APS´97, 4. ITG / GI - Fachtagung Arbeitsplatz-Rechensysteme
Koblenz-Landau, 1997

Ein Workstation-Cluster für paralleles Rechnen in Robotik-Anwendungen[1]

Christian WURLL und Dominik HENRICH

Institut für Prozeßrechentechnik und Robotik
Universität Karlsruhe, D-76128 Karlsruhe
Email: [wurll, dHenrich]@ira.uka.de

Die Bewegungsplanung für Industrieroboter ist eine notwendige Voraussetzung, damit sich autonome Systeme kollisionsfrei durch die Umwelt bewegen können. Die Berücksichtigung von dynamischen Hindernissen zur Laufzeit erfordert allerdings leistungsfähige Algorithmen, zur Lösung dieser Aufgabenstellung in Echtzeit. Eine Möglichkeit zur Beschleunigung der Algorithmen ist der effiziente Einsatz von skalierbarer Parallelverarbeitung. Die softwaretechnische Umsetzung kann aber nur dann erfolgreich sein, wenn ein Parallelrechner zur Verfügung steht, der einen hohen Datendurchsatz bei geringer Latenzzeit bietet. Darüber hinaus muß dieser Parallelrechner unter vertretbarem Aufwand bedienbar sein und ein gutes Preisleistungsverhältnis aufweisen, damit die Parallelverarbeitung verstärkt in der Industrie zum Einsatz kommt. In diesem Artikel wird ein Workstation-Cluster auf der Basis von neun Standard-PCs vorgestellt, die über eine spezielle Kommunikationskarte miteinander vernetzt sind. In den einzelnen Abschnitten werden die gesammelten Erfahrungen bei der Inbetriebnahme, Systemadministration und Anwendung geschildert. Als Beispiel für eine Anwendung auf diesem Cluster wird ein paralleler Bewegungsplaner für Industrieroboter beschrieben.

Schlüsselwörter: Workstation-Cluster, Netz-Architekturen, Netzwerkmanagement, Parallel Virtual Machines (PVM), Robotik, Bewegungsplanung, Suchverfahren

1. Einleitung

Ein Teilgebiet der Robotik, die Bewegungsplanung, ermöglicht autonomen Robotern sich kollisionsfrei durch die Umwelt zu bewegen. Leider werden bei der notwendigen Berücksichtigung von dynamischen Hindernissen die Laufzeiten für die Bewegungsplanung zu lang, als daß sie sinnvoll in ein reaktives Robotiksystem integriert werden könnten. Daher beschäftigt sich ein Projekt am Institut für Prozeßrechentechnik und Robotik der Universität Karlsruhe damit, die Resultate von sequentieller Graphensuche und hierarchischer Abstandsberechnung in einem parallelen Bewegungsplaner für reale Robotersysteme zu verschmelzen. Die Parallelisierung erfordert gut skalierbare Algorithmen, um die vorhandene Rechenparallelität bei wachsender Prozessorzahl effizient auszunutzen. Insbesondere sollten die Algorithmen in der Lage sein, sich an die Charakteristika von unterschiedlichen Paral-

[1] Diese Arbeit ist am Institut für Prozeßrechentechnik und Robotik (IPR) der Universität Karlsruhe unter der Leitung von Prof. Dr. U. Rembold, Prof Dr. H. Wörn und Prof. Dr. R. Dillmann entstanden und wurde im Rahmen des DFG-Projektes „Skalierbare Algorithmen für die parallele Bewegungsplanung in dynamischer Umgebung“ im Schwerpunktprogramm „Effiziente Algorithmen für diskrete Probleme und ihre Anwendungen“ gefördert. Weitere Informationen gibt es auf der Web-Seite der PaRo-Gruppe (Parallele Robotik) des IPRs unter http://wwwipr.ira.uka.de/~paro/.

lelrechner-Architekturen anzupassen. Neben „typischen“ Parallelrechnern (Parsytec, IBM-SP2, MasPar MP-1/2, Intel Paragon, usw.) eignet sich zur Lösung dieser Aufgabenstellung auch immer mehr die Vernetzung von einzelnen Workstations zu einem Cluster, die über eine Message-Passing-Schnittstelle (wie z.B. PVM, MPI, P4, ...) miteinander kommunizieren. Der typische Flaschenhals dieser Workstation-Cluster ist die Verbindung über Bussysteme, die eine gleichzeitige 1-zu-1-Kommunikation verhindern. Gesucht ist also eine Hardware-Architektur für vernetzte Arbeitsplatzrechner, die diesen Engpaß vermeidet.

