Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

MATTHIAS LINKE[1] CARSTEN KREBS[2]
MARION SCHÜNEMANN[3]

Metacomputing zwischen Berlin und Rostock

[1] Universität Rostock
[2] Konrad-Zuse-Zentrum für Informationstechnik Berlin
[3] Konrad-Zuse-Zentrum für Informationstechnik Berlin

Preprint SC 99-35 (Oktober 1999)

Metacomputing zwischen Berlin und Rostock

Matthias Linke
Universität Rostock Rechenzentrum
Albert-Einstein-Str. 21 D-18059 Rostock
Email: matthias.linke@rz.uni-rostock.de

Carsten Krebs
Konrad-Zuse-Zentrum für
Informationstechnik Berlin
Takustr. 7 D-14195 Berlin
Email: carsten.krebs@team-konzept.de

Marion Schünemann
Konrad-Zuse-Zentrum für
Informationstechnik Berlin
Takustr. 7 D-14195 Berlin
Email: schuenemann@zib.de

12. Oktober 1999

Zusammenfassung

Das Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB) und das Rechenzentrum der Universität Rostock, beide Einrichtungen betreiben massiv-parallele Rechner vom Typ CRAY T3E, haben sich an Hand einer Anwendung aus dem Bereich der Botanik mit den Möglichkeiten der Verteilung auf zwei Parallelrechner, gekoppelt nur über das Breitband-Wissenschaftsnetz des Deutschen Forschungsnetzes, vertraut gemacht.
Das Projekt besteht aus zwei Teilen. Im ersten Teil wird ausführlich das zwischen den Rechnern liegende Netz und die Ergebnisse einer Performanceanalyse dieses Netzes dokumentiert. Der zweite Teil beschreibt die Anwendung „Simulation des Wachstums von Pflanzen“ sowie die Ergebnisse der Verteilung dieser Anwendung auf beide Rechner.

Abstract

The “Konrad-Zuse-Zentrum für Informationstechnik Berlin“ (ZIB) and the Computing Centre of the University of Rostock operate high-performance computers of type CRAY T3E. In this project an application from the area of the botany is distributed on two parallel computers, only coupled through the Broadband Science Network of the German Research Network (DFN).
This project consists of two parts. In the first part the network which is situated between the computers and the results of a performance analysis is documented in detail. The second part describes the application “Simulation of the growth of plants“ as well as the results of the distribution of this application to both computers.

Inhaltsverzeichnis

1 Einleitung

Obwohl sich die Leistungsfähigkeit von Rechnern etwa alle 18 Monate verdoppelt, können genügend viele Probleme noch nicht oder nur unter großen Anstrengungen auf den derzeit vorhandenen extrem leistungsfähigen Rechnern, zum Beispiel massiv-parallelen Rechnern, bearbeitet werden. Eine Methode, um nicht auf die nächste Generation von leistungsfähigeren Rechnern warten zu müssen, liegt in der gleichzeitigen Abarbeitung von Teilen einer Anwendung auf mehreren Rechnern, so daß die gesamte Anwendung in kürzerer Zeit abgearbeitet ist. Diese Methode wird mit „Metacomputing“ bezeichnet. Der Begriff „Metacomputing“ hat noch keine klaren Abgrenzungen: Man kann darunter sowohl das Rechnen auf zusammengeschalteten Computern innerhalb eines lokalen Netzes als auch zwischen verschiedenen Organisationen über große Entfernungen hinweg, selbst über Kontinente hinweg, verstehen. Die beteiligten Rechner können vom selben Typ sein („homogenes Metacomputing“) oder unterschiedlich sein („inhomogenes Metacomputing“).

Von besonderem Interesse sind Metacomputing-Anwendungen, die in einer parallelisierten Version vorliegen und auf massiv-parallelen Rechnern bearbeitet werden. Läuft die Anwendung zum Beispiel auf 100 Prozessoren eines Parallelrechners, so sollte sie auch auf je 50 Prozessoren zweier gekoppelter Parallelrechner abgearbeitet werden können. Neben einer Reihe von organisatorischen Fragen (wie erreicht man zum Beispiel, daß auf beiden Rechner der Auftrag zur selben Zeit gestartet wird) ist die Kommunikationszeit zwischen beiden Rechnern ein zentraler Punkt des Erfolgs einer solchen Verteilung: Die Zeit für die Kommunikation zwischen einzelnen Prozessoren eines Rechners (am Beispiel einer CRAY T3E) liegt in der Größenordnung von 1 μs, die für zwei entfernt in Deutschland über das Wissenschaftsnetz des DFN verbundene Rechner in der Größenordnung von 10 ms, also 10000-mal so viel. In dieser Zeit kann ein einzelner Prozessor bereits bis zu 9 Millionen Gleitkommaoperationen (Cray T3E 900) durchführen!

An diesem drastischen Zahlenverhältnis wird deutlich, daß Metacomputing-Anwendungen mit entfernt aufgestellten Rechnern selbst bei optimalen Netzverbindungen nur dann erfolgreich eingesetzt werden können, wenn die Anwendung geeignet in Teile mit geringerem Kommunikationsbedarf untereinander gegliedert werden kann.

Das Konrad-Zuse-Zentrum für Informationstechnik (ZIB) und das Rechenzentrum der Universität Rostock, beide Einrichtungen betreiben massiv-parallele Rechner vom Typ CRAY T3E, haben sich an Hand einer Anwendung aus dem Bereich der Botanik mit den Möglichkeiten der Verteilung auf zwei Parallelrechner, gekoppelt nur über das Breitband-Wissenschaftsnetz des Deutschen Forschungsnetzes, vertraut gemacht. Im folgenden Kapitel 2 werden die verwendeten Rechner sowie das dazwischen liegende Netz beschrieben. Kapitel 3 beschreibt ausführliche Performanceanalysen des Netzes zwischen Berlin und Rostock. Kapitel 4 beschreibt die Anwendung „Simulation des Wachstums von Pflanzen“ sowie die Ergebnisse der Verteilung der Simulation auf beide Rechner.

