nachrichten [Kar91], ist bereits vordefiniert. Nachrichten werden sowohl beim Senden, als auch beim Empfangen zwischengepuffert. Verschiedene Funktionen erlauben es die Empfangswarteschlangen zu manipulieren. Potentiell kann der Nachrichtenmechanismus auch für Nachrichten zu entfernteren Knoten verwendet werden. Die entsprechenden Routinen sind bereits vorhanden.

Obwohl der Nachrichtenmechanismus sehr sicher ist, wurden noch zusätzlich besondere Protokolle implementiert, um die Nachrichtenübertragung sicherer zu gestalten.

2.1.6 Shared-Memory-Mechanismus

Als weiterer Kommunikationsmechanismus ist in MEMSOS die Kommunikation über gemeinsame Speicherbereiche realisiert. Zur Unterstützung dieses Mechanismus wird von der Kommunikationsspeicherverwaltung eine Schnittstelle bereitgestellt.

Über Anforderungs- und Freigabeaufrufe kann Speicher mit einer Granularität von 4kByte Seiten verwaltet werden. Ein angeforderter Speicherbereich muß sich nicht notwendigerweise aus physikalisch zusammenhängenden Seiten zusammensetzen. Es muß lediglich eine Liste mit den aufeinanderfolgenden Seitennummern geführt werden, die im Kommunikationsspeicher abgelegt wird. Die Nummer der ersten Seite genügt dann als Referenz. Auf diese Art kann die notwendige Information über ein angefordertes Speichersegment leicht einem Nachbarknoten zugänglich gemacht werden, indem die Nummer der ersten Seite übermittelt und Zugriff auf die Seitentabelle ermöglicht wird. Mit Hilfe von Funktionen zum Ein- und Ausblenden von Speicherbereichen in den virtuellen Adreßraum können die einzelnen Seiten in der richtigen Reihen-

Der erste Bereich liegt immer am Anfang des Speichers, hat eine feste Größe und ist für wichtige Datenstrukturen der einzelnen Mechanismen reserviert. Hier ist unter anderem die Nummer des zugehörenden Knotens, die Größe des Speichers, Referenzen auf besondere Speicherbereiche und Datenstrukturen und die Datenstrukturen für den initialen Verbindungsaufbau abgelegt.

Der zweite Bereich unterliegt der Kommunikationsspeicherverwaltung. Analog zum Kernadreßraum kann über eine Schnittstelle physikalisch zusammenhängender Speicher angefordert werden. Dies muß bereits während der Initialisierungsphase des Systems erfolgen. Das Nachrichten- und das Transportmodul fordern zum Beispiel über diese Schnittstelle Speicher für Puffer an. Eine Ausnahme bildet hier das Shared-Memory-Modul, dem sämtlicher verbliebener Speicher zugeteilt wird, wenn alle Speicheranforderungen der anderen Module befriedigt sind.

Für den Shared-Memory-Mechanismus wird von der Kommunikationsspeicherverwaltung eine eigene Schnittstelle zur Verfügung gestellt, die in Abschnitt 2.1.6 näher erläutert ist.

2.1.4 Unterbrechungsmechanismus

Damit effizient mit den Nachbarknoten kommuniziert werden kann, ist es unerläßlich Unterbrechungen (Interrupts) bei einem direkten Nachbarn auslösen zu können. Dazu wird eine eigene Baugruppe eingesetzt, auf die hier allerdings nicht näher eingegangen werden soll.

Mit jeder Unterbrechung, die bei einem Nachbarknoten ausgelöst wird, wird ein Datenwort im eigenen Kommunikationsspeicher für den Nachbarknoten abgelegt. Die Unterbrechungs-Hardware kann erkennen von welchem Nachbarn die Unterbrechung ausgelöst wurde. Mit dieser Information kann der Kommunikationsspeicher des auslösenden Knotens bestimmt und das Datenwort ausgelesen werden.

Das Datenwort besteht aus zwei Teilen, einer Typinformation und den eigentlichen Nutzdaten, deren Interpretation vom Typ abhängen. Ein Teil der möglichen Typen wird für interne Aufgaben, wie Verbindungsaufbau und Statusmeldungen benötigt. Die restlichen sind im Kern frei verfügbar.