In Abschnitt 2 wird ein Workstation-Cluster auf der Basis von Standard-PCs vorgestellt. Abschnitt 3 gibt einen Überblick über die Programmierumgebung des Clusters. Die gemessenen Kommunikationsleistungsdaten werden in Abschnitt 4 dargestellt. In Abschnitt 5 wird eine reale Anwendung, die Bewegungsplanung von Industrierobotern, beschrieben, die auf diesem Workstation-Cluster implementiert ist. Der Artikel endet mit einer Zusammenfassung und einem Ausblick.

2. Hardware

Aufgrund der großen Verbreitung, einem guten Preisleistungsverhältnis und der leichten Erweiterbarkeit ist der Einsatz von Standard-PCs in industriellen Anwendungen nachwievor ansteigend. Auch im Bereich der Robotik werden PCs nun vermehrt verwendet. Zum Beispiel eignen sich PCs als Robotersteuerung oder durch die Verfügbarkeit von qualitativ hochwertigen Bildverarbeitungskarten mittlerweile auch als Sensorrechner. Daher bietet es sich an, Standard-PCs als Plattform für ein Workstation-Cluster zu verwenden, um paralleles Rechnen in Robotik-Anwendungen verfügbar zu machen.

Arbeitsplatzrechner werden in der Regel über Ethernet-basierte LAN's zusammengefaßt. Das üblicherweise eingesetzte Kommunikationsprotokoll "Carrier Sense Multiple Access / Collision Detection" (CSMA / CD) verhindert den gleichzeitigen Sendezugriff von mehreren Stationen auf den Bus. Für eine effiziente Ausführung von parallelen Algorithmen ist dies allerdings von großem Nachteil, da die Übertragungszeiten und Kapazitäten viel zu schlecht sind.

In Abschnitt 2.1 wird eine Kommunikationskarte vorgestellt mit deren Hilfe PC-basierte Workstation-Cluster wie z.B. die IPR-ParaStation (siehe Abschnitt 2.2) aufgebaut werden können.

2.1 ParaStation-Karte

Zur effizienten parallelen Vernetzung eignen sich Hochgeschwindigkeitskommunikationskarten wie z.B. die ParaStation-Karte, die in einem PCI-Slot eingesteckt werden kann. Die Kommunikation erfolgt über externe Verbindungen mit Hilfe von 50-poligen Flachband-Kabeln. Die Karte wurde am Institut für Programmstrukturen und Datenorganisation der Universität Karlsruhe entwickelt und ist bereits auf zahlreichen Konferenzen und Messen vorgestellt worden [Warschko95, Warschko96a, Warschko96b, Blum96a, Blum96b]. Bei der Entwicklung der ParaStation-Karte wurde vor allem darauf geachtet, kurze Latenzzeiten und hohe Übertragungsraten zu erreichen, die eine feinkörnige Parallelisierung erlauben.

Zusammenfassend besitzt die ParaStation-Karte folgende Eigenschaften:

- Da die Karte über zwei Ein- und Ausgänge verfügt, basiert die Netzwerktopologie entweder auf einem zweidimensionalen, rückgekoppeltem Gitter (2D-Torus) oder - speziell für kleinere Konfigurationen - auf einem bidirektionalen Ring. Das Netzwerk arbeitet vollkommen autonom auf der Basis einer tabellengesteuerten Paketvermittlung.
- Die einzelnen Pakete werden mittels eines virtual cut-through Verfahrens weitergeleitet, wobei die Flußkontrolle vollständig auf Kanalebene in Hardware implementiert ist. Jeder Knoten ist mit einer eigenen Routingtabelle und drei Eingangspuffern ausgestattet. Zwei dieser Puffer dienen zur Zwischenspeicherung von Paketen, die von anderen Knoten

kommen; über einen weiteren Puffer wird die Kommunikation mit dem zugeordneten Prozessor abgewickelt.

- Die ParaStation-Karte ermöglicht einen maximalen Datendurchsatz von 20 MBytes/s bei einer Latenzzeit von 250 ns. Die Pakete können zwischen 4 - 508 Bytes groß sein. Insgesamt können Cluster von 2 bis 100 Stationen gebildet werden [Warschko96a].

2.2 IPR-ParaStation

Auf der Basis der ParaStation-Karten ist am Institut für Prozeßrechentechnik und Robotik (IPR) der Universität Karlsruhe ein Workstation-Cluster aufgebaut worden. In der derzeitigen Ausbaustufe besteht die IPR-ParaStation aus neun PCs in einem Standard-Desktop-Gehäuse, die auf dem freien Markt verfügbar sind. Alle Rechner wurden in einen Rechnerschrank eingebaut, um die Wartung und Systemadmistration zu vereinfachen.