Obwohl beide Einrichtungen diese Studie gemeinsam durchführten, lagen die Schwerpunkte der Arbeit des ZIB bei der Performanceanalyse des Netzes und die Schwerpunkte der Arbeit der Universität Rostock bei der Verteilung der Simulation auf beide Rechner und den durchgeführten Rechenläufen.

Hubert Busch (Leiter der Abteilung Höchstleistungsrechner im ZIB),
Christa Radloff (Leiterin des Rechenzentrums der Universität Rostock)

2 Rechnersysteme und Netztopologie

Carsten Krebs, Marion Schünemann

2.1 Verwendete Systeme

Sowohl im ZIB als auch im Rechenzentrum der Universität Rostock kam jeweils eine *Cray T3E* zum Einsatz. Identisch auf beiden Parallelrechnern war das verwendete Betriebssystem *UNICOS/mk* und der Typ der verwendeten Prozessorelemente (*DEC Alpha EV5.6*). Unterschiede weisen die beiden Rechner dagegen bei der Anzahl der verwendeten Prozessorelemente (PEs) auf. So ist der Rechner des ZIB (`berte`) mit 272 Prozessorelementen bestückt. Von diesen 272 PEs stehen 256, mit jeweils 128 MByte Hauptspeicher, für parallele Applikationen zur Verfügung. Die restlichen 16 PEs, davon 8 mit jeweils 128 MByte und 8 mit jeweils 256 MByte Hauptspeicher, stehen dem Betriebssystem, single PE Applikationen und interaktiven Kommandos (Shell, Compiler, etc.) zur Verfügung. Der Parallelrechner der Universität Rostock (`cros`) dagegen ist lediglich mit 44 PEs, mit je 128 MByte Hauptspeicher, bestückt. Von diesen 44 PEs stehen 40 für parallele Applikationen, ein PE dem Betriebssystem und die restlichen drei PEs den interaktiven Kommandos zur Verfügung. Ausführliche Daten zu den Rechnersystemen sind dem Anhang zu entnehmen.

2.2 Netzumgebung

Die Verbindung zwischen dem ZIB und Rostock wird über das Breitband-Wissenschaftsnetz (B-WiN) des DFN-Vereins hergestellt. Das B-WiN wird als privates virtuelles Netz (VPN) auf dem ATM-Cross-Connect-Netz der Deutschen Telekom AG betrieben. Die Anbindung an den Berliner 155 Mbit/s B-WiN-Gemeinschaftsanschluß erfolgt mittels eines sogenannten ATM Kunden Service Switches (KSS) über PVCs (Permanent Virtual Circuits). Dem ZIB selber standen von den 155 Mbit/s Bandbreite des Gemeinschaftsanschlusses bis Mitte Januar 1998 lediglich 12 Mbit/s, danach nur noch 10 Mbit/s zur Verfügung. In dem gleichen Zeitraum stellte der DFN-Verein das Routing um, so daß die vorher direkte Verbindung zwischen Berlin und Rostock (Abbildung 1) durch eine über den B-WIN-Backbone Berlin-Hamburg geroutete Verbindung ersetzt wurde (Abbildung 2). Seit Ende März 1999 stehen dem ZIB wieder 12 Mbit/s zur Verfügung. Der B-WiN-IP-Dienst wird über ATM unter Nutzung von *Multiprotocol Encapsulation over AAL5 (Encapsulation)* [RFC1483] realisiert.

Die Netzstruktur in der Uni Rostock kann Abbildung 3 entnommen werden. Der Rechner `cros` ist über einen Cisco LS1010 mit dem Uni Router (Cisco 7500) über einen 155 Mbit/s ATM-PVC verbunden. Zwischen dem Uni Router und dem DFN Router (Cisco 7500) ist ein 106 Mbit/s PVC geschaltet. Der DFN Router ist über einen GDC Apex/Mac mit dem WIN-Router verbunden.

Die Verbindung zwischen dem KSS des B-WiN und dem Rechner `berte` kann über zwei zum Teil verschiedene Wege erfolgen. Einerseits ist `berte` nur über das lokale ATM-Netz und andererseits über einen FDDI-Ring und einen Teil des ATM-Netzes mit dem KSS verbunden. Es wurden bei den, im Rahmen dieses Projekts durchgeführten Messungen, alle IP-Pakete von `berte` nach `cros` über den FDDI-Ring versendet. In umgekehrter Richtung wurden IP-Pakete, die von `cros` nach `berte` versendet wurden, sowohl über den FDDI-Ring als auch über das lokale ATM-Netz geroutet, gemessen.

Die Netzstruktur im ZIB ist in Abbildung 4 dargestellt. Die Verbindung zwischen `berte` und dem KSS über den FDDI-Ring sieht im einzelnen wie folgt aus. `Berte` ist über den FDDI-Ring (100 Mbit/s) mit einem Cisco 7513/4 Router (`rout1`) verbunden. Von diesem wird über einen Cisco LS100 ATM-Switch eine 155 Mbit/s-Verbindung zum KSS realisiert. Die ATM-Verbindung zwischen KSS und `berte` unterscheidet sich von der FDDI Anbindung in der Überbrückung der Strecke zwischen dem rout1 und `berte`. Wie in Abbildung 5 dargestellt, ist der rout1 mit 155 Mbit/s mit einem Cisco LS1010 ATM-Switch verbunden, dieser ist mit 622 Mbit/s mit einem weiteren Cisco LS1010 ATM-Switch verbunden, welcher wiederum mit 155 Mbit/s die Verbindung zum ATM-Interface von `berte` herstellt. Sämtliche ATM-Verbindungen zwischen KSS und `berte` werden dabei über PVCs realisiert. Auf IP-Ebene ist berte mit 109 Mbit/s von rout1 bis zum WIN-Router verbunden und zum Router `rout1` mit 100 Mbit/s dedizierter Bandbreite (Abbildung 6). Der IP-Dienst zwischen `berte` und dem Router `rout1` über ATM wird mittels Encapsulation realisiert.