Damit dieser Mechanismus von anderen Moduln genutzt werden kann, wird eine entsprechende Schnittstelle angeboten. Von einem Modul müssen lediglich ein oder mehrere Unterbrechungstypen reserviert und entsprechend viele Behandlungsfunktionen, welche die Nutzdaten verarbeiten, registriert werden. Statusmeldungen können auch über diese Funktionen zugänglich gemacht werden. Das Auslösen einer Unterbrechung erfolgt über eine Funktion, die als Parameter im wesentlichen die Nummer des Empfängerknotens, den Unterbrechungstyp und den Nutzdatenanteil benötigt. Der Rückgabewert der Funktion gibt an, ob die Unterbrechung durch den Nachbarknoten erkannt wurde. Erkennt ein Knoten eine Unterbrechung, so wird die für den erkannten Typ registrierte Behandlungsfunktion aufgerufen, welche die Weiterverarbeitung übernimmt.

Während der Prozeßumschaltung kann sich diese Bindung eines Prozesses abhängig von der Systembelastung ändern. Die Prozeßumschaltung ist eine atomare Aktion und kann von allen Prozessoren parallel vorgenommen werden. Eine Prozeßwarteschlange kann immer nur exklusiv von einem Prozessor verändert werden. Das Umschalten eines Prozessors auf eine andere Warteschlange ist möglich. Für Prozesse, vor allem für Treiber, ist es wichtig, daß sie in der Lage sind den Prozessor auszuwählen, auf dem sie ausgeführt werden sollen, da manche Hardware-Bausteine nicht von allen Prozessoren aus zugänglich sind. In der aktuellen Implementierung existiert nur eine Prozeßwarteschlange für alle Prozessoren.

2.1.2.2 Änderungen und Erweiterungen für MEMSOS

Zieht man in Betracht, daß auf MEMSY sowohl Applikationen mit langen Lauf- und Rechenzeiten und wenigen oder keinen Benutzer-Interaktionsphase zur Ausführung kommen sollen, als auch Testbetrieb, das heißt möglicherweise viele kurze Applikationen mit längeren Interaktionsphasen, möglich sein soll, wird klar, warum die Prozessorvergabe verändert werden muß.

Beim Erzeugen einer Applikation wird dieser eine bestimmte Priorität zugewiesen. Diese Priorität kann durch einen Systemaufruf verändert werden. Auf Knotenebene wird zusätzlich zur vorhandenen Prozeßwarteschlange, im folgenden Systemwarteschlange genannt, je eine Warteschlange für jede mögliche Applikationspriorität, die Applikationswarteschlangen, eingeführt. Die Systemwarteschlange wird fest einem Prozessor zugeordnet. Nur wenn keine ‘Systemprozesse’ mehr vorliegen, kann dieser Prozessor auch eine Applikationswarteschlange abarbeiten. Auf jedem Knoten wird ein weiterer Dämon gestartet, der für die ‘globale’ Prozeß- und Applikationsumschaltung verantwortlich ist. Er wertet die Laufzeiten der Applikationen aus und ist in der Lage die Applikationsprioritäten nach globalen Gesichtspunkten zu verändern. So sollen zum Beispiel Langläufer in ihrer Priorität zurückgestuft und interaktive Applikationen bevorzugt werden. Zudem ist es möglich verschiedene Umschaltstrategien einzusetzen und zu testen. Die dazu notwendigen globalen Mechanismen sind noch nicht vollständig implementiert. In [Rotzoll92] wird näher auf diese Thematik eingegangen.

2.1.3 Kommunikationsspeicher und seine Verwaltung

Wie in Abschnitt 1.2 angeführt, ist jedem Knoten mindestens ein physikalischer Kommunikationsspeicher zugeordnet. Jeder Knoten ist exklusiv für die Verwaltung des jeweiligen Kommunikationsspeichers verantwortlich. Durch diese eindeutige Zuordnung wird die Verwaltung des Kommunikationsspeichers, die Behandlung von Knotenausfällen und der initiale Verbindungsaufbau zu den direkten Nachbarn wesentlich vereinfacht.

Jeder Kommunikationsspeicher ist gleichermaßen in zwei aufeinanderfolgende Bereiche unterteilt. Dadurch wird erreicht, daß die Sicht eines Knotens auf alle erreichbaren Kommunikationsspeicher gleich bleibt.