Jeder Rechner verfügt über einen Pentium 133 MHz-Prozessor, auf einem Gigabyte PCI-Bus-Board mit 256 kByte Cache. Außerdem sind 32 MBytes Hauptspeicher, eine 1 GByte-Festplatte sowie eine SMC-Ultra-Ethernet-Netzwerkkarte installiert. Für die parallele Hochleistungskommunikation besitzt jeder Rechner eine ParaStation-Karte. Da die Bedienung über Hostrechner erfolgt und für die Inbetriebnahme eine textbasierte Terminalansicht völlig ausreichend ist, wurde auf die Integration einer leistungsfähigen Grafikkarte verzichtet. Als Betriebssystem wird Suse-Linux in der Version 4.2 verwendet, das einen Kernel von 2.0.0 enthält.

Gemäß den Leistungsmerkmalen der ParaStation-Karte können die Topologien bidirektionaler Ring und unidirektionales Gitter aufgebaut werden. Aus Abbildung 1 ist die gewählte Topologie zu entnehmen.

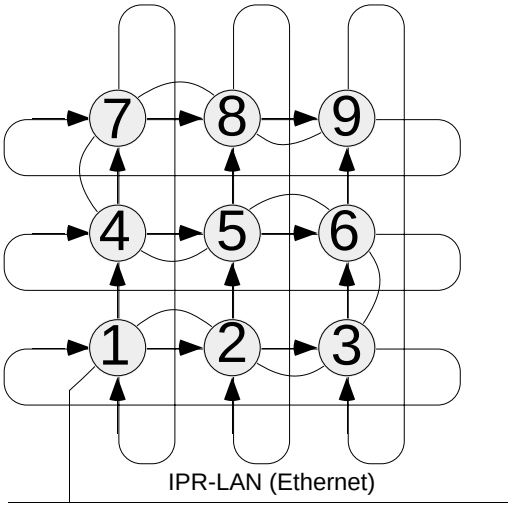

Abbildung 1: Die Verbindungstopologie der IPR-ParaStation mit neun Standard-PCs (2D-Torus) und die Anbindung an das Hausnetz über Ethernet.

Zusätzlich zur Vernetzung mit der ParaStation-Karte wurden die einzelnen Knoten über Ethernet zu einem lokalen Netz zusammengeschlossen. Um die IPR-ParaStation von der hohen Netzbelastung des Institutsnetzes abzuschotten enthält der Knoten 1 eine zweite Netzwerkkarte und muß somit als sogenannter *Proxy-ARP-Server*[2] parametriert werden. Auf diese Weise ist es dennoch möglich alle Knoten vom Institutsnetz aus anzusprechen, obwohl nur ein Knoten physikalisch mit dem Hausnetz verbunden ist. Für die Konfiguration eines Rechners als Proxy-ARP-Server müssen zwei wesentliche Punkte beachtet werden:

- Da der Proxy-ARP-Server zwei Netzwerkkarten besitzt, muß eindeutig definiert werden, welche Nachrichten über welche Karte geschickt werden (Routing).
- Da auch Nachrichten von außen an einen Rechner innerhalb des lokalen Netzes der IPR-ParaStation und umgekehrt geschickt werden sollen, muß der Proxy-ARP-Server die Rolle des Vermittlers übernehmen und die Nachrichten weiterleiten.

[2] ARP = Address Resolution Protocoll

Das Routing stellt somit einen zentralen Mechanismus dar. Die Aufgabe des Routing besteht darin, einen geeigneten Weg in einem stark verzweigten Netz zwischen Sender und Empfänger zu finden. Konkret bedeutet dies für die IPR-ParaStation, daß beim Hochfahren des Knoten 1 folgende Anweisungen ausgeführt werden müssen:

```
route add default gw 129.13.7.70
route add -host i60ps2 gw 129.13.7.71
route add -host i60ps3 gw 129.13.7.71
...
route add -host i60ps8 gw 129.13.7.71
route add -host i60ps9 gw 129.13.7.71
```

Durch die erste Zeile wird festgelegt, daß defaultmäßig alle Nachrichten über die Ethernet-Karte (Gateway) mit der IP-Adresse *129.13.7.70* (eth0) gesendet werden. Nachrichten, die für Knoten innerhalb der Subnetzes bestimmt sind, werden im Gegensatz dazu aber über die Ethernet-Karte mit der IP-Adresse *129.13.7.71* (eth1) geleitet (vgl. Abbildung 2).