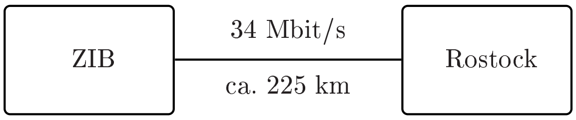

Abbildung 1: Verbindung zwischen ZIB und Rostock

Abbildung 2: Verbindung zwischen ZIB und Rostock über Hamburg

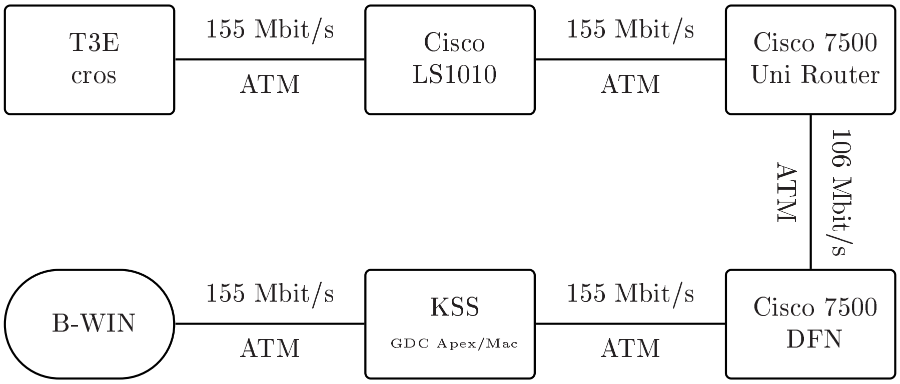

Abbildung 3: Netzstruktur in der Uni Rostock

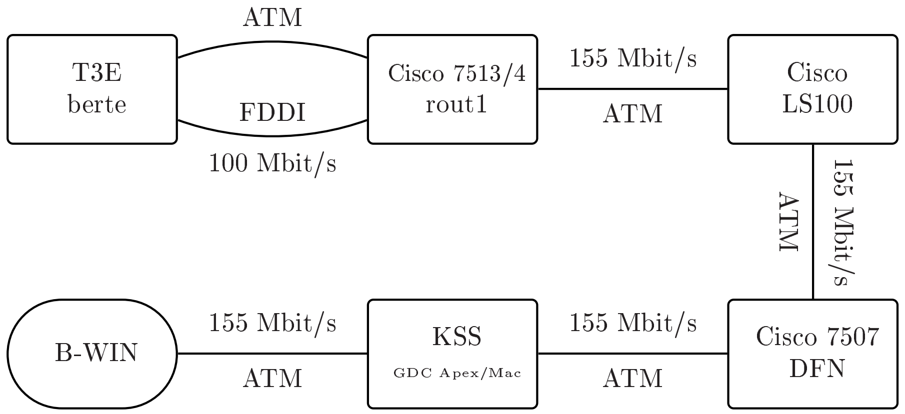

Abbildung 4: Netzstruktur im ZIB

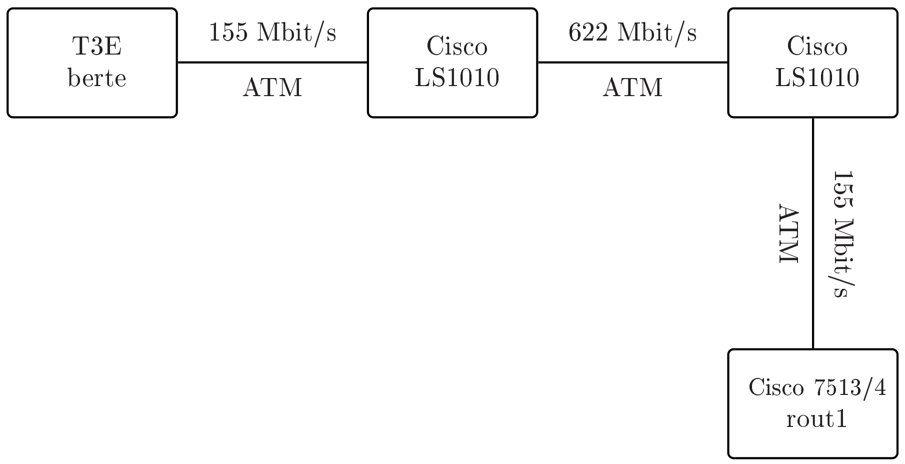

Abbildung 5: ATM-Weg im ZIB

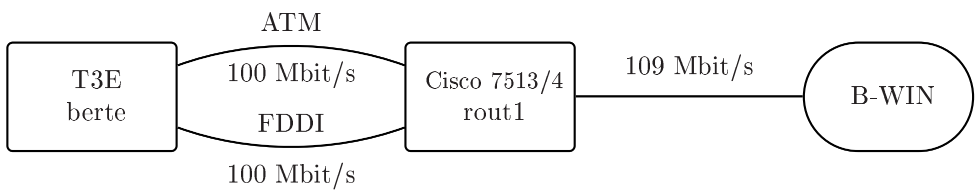

Abbildung 6: Dedizierte Bandbreite für IP-Verkehr im ZIB

3 Performanceanalyse der Netzwerkumgebung

Carsten Krebs, Marion Schünemann

3.1 Messungen

3.1.1 Verwendete Testsoftware

Für die durchgeführten Netzwerkleistungstests wurde eine adaptierte Version des von Hewlett-Packard entwickelten Programms `netperf`[netperf] verwendet. Das Programm `netperf` ist frei verfügbar und ermöglicht es, unidirektionale Netzwerkleistungstests, u.a. für die Protokolle TCP und UDP, durchzuführen. Das Programm ist ursprünglich für 32-Bit Rechnerarchitekturen entwickelt worden und wurde von uns für die Verwendung auf der Cray T3E angepasst.

`Netperf` bietet die Möglichkeit, eine Vielzahl von Testparametern, wie z.B. die Testdauer, die Socketbuffergrößen oder die Länge der gesendeten Nachrichten zu variieren. Darüberhinaus lassen sich die Tests mit `netperf` relativ leicht automatisieren.

Wir beschränkten unsere Leistungstests auf die TCP/IP-Ebene, da die zu entwickelnde Metacomputing-Anwendung auf der Kommunikationsbibliothek *PVM (Parallel Virtual Machine)* aufsetzen sollte. Die Intertaskkommunikation zweier PVM-Tasks wird ausschließlich über das TCP/IP-Protokoll abgewickelt[PVM].