2.1.1 Anwendungsunterstützung - Applikationen

Von Seiten der Anwender können an MEMSY verschiedene Anforderungen gestellt werden. Hier sind unter anderem hohe Verfügbarkeit des Systems (Ausfallsicherheit und einfaches Aufsetzten nach einem Knotenausfall), einfache Handhabbarkeit, kurze Aufsetzphasen, gewohnter oder gar erweiterter Funktionalitätsumfang und die Möglichkeit zu interaktivem Testen von Anwendungen, ohne den Produktionsbetrieb zu stark zu stören, zu nennen.

Damit die oben angeführten Anforderungen erreicht werden können, mußte ein Applikationskonzept erstellt werden. Eine Applikation soll alle Prozesse, die durch eine Anwendung auf dem System erzeugt werden umfassen. Eine Applikation ist somit ein knotenübergreifendes Prozessystem. Um mehrere parallele Applikationen realisieren zu können, muß jede Applikation global eindeutig identifizierbar sein. Dies wird durch eine Applikationsnummer erreicht.

Beim Systemstart wird initial die ausgezeichnete Applikation 0 erzeugt. Diese ausgezeichnete Applikation setzt sich aus Applikations-Dämonen zusammen, die auf jedem Knoten gestartet werden. Die Dämonen haben zunächst die Aufgabe auf Anforderung hin neue Applikationen zu erzeugen. Dazu wird auf einem beliebigen Knoten durch das Anwenderprogramm eine entsprechende Anforderung zur Erzeugung einer neuen Applikation gestellt. Die Dämonen vergeben eine noch freie Applikationsnummer und erzeugen die sogenannten Leader-Prozesse für diese Applikation. Dabei wird genau ein Leader-Prozeß auf jedem Knoten oder, je nach Vorgabe, auf einer Teilmenge der Knoten gestartet. Dieser Leader-Prozeß ist seinerseits wieder ein Dämon, der die benötigte Umgebung für die Applikation bereitstellt, initiale Koordinationsaufgaben übernimmt und insbesondere Leben und Sterben der von ihm gestarteten Prozesse überwacht. Knotenlokal lassen sich Prozesse, die als Kinder einer Applikation gestartet wurden, wiederrum eindeutig anhand ihrer sogenannten Tasknummer identifizieren. Diese Prozesse arbeiten an der eigentlichen, durch die Anwendung gestellten Aufgabe.

Aus dieser Organisation ergibt sich ein hierarchischer Aufbau, wodurch eine verhältnismäßige einfache Behandlung von Applikationen möglich ist.

2.1.2 Prozessorvergabe

Wesentlich für eine gute Auslastung der einzelnen Prozessoren auf Knotenebene ist, wie die Prozeß-Prozessor Bindung realisiert und welche Prozessorvergabestrategie eingesetzt wird. Im Hinblick auf das vorgestellte Applikationskonzept muß somit untersucht werden, wie sie das Konzept unterstützen. Im folgenden soll daher zunächst kurz auf den Status Quo im verwendeten UNIX eingegangen werden. Anschließend werden die Änderungen vorgestellt.

2.1.2.1 Vorgehensweise in SYSTEM V/88 Release 3

Die Prozeß-Prozessor-Bindung wird durch Prozeßwarteschlangen (Run Queues) ausgedrückt. Jedem Prozessor ist eine eigene Prozeßwarteschlange zugeordnet, die von ihm bevorzugt abgearbeitet wird. Jeder Prozeß ist durch eine Referenz an eine bestimmte Warteschlange gebunden. Steht ein Prozeß zur Ausführung an, wird er in die referenzierte Warteschlange eingetragen.

1.3 Einsatzgebiete

Aus konzeptioneller Sicht kann nicht jede Architektur gleichermaßen gut für alle Problemklassen geeignet sein. Eine bedeutende Rolle spielt hier, wie stark sich Probleme überhaupt parallelisieren lassen, wie gut sie für Datenpartitionierung geeignet sind und wie das Kommunikationsprofil aussieht. Durch seine Topologie und seinen Hardware-Aufbau eignet sich MEMSY vorwiegend für hochparallelisierbare Aufgaben mit überwiegend Nearest-Neighbour-Datenabhängigkeit. Dies trifft zum Beispiel für Aufgaben aus den Klassen numerische Gitteraufgaben (z.B. Multi-Grid-Algorithmen),Vielteilchenmodelle und quantenmechanische Berechnungen zu.