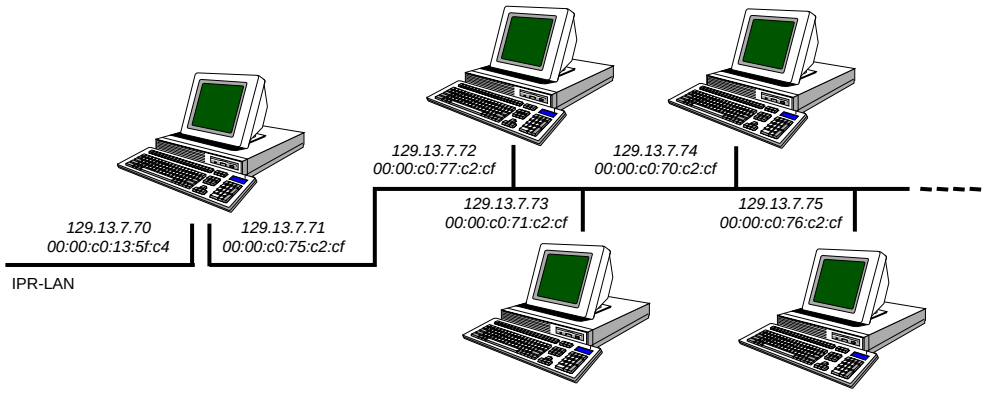

Abbildung 2: Die Konfiguration der IPR-ParaStation aus der Sicht des Haus-Netzwerkes. Die linke Workstation ist mit zwei Ethernet-Karten ausgestattet und als Proxy-ARP-Server konfiguriert.

Die zweite Eigenschaft, die ein Proxy-ARP-Server erfüllen muß, ist das Antworten auf sogenannte ARP-Anfragen, um Nachrichten nach innen oder außen über den Proxy-ARP-Server weiterleiten zu können. Will ein Rechner eine Nachricht an einen anderen Rechner verschicken, so versucht der Sender den Empfänger zunächst in seinem lokalen Netz zu finden. Hierbei sendet er einen Broadcast in das Netz, um via ARP die MAC-Adresse[3] des Empfängers herauszufinden, falls diese noch nicht in der eigenen ARP-Tabelle enthalten ist.

Ist dies erfolgreich, so können die Daten direkt über die MAC-Adresse zugestellt werden. Kann die Zustellung nicht lokal erfolgen, so wird bei richtiger Konfiguration des Proxy-ARP-Servers dieser auf den Broadcast antworten und dem Sender seine MAC-Adresse zurückschicken. Der Sender schickt die Nachricht dann an den Proxy, der diese dann an den entsprechenden Rechner außerhalb weiterleitet. Diese Funktionalität muß der Proxy-ARP-Server in beide Richtungen erfüllen, und wird bereitgestellt durch Ausführen der folgenden Anweisungen:

```
# PROXY ARP for request from the OUTSIDE to the INSIDE

/sbin/arp -s 129.13.7.70 00:00:C0:13:5F:C4 pub
/sbin/arp -s 129.13.7.71 00:00:C0:13:5F:C4 pub
/sbin/arp -s 129.13.7.72 00:00:C0:13:5F:C4 pub
....
/sbin/arp -s 129.13.7.78 00:00:C0:13:5F:C4 pub
/sbin/arp -s 129.13.7.79 00:00:C0:13:5F:C4 pub
```

---

3 MAC = Media Access Control. Die MAC-Adresse hat eine feste Länge von 48 Bit und ist weltweit eindeutig. Die ersten 24 Bit kennzeichnen den Hardware-Hersteller, wohingegen die verbleibenden 24 Bit einzelnen Geräten zugewiesen werden. In der Regel werden MAC-Adressen in hexamdezimaler Form angegeben (vgl. Abbildung 2).

```
# PROXY ARP for request from the INSIDE to the OUTSIDE

/sbin/arp -s 129.13.7.113 00:00:C0:75:C2:CF pub -i eth1
/sbin/arp -s 129.13.7.114 00:00:C0:75:C2:CF pub -i eth1
/sbin/arp -s 129.13.7.115 00:00:C0:75:C2:CF pub -i eth1
/sbin/arp -s 129.13.7.116 00:00:C0:75:C2:CF pub -i eth1
....
```

Der erste Block enthält Anweisungen, damit Knoten 1 auf Anfragen für Rechner innerhalb des Subnetzes antworten kann. Die MAC-Adresse entspricht der Adresse der Ethernet-Karte, die das Subnetz mit dem IPR-LAN verbindet. Der zweite Block besteht aus Anweisungen um den Weg nach außen zu definieren. Hierzu müssen alle Rechner des IPR-LAN angegeben werden, die man vom Subnetz aus erreichen möchte. Die MAC-Adresse gehört zur Ethernet-Karte die das Subnetz aufbaut (vgl. Abbildung 2).

Es empfiehlt sich diese Anweisungen beim Hochfahren des Proxy-ARP-Servers automatisch ausführen zu lassen. Dazu bietet es sich an, bei einem Unix-System der Version 5, die Anweisungen in eine Datei z.B. mit dem Namen */sbin/init.d/network.local* aufzunehmen, die dann von */sbin/init.d/network* aufgerufen wird.