Wir führten zwei Testarten durch. Zum einen den sogenannten *TCP Stream Test* und zum anderen den *TCP Request-Response Test.* Beim *TCP Stream Test* wird der über die Testdauer gemittelte Datendurchsatz eines TCP Datenstromes ermittelt. Beim *TCP Request-Response Test* dagegen wird ermittelt, wieviele Anfragen das entfernte (remote) System pro Zeiteinheit im Mittel beantworten kann. Die Anfragen werden vom lokalen System nacheinander und nicht nebenläufig gestellt. Die *Transaktionsperformance* für leere Anfragen/Antworten diente uns als Maß für die Latenzzeit (*round-trip-time*) der Verbindung.

3.1.2 Durchführung der Messungen

Problematisch bei der Durchführung unserer Messungen war der Umstand, daß sie in einer nicht isolierten Netzwerkumgebung stattfinden mußten. Das hatte zur Folge, daß eine einzelne Messung keine brauchbare Aussage über das verwendete Medium liefern kann, da sie von einer Vielzahl von Parametern beeinflußt wird, wie z.B. der Systemlast oder durch dritte erzeugte zusätzliche Netzlast.

Um dennoch brauchbare Aussagen über die Qualität des Mediums machen zu können, führten wir eine Vielzahl von Messungen durch und versuchten, die Netzwerkverbindung durch die statistischen Lage- und Streuungsparameter Maximal- und Mittelwert, sowie Minimum und Standardabweichung des Durchsatzes zu beschreiben.

Aber selbst das Durchführen von vielen Messungen stellt noch nicht sicher, daß die gesammelten Daten aussagekräftig sind. So muß z.B. darauf geachtet werden, daß die Messungen nicht immer wieder mit periodisch ausgeführten Diensten kollidiert die das Netz belasten. Es muß auch darauf geachtet werden, daß das Zeitintervall, in dem die Messungen durchgeführt werden, möglichst groß gewählt wird, zumindest aber so groß, daß sie über einen kompletten Tag möglichst gleichverteilt ausgeführt werden. So unterliegt das Netz üblicherweise zu unterschiedlichen Tageszeiten auch unterschiedlichen Lastverhältnissen.

Um diesen Anforderungen gerecht zu werden, verteilten wir unsere Messungen auf jeweils mindestens einen Tag. Jede Messung wurde zu einer bestimmten Parametereinstellung innerhalb eines Viertelstundenintervalls zu einem zufälligen Zeitpunkt ausgeführt. Innerhalb einer Viertelstunde wurde somit für jede interessierende Parametereinstellung eine Messung zu einem zufälligen Zeitpunkt durchgeführt. Die Dauer einer Messung versuchten wir möglichst kurz zu wählen, um so etwas wie eine *Momentaufnahme* vom Zustand des Mediums machen zu können. Konkret wählten wir für den *TCP Stream Test* eine Intervallänge von 10 Sekunden für einen Test und für den *TCP Request-Response Test* eine Intervallänge von 30 Sekunden.

Führt man eine Vielzahl von Messungen durch, so hat das den Nachteil, daß man nicht mehr die Möglichkeit hat, in akzeptabler Zeit eine größere Anzahl von Parametereinstellungen systematisch auf ihre Brauchbarkeit hin zu testen. Bei unseren Tests beschränkten wir uns deshalb neben den Defaulteinstellungen des Systems auf einige wenige Parametereinstellungen für die Größe der gesendeten Nachrichten (*Messagesize*) und der Socketbuffer (*Socketbuffersize*). Insgesamt wurden folgende Parameterkombinationen getestet:

Socketbuffersize	**Messagesize**		
	64 KByte	32 KByte	16 KByte
64 KByte	X	X	
32 KByte		X	X

3.2 Theoretische Leistungsdaten

Um die im folgenden vorgestellten Leistungsdaten besser einschätzen zu können, soll hier eine obere Schranke für die Nutzlast einer TCP/IP über ATM Übertragung angegeben werden. Eine obere Schranke für die Nutzlast läßt sich berechnen, indem man den in den einzelnen Protokollschichten entstehenden Protokolloverhead mit den übertragenen Nutzdaten ins Verhältnis setzt[ATM].

Von den einzelnen Protokollschichten verfallen auf der ATM-Ebene auf eine 53 Byte große ATM-Zelle 5 Bytes auf den Header. Für *Multiprotocol Encapsulation over AAL5* [RFC1483] kommen 8 Bytes Header und für AAL5 8 Bytes für den AAL5 Trailer hinzu. Um für die AAL5-Ebene die Größe der PDU (Protocol Data Unit) durch 48 teilbar zu machen, kommen dann nochmal, bei einer MTU (Maximum Transmission Unit) von 4470 Bytes, 26 Bytes Füllzeichen, hinzu. Der IP-Header ist 20 Bytes groß. Von den 4470 Bytes der IP-SDU (IP Service Data Unit) verfallen 20 Bytes auf den TCP-Header.

Insgesamt reduziert sich die nutzbare Bandbreite durch den Protokolloverhead damit durch ...

TCP	um ca. 0,45%
IP	um ca. 0,45%
Encapsulation/AAL5	um ca. 0,8%
ATM	um ca. 9,43%

Damit ergibt sich für **12 Mbit/s** nominale Bandbreite mit TCP/IP über ATM **10,68 Mbit/s** als obere Schranke und für eine nominale Bandbreite von **10 Mbit/s** verbleiben **8,9 Mbit/s** .

Bei Wahl der Verbindung im ZIB über den FDDI-Ring erhält man einen zusätzlichen FDDI-Overhead von 30 Byte und der AAL5-Overhead beträgt 12 Byte, damit reduziert sich die Bandbreite (bei einer MTU von 4352 Byte) durch ...

TCP	um ca. 0,46%
IP	um ca. 0,46%
FDDI	um ca. 0,68%
Encapsulation/AAL5	um ca. 0,27%
ATM	um ca. 9,43%

Damit ergibt sich über den FDDI-Ring für **12 Mbit/s** nominale Bandbreite mit TCP/IP über ATM **10,66 Mbit/s** als obere Schranke und für eine nominale Bandbreite von **10 Mbit/s** verbleiben **8,88 Mbit/s** .