2 Betriebssystemarchitektur

Als Basis für MEMSOS, dem Betriebssystem von MEMSY, wird das auf die Multiprozessorarchitektur des HYPERmodule abgestimmte UNIX[2] SYSTEM V/88 Release 3 von MOTOROLA verwendet. SysV/88 Unix[3] ist eine Betriebssystemarchitektur für gleichberechtigte Prozessoren. Gleichberechtigt bedeutet, daß jeder Prozessor sowohl die E/A bedienen kann, als auch Benutzercode ausführen. Viele Teile des Kerns können parallel von allen Prozessoren ausgeführt werden. Dies trifft zum Beispiel für die Prozeßumschaltung zu, die neben den Prozessorvergaberoutinen atomar ist. Zusätzlich sind neue Primitive eingeführt worden, die den gegenseitigen Ausschluß gewährleisten. Ein großer Teil des Kerns wird durch eine Semaphorvariable vor paralleler Ausführung geschützt. Viele weitere Multiprozessorkonzepte sind bereits in SysV/88 enthalten [MOT89], da es sich aber um ein Konzept handelt, das auf eine Hardwarearchitektur mit globalem gemeinsamen Speicher aufbaut, müssen zum einen die Realisierungen dieser Konzepte modifiziert und zum anderen zusätzliche Erweiterungen eingebracht werden.

2.1 Erweiterungen am eingesetzten Betriebssystem

Wie bereits erwähnt sind nicht alle Mechanismen von SysV/88 ohne Modifikationen für MEMSOS einsetzbar. Dies trifft in besonderem Maße für die Interprozessorkommunikation zu, die sich mit Hilfe eines globalen gemeinsamen Speichers und simpler Lock-Techniken einfacher durchführen läßt, als in einem System mit verteiltem gemeinsamen Speicher. Innerhalb eines Knotens kann auf die vorhanden Mechanismen zurückgegriffen werden. Für die Kommunikation und Koordinierung zwischen den Knoten werden neue Mechanismen eingeführt. Dies sind unter anderem Signale, Semaphore, Nachrichten und Shared-Memory-Mechanismen. Als Voraussetzung und Basis für diese Erweiterungen dient ein Unterbrechungsmechanismus und eine Kommunikationsspeicherverwaltung. Damit MEMSY den verschiedensten Anforderungen der Anwender gerecht werden kann ist zudem das Applikationskonzept eingeführt worden.

2. UNIX ist ein eingetragenes Warenzeichen von AT&T
3. Im weiteren Text ist mit SysV/88 immer das UNIX SYSTEM V/88 Release 3 von Motorola gemeint.

einer Ebene, symmetrisch ist. Das heißt, die Knoten der B-Ebene können zwar auf den Kommunikationsspeicher der zugeordneten Knoten der A-Ebene zugreifen, nicht aber umgekehrt. Die C-Ebene besteht nur aus einem einzigen Knoten, der über einen Bus mit den Knoten der B-Ebene verbunden ist.

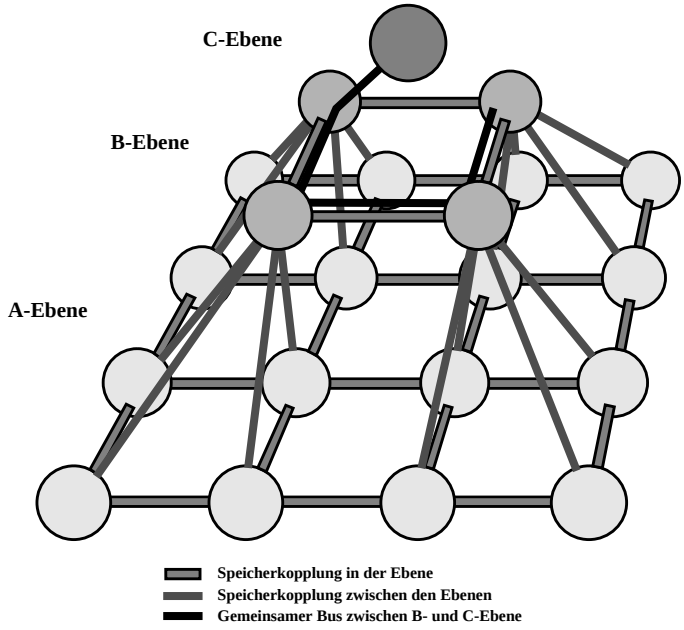

Abb. 1.1 20er MEMSY System (ohne Torusverbindungen)