Für eine benutzerfreundliche Umgebung und zur Vereinfachung der Systemadministration fungiert der Knoten 1 zusätzlich auch als File-Server für Programme (emacs, g++, (ps)pvm, tcl/tk etc.) und die Home-Verzeichnisse der Benutzer. Die entsprechenden Verzeichnisse werden an alle Rechner des Subnetzes exportiert, die diese dann mounten können. Entsprechende Eintragungen müssen dafür in die jeweiligen */etc/export* und */etc/fstab* Dateien vorgenommen werden.

Eine weitere Funktionalität, die ein Rechnernetzwerk erfüllen sollte, ist die konsistente User- und Paßwortverwaltung. Dies kann durch Installation von NIS (Network Information Service) - auch bekannt unter dem Namen *Yellow Pages* - erreicht werden. Auf einer dedizierten Maschine läuft ein NIS-Server der lokal die User, Paßwörter, Hosts usw. verwaltet. Alle anderen Rechner verbinden sich über einen NIS-Client mit dem Server und können auf diese Weise die korrekte Eingabe z.B. des Paßwortes überprüfen. Auf der Suse-Distribution ist allerdings nur ein NIS-Client vorhanden. Einen entsprechenden Server findet man im Internet z.B. auf dem FTP-Server *ftp.uni-paderborn.de*.

Die in der Suse-Distribution vorhandenen Werkzeuge zur Unterstützung des Systemadministrators bei der Benutzerverwaltung (useradd, usermod, userdel) unterstützen leider nicht die NIS-Funktionalität, so daß auch hier mit im Internet vorhandenen Programmen Abhilfe geschaffen werden muß.

3. Programmierumgebung

Neben der hardwaremäßigen Ausstattung der IPR-ParaStation ist die zur Verfügung stehende Software von entscheidender Bedeutung. Auf jedem Knoten ist in jedem Fall eine Minimalinstallation der Systemsoftware vorhanden. Alle weiteren Programme wie Editoren, Compiler und Grafikunterstützung werden, wie oben erwähnt, zentral verwaltet und stehen über gemountete Verzeichnisse zur Verfügung. Schwerpunkt dieses Abschnittes soll aber vielmehr die Beschreibung der Programmierumgebung für parallele Kommunikation sein, die zur optimalen Ausnutzung der Hardware notwendig ist.

Herkömmliche Message-Passing Programmierumgebungen in Workstation-Clustern, wie z.B. MPI, PVM, P4 haben zumindestens einen der folgenden Nachteile:

- Zu großen Betriebssystem-Overhead, da die Kommunikationshardware nur über das Betriebssystem angesprochen werden kann.
- Mehrere parallele Anwendungen stellen das Betriebssystem vor große Scheduling- und Synchronisationsprobleme.

- Parallele Threads pro Anwendung erhöhen den Verwaltungsaufwand für das Betriebssystem.
- Nachrichten, die nicht im Netzwerk gepuffert werden können, müssen durch Unterbrechnungsbehandlung in zentralen Puffern zwischengespeichert werden.

Bei der Entwicklung der ParaStation-Software zum Ansprechen der Hardware wurde großen Wert darauf gelegt, die oben aufgezählten Nachteile zu beseitigen oder zumindestens so klein wie möglich zu halten.

Zusammenfassend bietet die ParaStation-Systembibliothek drei verschiedene Adressierungsarten an, um auf verschiedenen Ebenen parallel kommunizieren zu können:

- Als erstes kann ein Empfängerknoten explizit adressiert werden (Rawdata-Protokoll). Hierfür muß es auf dem Empfängerknoten einen Prozeß geben, der solche Nachrichten empfängt.
- Als zweites bietet das Protokoll die Adressierung an einen *Port*. Ein Port ist eine Eingangspforte, die an einen Prozeß gebunden werden kann. Der Prozeß kann Nachrichten über diesen Port empfangen und senden.
- Die dritte Möglichkeit, die das Portieren von verteilten Anwendungen sehr erleichtert, ist eine Adressierung, die den Sockets unter BSD-Unix nachempfunden ist. Sie bietet die gleiche Semantik wie die in Unix vorhandenen Sockets und verwaltet sogar Sockets, die nicht zur ParaStation gehören. Die Programmierschnittstelle zu diesen Sockets ist identisch zu der der Unix-Sockets bis auf einen den Funktionsnamen vorangestellten Präfix PSS (für ParaStation-Sockets).