3.3 Meßergebnisse

Die Messungen zwischen Rostock und Berlin wurden in vier verschiedenen Zeiträumen, jeweils unter leicht verschiedenen Randbedingungen, durchgeführt.

3.3.1 4. Quartal 1997

Ende 1997 stand über die direkte B-WiN-Verbindung zwischen dem ZIB und der Universität Rostock noch eine Bandbreite von 12 Mbit/s zur Verfügung. Getestet wurden drei unterschiedliche Wege:

1. Von `berte` erst über den FDDI-Ring des ZIBs, dann über das B-WiN bis zu `cros`
2. Von `cros` erst über das B-WiN, dann über den FDDI-Ring des ZIBs bis zu `berte`
3. Von `cros` erst über das B-WiN, dann über das lokale ATM-Netz des ZIBs bis zu `berte`

Getestet wurde der *TCP Stream*-Durchsatz und die Latenzzeit (*TCP Request-Response Test*). Die Ergebnisse sind in den Tabellen 1-4 zusammengefaßt.

Auffällig an den Ergebnissen ist, daß das lokale ATM-Netz im ZIB einen erheblichen Einfluß auf den Durchsatz hat. So liegt der über den FDDI-Ring erzielte Spitzenwert für den Durchsatz noch erstaunlich dicht an der theoretischen oberen Schranke (s. S. 12). Wurde jedoch über das lokale ATM-Netz des ZIB geroutet, so konnte lediglich ein um ca. 15% niedrigerer Spitzenwert erzielt werden. Dieses ist insofern merkwürdig, da zum einen der Großteil der Strecke für beide Routen identisch ist und zum anderen das lokale ATM-Netz des ZIB eine 10fach höhere Bandbreite aufweist als die Route über das B-WiN.

Anhand der Mittelwerte läßt sich sehen, daß die Größe der Socketbuffer einen größeren Einfluß auf den Durchsatz hat als die Wahl der Nachrichtengröße.

Wie anhand des in Bild 7 dargestellten Boxplots[1] zu sehen ist, konzentrieren sich in beiden Fällen die Durchsatzraten auf einen relativ kleinen Bereich. So lagen für die Strecke von `cros` nach `berte` über das lokale ATM-Netz des ZIB 75% der Meßpunkte in dem Bereich zwischen 8,47 Mbit/s und 8,76 Mbit/s. Für die Strecke von `cros` nach `berte` über den FDDI-Ring des ZIB lagen 75% der Meßpunkte immerhin noch zwischen 9,23 Mbit/s und 10,32 Mbit/s.

3.3.2 1. Quartal 1998

Problematisch bei der Verwendung von `netperf` auf den T3Es ist, daß das Programm nur auf interaktiven PEs läuft. Dies hat zum Nachteil, daß die von anderen Benutzern verursachte CPU-Last sich eventuell negativ auf die Messungen auswirken könnte. Um einen Vergleich zu den vorherigen Messungen zu bekommen, wurden einige Messungen zeitgleich mit Wartungsarbeiten auf `berte` durchgeführt, da zu diesem Zeitpunkt der Benutzerbetrieb stark eingeschränkt war.

Wegen der sehr begrenzten Zeit konnten jedoch nur sehr wenige Daten gesammelt werden. Die Messungen beschränken sich deshalb auch nur auf eine Socketbuffer- und Nachrichtengröße von 64 KByte. Während dieser Messungen bestand noch die 12 Mbit/s Verbindung zwischen Berlin und Rostock. In Tabelle 5 sind die Ergebnisse dargestellt.

Neue Erkenntnisse konnten aus diesem Test leider nicht gefolgert werden.

3.3.3 3. und 4. Quartal 1998

Im 3. und 4. Quartal wurden die Tests zwischen den beiden Einrichtungen nochmals wiederholt. Dies bot sich aus zwei Gründen an. Einmal kam in der Zwischenzeit in beiden T3Es ein neues ATM-Interface (FORE Runner 200E, SBUS) zum Einsatz, und zum anderen hatte sich die Route über das B-WiN verändert. So wurde anstatt der direkten Route zwischen Berlin und Rostock nun über Hamburg geroutet. Die Bandbreite der Route verringerte sich gleichzeitig auf 10 Mbit/s. Es war deshalb zu erwarten, daß sich die nun mehr als doppelt so lange Route auch in der Latenzzeit (*round-trip-time*) niederschlagen würde. Die Ergebnisse sind in den Tabellen 6-9 dargestellt.

Der noch im 4. Quartal 1997 auffällige Unterschied zwischen der Route über das lokale ATM-Netz und dem FDDI-Ring im ZIB tritt hier offensichtlich nicht mehr auf. Es scheint also, als ob sich mit dem Einbau der neuen ATM-Karte in `berte` dieses Problem gelöst hätte.

Bei Betrachtung der Latenzzeiten fällt auf, daß sich hier, wie bereits vermutet wurde, durch das neue Routing erhebliche Verschlechterungen ergeben haben. So beträgt die Steigerung im Vergleich zur direkten Route ca. 60%. Die Lichtgeschwindigkeit beträgt in Glas ca. 200 km/ms. Mit diesem Wert erhöht sich die Latenzzeit theoretisch alleine durch die längere Dauer des Lichtes um 2,6 ms. Die zusätzlichen Router, die auf der neuen Strecke durchlaufen werden können ebenfalls eine erhöhende Auswirkung auf die Latenzzeit haben.

[1] Die mittleren Balken repräsentieren den Median. Untere und obere Begrenzungen der Kästchen geben das Percentil 25 und 75 wieder. Die Endbegrenzungen markieren die Extremwerte.