1.2 Aufbau eines MEMSY-Knotens

Ein Knoten der A- oder B-Ebene setzt sich aus dem Prozessormodul mit integriertem Cache, dem lokalen Hauptspeicher und spezieller Zusatzbaugruppen zusammen. Diese dienen zur Ansteuerung des Kommunikationsspeichers und der Meßschnittstelle, zum optionalen Anschluß eines Spezialprozessors und zur Erzeugung von Unterbrechungen bei Nachbarknoten über dedizierte Verbindungen. Darüberhinaus ist pro Knoten ein privater Plattenspeicher und ein FDDI-Anschluß vorhanden. Das Konzept von MEMSY sieht vor, daß sich das System soweit möglich aus käuflichen Komponenten zusammensetzten soll. Als Prozessormodul wurde daher das Multiprozessor HYPERmodule[1] mit vier RISC-Prozessoren der 88k-Familie von MOTOROLA gewählt.
Neben dem Hauptspeicher ist jedem Knoten ein physikalischer Kommunikationsspeicher zugeordnet. Dieser Speicher wird grundsätzlich wie normaler Hauptspeicher behandelt. Im gekoppelten System entscheidet lediglich die Adressierung auf welchen der verfügbaren Speicher (Hauptspeicher, eigener Kommuniakationsspeicher oder Nachbarkommunikationsspeicher) zugegriffen wird. Eine detailiertere Beschreibung der Hardware erfolgt in [Bericht92].

1. HYPERmodule ist ein Warenzeichen der MOTOROLA Inc.

Betriebssystemarchitektur von MEMSY

Thomas Thiel

Universität Erlangen-Nürnberg
Institut für mathematische Maschinen und Datenverarbeitung IV
Lehrstuhl für Betriebssysteme

Überblick

MEMSY ist ein speichergekoppeltes Hochleistungsmultiprozessorsystem mit verteiltem, lokal gemeinsamen Speicher. Um dem Anwender eine adäquate Nutzung von MEMSY zu ermöglichen wird das dedizierte Betriebssystem MEMSOS entwickelt, welches speziell an die vorliegende Hardware angepaßt ist, da keines der käuflichen Betriebssysteme die besonderen Erfordernisse der für MEMSY entwickelten Speicherkopplung unterstützt. Im folgenden soll kurz auf die wichtigsten Erweiterungen und Änderungen eingegangen werden.

1 Das Multiprozessorsystem MEMSY

MEMSY ist ein modular erweiterbares Multiprozessorsystem, das im Rahmen des Sonderforschungsbereiches 182 aufgebaut, untersucht und eingesetzt wird. Es soll als interdisziplinäres Querschnittsthema eine Brücke zwischen den einzelnen Schwerpunkten der beteiligten Forschungsgruppen schlagen. Insbesondere soll auch die Anwenderseite miteinbezogen werden.

1.1 Topologie

MEMSY hat eine pyramidenförmige Struktur und kann in drei Ebenen unterteilt werden (A-, B- und C-Ebene). Die Knoten der unteren beiden Ebenen sind jeweils in einem Gitter angeordnet, wobei die Ränder zum Torus geschlossen sind. Jedem Knoten aus der B-Ebene sind genau vier Knoten aus der untersten Ebene, der A-Ebene, zugeordnet. Die Kopplung zwischen den Knoten innerhalb einer Ebene erfolgt über gemeinsame Speicherbereiche. Dazu ist jedem Knoten ein Multiportspeicher, der Kommunikationsspeicher, zugeordnet. Ein Knoten kann dann auf den Kommunikationsspeicher seiner direkten Nachbarn zugreifen (Nearest-Neighbour-Prinzip). Dadurch ergibt sich eine konstante lokale Verbindungskomplexität. Die Verbindung zwischen den Ebenen erfolgt ebenfalls über gemeinsame Speicher, wobei sie nicht, wie innerhalb

Betriebssystemarchitektur von MEMSY

Thomas Thiel

Oktober 1992 TR-I4-10-92

Interner Bericht

Institut für
Mathematische Maschinen
und Datenverarbeitung
der
Friedrich-Alexander-Universität
Erlangen-Nürnberg

Lehrstuhl für Informatik IV
(Betriebssysteme)