Mit Hilfe der ParaStation-Sockets wurde aus einer Standard-PVM-Version eine ParaStation-PVM-Version (PSPVM) erstellt. Der Vorteil hierbei ist, daß die Benutzerschnittstelle von PSPVM identisch mit der von Standard-PVM ist. Um in den eigenen Programmen die ParaStation-Karte zu benutzen, müssen somit keine Änderungen im Programmquelltext vorgenommen werden. Die Programme müssen nur mit der neuen Bibliothek gebunden werden.

Dadurch das die ParaStation-Systembibliothek sowohl ParaStation-Sockets als auch Unix-Sockets verwalten kann, ist es ebenso möglich ein hybrides Workstation-Cluster aufzubauen, wie es in Abschnitt 5 beschrieben wird.

4. Experimentelle Leistungsdaten der Kommunikation

Um für die verschiedenen Anwendungen in der Robotik die Kommunikationsleistung der IPR-ParaStation besser einschätzen zu können, wurde eine Reihe von Experimenten durchgeführt. Auf das Rawdata-Protokoll und die Socket-Emulation, die der UNIX-Socket-Schnittstelle entspricht, soll hier nicht näher eingegangen werden. Keine der beide verfügt über die für die meisten Anwendungen benötigte Funktionalität. Die experimentelle Auswertung der Kommunikationsschnittstellen umfaßt daher nur die Port-Adressierung, die PVM- sowie die PSPVM-Version.

Mit Hilfe des Bechmark-Programmes „Pairwise-Exchange“ aus [Warschko95] können die Geschwindigkeitsunterschiede zwischen den verschiedenen Kommunikationsarten aufgezeigt werden. Dabei werden von zwei parallelen Prozessen in mehreren Durchläufen jeweils eine Nachricht verschickt und empfangen. In unseren Experimenten beträgt die Größe der Nachrichten zwischen 1 und 1024 Bytes. Für eine genauere Zeitbestimmung werden jeweils mindestens 1.000 Durchläufe pro Nachrichtengröße ausgeführt und gemittelt.

In Abbildung 3 und 4 sind die experimentellen Ergebnisse für die Übertragungszeiten und den Datendurchsatz dargestellt. Die Ergebnisse zeigen, daß PVM mit Abstand die langsamste Kommunikation zur Verfügung stellt. Im Gegensatz dazu ist PSPVM wesentlich schneller (> Faktor 10). Zwar sind die Port- und die PSPVM-Kommunikation bezüglich den

Übertrangungszeiten relativ ähnlich, sie unterscheiden sich aber deutlich in den gemessenen Datendurchsätzen.

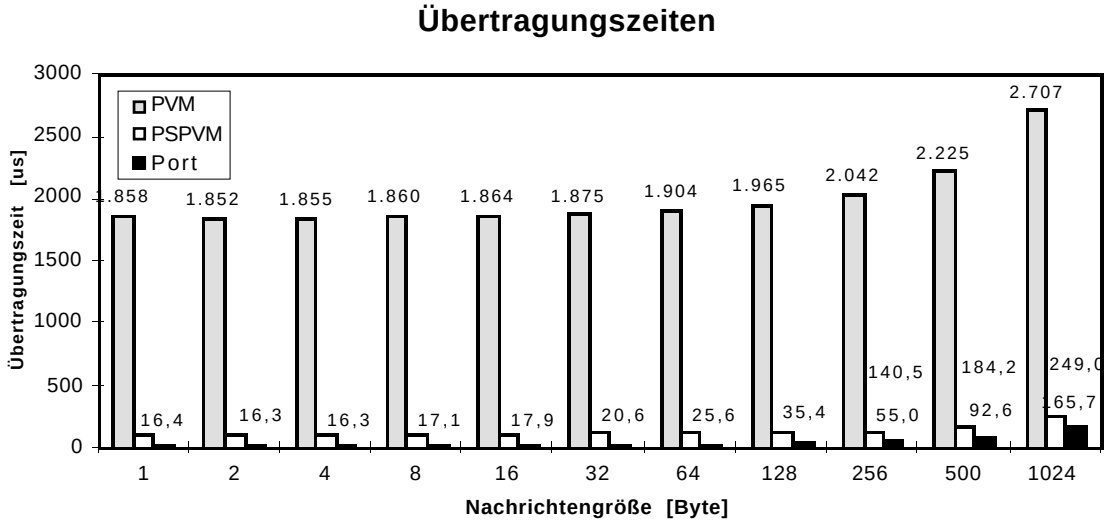

Abbildung 3: Balkendiagramm mit den gemessenen Übertragungszeiten für die unterschiedlichen Kommunikationsarten in Abhängigkeit von der Nachrichtengröße für den Benchmark „Pairwise-Exchange“