Tabelle 1: `cros` $\rightarrow$ `berte` (ATM) (4. Quartal 1997)

Socketbuffersize	**Messagesize** 64 KByte	32 KByte	16 KByte	
64 KByte	8,76	8,76		Maximum
	$8,08 \pm 0,15$	$8,09 \pm 0,15$		Mittelwert
	0,94	2,23		Minimum
32 KByte		8,39	8,42	Maximum
		$7,56 \pm 0,07$	$7,56 \pm 0,10$	Mittelwert
		1,46	3,47	Minimum

Alle Angaben in Mbit/s

Tabelle 2: `cros` $\rightarrow$ `berte` (FDDI) (4. Quartal 1997)

Socketbuffersize	**Messagesize** 64 KByte	32 KByte	16 KByte	
64 KByte	10,32	10,33		Maximum
	$9,48 \pm 0,24$	$9,57 \pm 0,22$		Mittelwert
	3,38	6,42		Minimum
32 KByte		9,61	9,63	Maximum
		$8,80 \pm 0,13$	$8,78 \pm 0,22$	Mittelwert
		2,57	3,67	Minimum

Alle Angaben in Mbit/s

Tabelle 3: `berte` (FDDI) $\rightarrow$ `cros` (4. Quartal 1997)

Socketbuffersize	**Messagesize** 64 KByte	32 KByte	16 KByte	
64 KByte	10,44	10,35		Maximum
	$9,77 \pm 0,19$	$9,72 \pm 0,22$		Mittelwert
	4,70	3,57		Minimum
32 KByte		9,87	9,88	Maximum
		$9,02 \pm 0,17$	$9,06 \pm 0,21$	Mittelwert
		0,27	4,80	Minimum

Alle Angaben in Mbit/s

Tabelle 4: Latenzzeiten (4. Quartal 1997)

	ATM `berte`	ATM `cros`	FDDI `berte`	
`cros`	13,90		13,54	Minimum
	$15,31 \pm 0,22$		$15,37 \pm 0,66$	Mittelwert
	35,35		186,57	Maximum
`berte`		13,40		Minimum
		$14,68 \pm 0,49$		Mittelwert
		197,63		Maximum

Alle Angaben in ms

Tabelle 5: TCP Durchsatz (1. Quartal 1998)

	berte (FDDI)	cros	
		9,18	Maximum
berte (FDDI)		$7,68 \pm 0,69$	Mittelwert
		6,42	Minimum
	8,80		Maximum
cros	$7,59 \pm 0,91$		Mittelwert
	5,48		Minimum

Alle Angaben in Mbit/s

Tabelle 6: cros → berte (ATM) (3. und 4. Quartal 1998)

Socketbuffersize	**Messagesize**			
	64 KByte	32 KByte	16 KByte	
	8,75	8,76		Maximum
64 KByte	$7,58 \pm 0,11$	$7,59 \pm 0,11$		Mittelwert
	1,36	1,46		Minimum
		8,05	8,02	Maximum
32 KByte		$6,20 \pm 0,07$	$6,20 \pm 0,10$	Mittelwert
		1,73	1,66	Minimum

Alle Angaben in Mbit/s

Tabelle 7: cros → berte (FDDI) (3. und 4. Quartal 1998)

Socketbuffersize	**Messagesize**			
	64 KByte	32 KByte	16 KByte	
	8,76	8,76		Maximum
64 KByte	$8,36 \pm 0,14$	$8,33 \pm 0,15$		Mittelwert
	5,44	4,60		Minimum
		7,81	7,83	Maximum
32 KByte		$6,96 \pm 0,09$	$6,98 \pm 0,13$	Mittelwert
		2,89	4,15	Minimum

Alle Angaben in Mbit/s

Tabelle 8: berte (FDDI) → cros (3. und 4. Quartal 1998)

Socketbuffersize	**Messagesize**			
	64 KByte	32 KByte	16 KByte	
	8,75	8,76		Maximum
64 KByte	$7,66 \pm 1,54$	$8,00 \pm 0,15$		Mittelwert
	1,37	3,72		Minimum
		8,09	8,00	Maximum
32 KByte		$6,70 \pm 0,11$	$6,74 \pm 0,17$	Mittelwert
		2,52	2,21	Minimum

Alle Angaben in Mbit/s

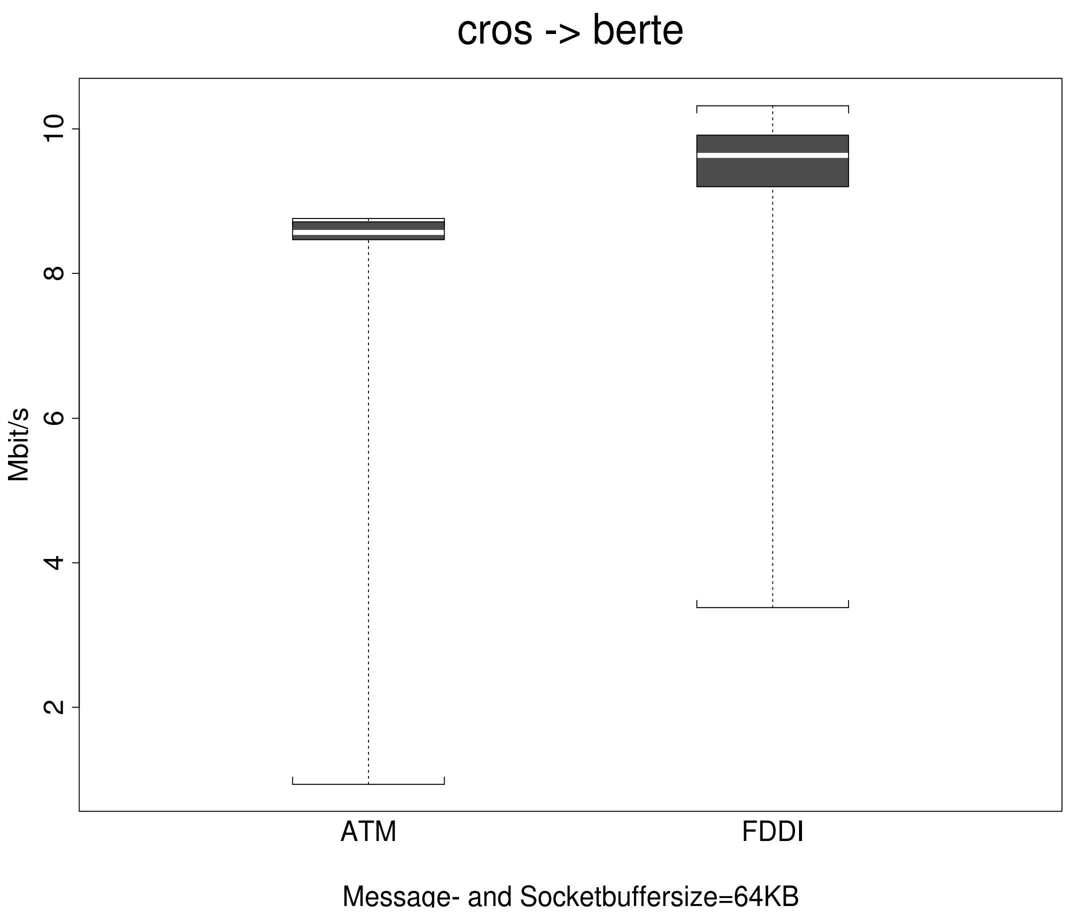

Abbildung 7: Verteilung der Durchsatzraten

Auffällig ist auch das *Ausreißerverhalten* auf der FDDI-Route. Wie auch schon bei den Messungen im Vorjahr, weisen die Latenzzeiten über den FDDI-Ring eine wesentlich höhere Spannweite auf. Die Verteilung der Latenzzeiten für beide Routen ist in Bild 8 dargestellt[2].

3.3.4 1.Quartal 1999

Ende März wurde die Bandbreite für das ZIB wieder auf 12 Mbit/s erhöht. Eine neue Messung wurde über den FDDI-Ring von `berte` aus nach `cros` und von `cros` über den ATM-Weg nach `berte` durchgeführt. Die Durchsatzraten sind in den beiden Tabellen 10 und 11 dargestellt. Die Ergebnisse zeigen, daß die Wahl des Weges im ZIB offenbar keine Rolle mehr spielt. Die Durchsatzraten, egal ob über den FDDI-Ring oder den ATM-Weg sind in etwa gleich. Die Maximalwerte liegen in beiden Fällen sehr dicht an den theoretischen Spitzenwerten (s. S. 12). Die Latenzzeiten (Tabelle 12) fallen verglichen mit der letzten Messung im 3. und 4. Quartal (Tabelle 9) besser aus. Sowohl die Durchschnitts- als auch die Minimalwerte liegen um ca. 3 ms niedriger.

3.4 Schaltung eines 2 Mbit/s-PVCs im 2. Quartal 1999

Im April 1999 wurde für unser Projekt, für die Dauer von einer Woche, ein dedizierter 2 Mbit/s-PVC zwischen den beiden T3Es im ZIB und in der Uni Rostock geschaltet. Es sollte getestet

[2] Latenzzeiten größer 50 ms wurden der besseren Darstellung halber im Bild nicht berücksichtigt.

Tabelle 9: Latenzzeiten (3. und 4. Quartal 1998)

	ATM		FDDI	
	berte	cros	berte	
cros	22,58		21,35	Minimum
	$27,34 \pm 0,38$		$24,87 \pm 0,81$	Mittelwert
	5555,56		657,80	Maximum
berte		20,62		Minimum
		$25,94 \pm 0,87$		Mittelwert
		119,76		Maximum

Alle Angaben in ms

Tabelle 10: berte (FDDI) $\rightarrow$ cros (1. Quartal 1999)

Socketbuffersize	**Messagesize**			
	64 KByte	32 KByte	16 KByte	
64 KByte	10,58	10,58		Maximum
	$10,30 \pm 0,38$	$10,31 \pm 0,38$		Mittelwert
	6,92	5,29		Minimum
32 KByte		8,96	9,12	Maximum
		$7,82 \pm 0,16$	$7,79 \pm 0,21$	Mittelwert
		4,50	5,42	Minimum

Alle Angaben in Mbit/s

Tabelle 11: cros $\rightarrow$ berte (ATM) (1. Quartal 1999)

Socketbuffersize	**Messagesize**			
	64 KByte	32 KByte	16 KByte	
64 KByte	10,49	10,50		Maximum
	$10,26 \pm 0,35$	$10,34 \pm 0,34$		Mittelwert
	8,26	9,88		Minimum
32 KByte		8,75	8,70	Maximum
		$7,74 \pm 0,19$	$7,68 \pm 0,32$	Mittelwert
		4,84	4,98	Minimum

Alle Angaben in Mbit/s

Tabelle 12: Latenzzeiten (1. Quartal 1999)

	berte	cros	
cros	18,81	2,17	Minimum
	$22,42 \pm 1,05$	$3,47 \pm 0,34$	Mittelwert
	39,86	13,68	Maximum
berte	2,44	18,68	Minimum
	$4,20 \pm 0,47$	$22,26 \pm 0,95$	Mittelwert
	16,44	28,55	Maximum