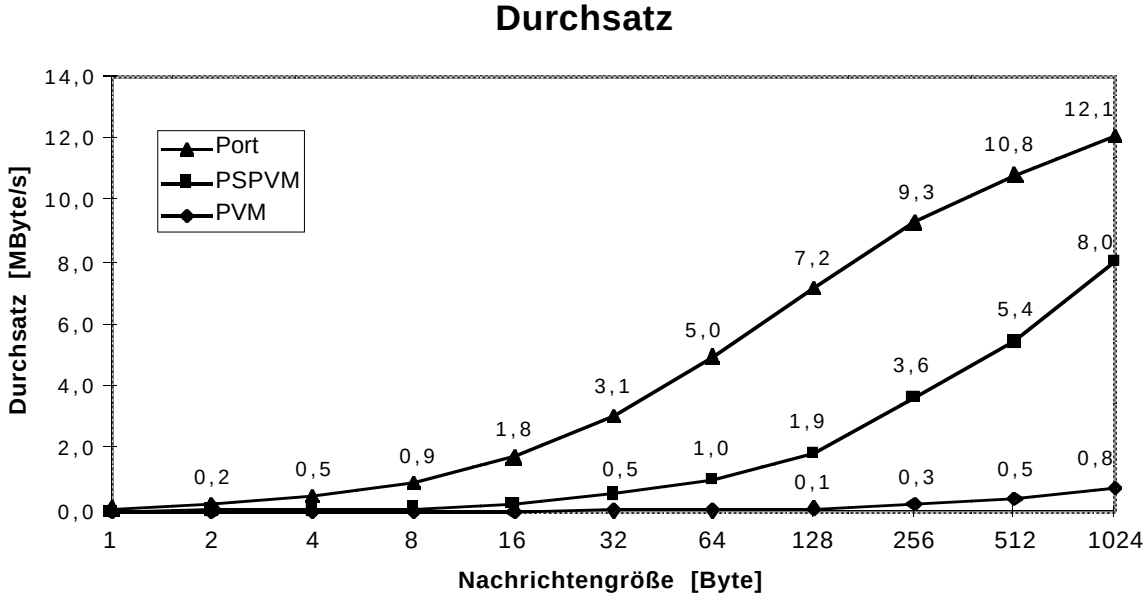

Abbildung 4: Liniendiagramm mit den gemessenen Datendurchsätzen für die unterschiedlichen Kommunikationsarten in Abhängigkeit von der Nachrichtengröße für den Benchmark „Pairwise-Exchange“

5. Parallele Bewegungsplanung in dynamischer Umgebung

Die Einsatzmöglichkeiten eines Parallelrechners in der Robotik und Automation sind sehr vielseitig. Eine erste Anwendung ist die Bewegungsplanung für Industrieroboter. Ziel der Bewegungsplanung ist es, eine kollisionsfreie Trajektorie von einer gegebenen Start- zu ei-

ner gegebenen Zielposition zu berechnen, wobei Kollisionen mit statischen aber auch dynamischen Hindernissen, die sich im Arbeitsbereich des Roboters befinden, vermieden werden müssen. In einer dynamischen Umwelt bewegen sich die Hindernisse mit einer zumeist bekannten maximalen Geschwindigkeit. Um sich dort kollisionsfrei bewegen zu können, sind reaktive Steuerungssysteme notwendig. Bei einer erkannten Kollisionsgefahr muß das Steuerungssystem ausreichend schnell reagieren können, da ansonsten die drohenden Kollisionen nicht zuverlässig vermieden werden. Von dem Bewegungsplaner sind Reaktionen innerhalb den vorgegebenen Zeitschranken notwendig. Die konventionellen sequentiellen Algorithmen sind trotz signifikanten Verbesserungen der Verarbeitungsgeschwindigkeiten der Prozessoren weit entfernt davon, echtzeitnahe Ergebnisse für fortgeschrittene Planungsaufgaben zu berechnen. Eine effiziente Parallelisierung kann eine lineare Beschleunigung bewirken, die dann eine Bewegungsplanung in industriellen Anwendungen ermöglicht.

Von den bisher untersuchten Ansätzen zur Bewegungsplanung scheint vor allem eine effiziente Suche im diskretisierten Gelenkwinkelraum die oben genannte Aufgabe zu erfüllen. Dabei werden die Winkelbereiche der einzelnen Gelenke in eine feste Anzahl von Intervallen unterteilt. Von der Startkonfiguration des Roboters aus betrachtet, wird ein Suchgraph mit den möglichen Wegen zu der gegebenen Zielkonfiguration aufgespannt. Zur Überprüfung ob eine Zelle frei ist oder nicht, wird bei jeder hinzukommenden Nachfolgerkonfiguration der kürzeste Abstandsvektor zwischen den Kollisionsklassen berechnet [Henrich92]. Eine Beschleunigung des oben beschriebenen Ansatzes kann durch den effizienten Einsatz von skalierbarer Parallelverarbeitung in Kombination mit roboterspezifischen Heuristiken bzw. Wissen erreicht werden [Henrich96].

In [Sandmann97] wird ein paralleler Bewegungsplaner auf der Basis des A*-Algorithmus entwickelt. Durch zyklische Einteilung des 6-dimensionalen Gelenkwinkelraums in Hyperkuben kann eine Lastverteilung und damit parallele Suche in diesem Raum realisiert werden. Die Ansteuerung des parallelen Bewegungsplaners erfolgt über eine selbstentwickelte Applikation mit dem Robotersimulationswerkzeug ROBCAD auf einer Silicon Graphics (SGI). Mit Hilfe dieser Entwicklungsumgebung ist es möglich, eine Arbeitszelle aus Robotern und Hindernissen im 3D-Raum aufzubauen und mögliche Start- und Zielpositionen zu parametrieren. Aus dieser Benutzeroberfläche heraus kann der parallele Bewegungsplaner auf der IPR-ParaStation gestartet werden. Nach erfolgreicher Suche wird das Ergebnis in ROBCAD animiert dargestellt und analytisch ausgewertet [Katz97].

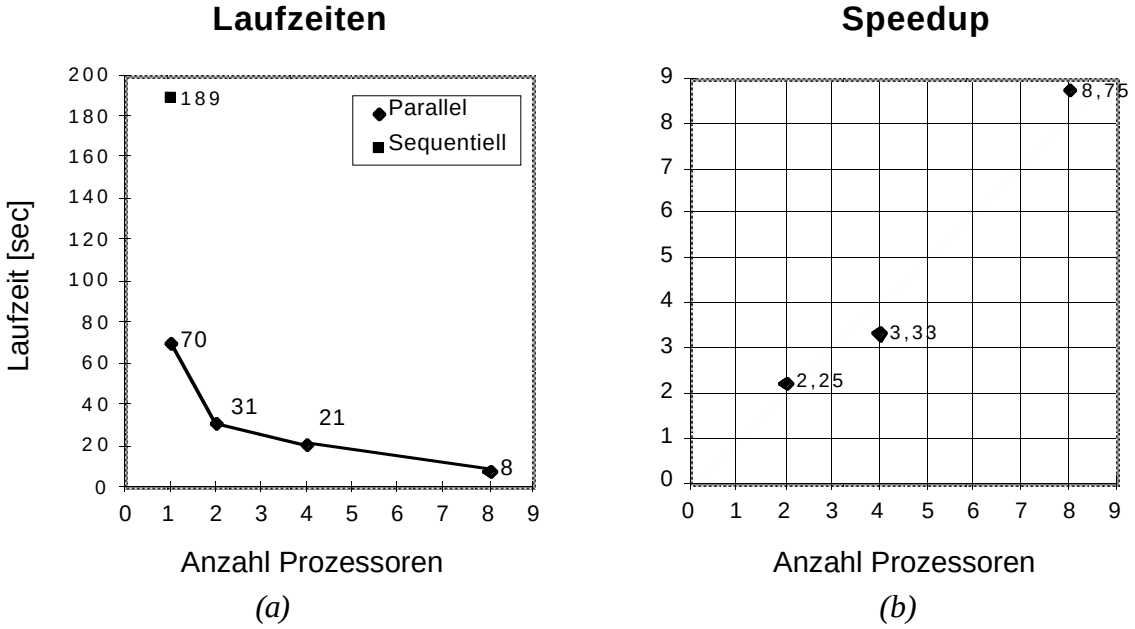

Abbildung 5: Laufzeiten (a) und Speedup (b) der parallelen Bewegungsplanung auf der IPR-ParaStation für Benchmark "STAR"

Erste Ergebnisse für exemplarische Benchmark-Umgebungen lassen erkennen, daß durch den Einsatz der parallelen Datenverarbeitung durchaus eine Beschleunigung der Laufzeit möglich ist. In Abbildung 5 sind für den Benchmark "STAR" (siehe Abbildung 6) die Laufzeiten für den sequentiellen und parallelen Fall dargestellt. Da das Laufzeitverhalten stark abhängig ist von der Aufgabenstellung und der Konfigurationsraumzerlegung, wurden zur Bestimmung der Speedup-Kurve jeweils 8 Prozesse auf 1, 2, 4 und 8 Rechnerknoten gestartet. Die Komplexität der Bewegungsplanung ist so groß, daß eine Parallelisierung in der Regel einen linearen Speedup erreicht, wie es in Abbildung 5b zu erkennen ist.