Alle Angaben in ms

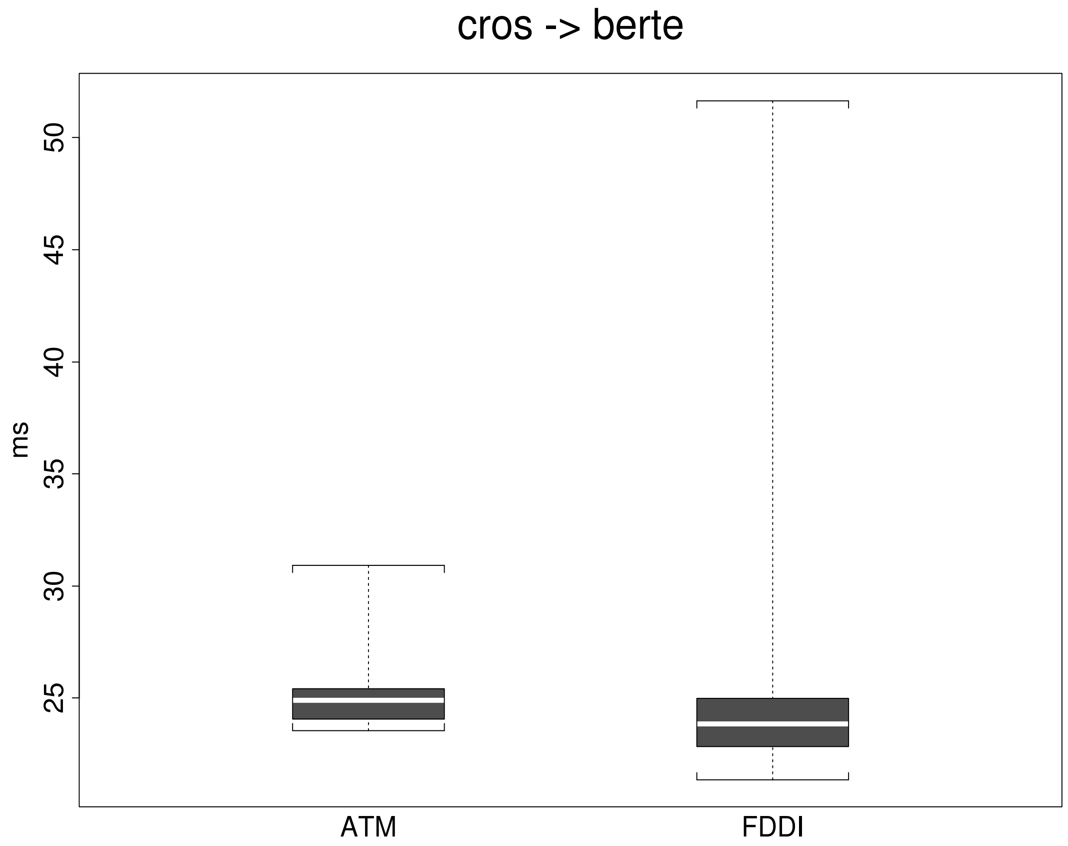

Abbildung 8: Verteilung der Latenzzeiten

werden, ob eine direkte ATM-Verbindung, über die kein fremder Datenverkehr läuft, zu besseren Ergebnissen führt. Die normale 10 Mbit/s-Verbindung läuft zwischen dem ZIB und der Uni Rostock über sechs Router, diese konnten bei der direkten Verbindung ausgelassen werden, so daß auch untersucht werden konnte, welchen Einfluß zwischengeschaltete Router auf die Latenzzeit haben.
An diesem PVC wurden ebenfalls Durchsatz- und Latenzzeitmessungen vorgenommen. Die Ergebnisse sind in den Tabellen 13-15 zusammengefaßt. Auffällig sind sofort die schlechten Durchsatzraten. Die gemessenen Maximalwerte liegen im besten Fall bei 1,31 Mbit/s, das ist überraschend wenig. Selbst spätere Messungen über den leeren PVC ergeben keine höheren Werte. Von den 2 Mbit/s Bandbreite bleiben also nur ca. 1,31 Mbit/s übrig, das sind nur 65,5%. Eine Erklärung für den hohen Verlust von Bandbreite liegt bei den internen Berechnungen der Interfaces an den T3Es, die nur mit ganzzahligen Prozent der maximalen Bandbreite von 139,5 Mbit/s (Datenrate) durchgeführt werden. Daraus ergeben sich größere Abrundungen, welche zu einer Reduzierung der Bandbreite von 2 Mbit/s auf 1,39 Mbit/s führen. Nach Abzug weiteren Protokolloverheads für TCP/IP und Encapsulation/AAL5 erhält man einen theoretischen Spitzenwert von 1,37 Mbit/s. Folglich verbleibt noch eine Differenz von 0,06 Mbit/s zwischen dem theoretisch berechneten Spitzenwert und dem gemessenen Maximalwert von 1,31 Mbit/s bestehen, das sind immerhin 3 % des 2 Mbit/s-PVCs.
Die gemessenen Latenzzeiten weisen keinen großen Unterschied zu den im 3. und 4. Quartal 1998 (Tabelle 9) gemessenen Werten auf. Die auf der 2 Mbit/s-Strecke eingesparten sechs Router gegenüber der 10 Mbit/s-Strecke, haben folglich keinen positiven Einfluß auf die Latenzzeit, was auf eine gute Konfiguration der Router schließen läßt. Wie schon die anderen Messungen

ergeben haben, spielt vor allem die Länge der Strecke eine Rolle, auch der 2 Mbit/s-PVC war über Hamburg geschaltet.
Die Delays der T3Es machen auf der Verbindung zwischen Berlin und Rostock je nach Messung zwischen 26 % und 50 % des Gesamt-Delays aus. Das ergeben die gemessenen Werte der Loopbacktests aus Tabelle 15 und 16 (berte → berte, cros → cros).
Das WiN-Labor in Erlangen [WIN] führt permanent ATM-Delay-Messungen zwischen verschiedenen B-WiN-Standorten durch. Eine Auswertung von Delaymessungen zwischen Berlin und Hamburg zeigt, daß die vom Netzbetreiber geschalteten Verbindungen große Schwankungen in der Güte aufweisen. Das ATM-Delay zwischen Berlin und Hamburg liegt z.B. zwischen dem 8. und 27.April 1999 bei ca. 2,2 ms - ca. 4,5 ms. Folglich hat die Konfiguration des PVCs durch den Netzbetreiber einen erheblichen Einfluß auf die Höhe der Latenzzeit. In Tabelle 16 sind die Latenzzeiten angegeben, die auf dem leeren PVC gemessen wurden und wie erwartet niedriger ausfallen als bei dem durch die Anwendung belasteten PVC. Der Unterschied beträgt ca. 4 ms.

Tabelle 13: cros → berte (2 Mbit/s-PVC)

Socketbuffersize	**Messagesize** 64 KByte	32 KByte	16 KByte	
64 KByte	1,31	1,31	1,31	Maximum
	$1,28 \pm 0,04$	$1,28 \pm 0,04$	$1,27 \pm 0,04$	Mittelwert
	1,24	1,25	1,15	Minimum
32 KByte	1,30	1,30	1,30	Maximum
	$1,29 \pm 0,04$	$1,29 \pm 0,02$	$1,28 \pm 0,04$	Mittelwert
	1,29	1,28	1,27	Minimum

Alle Angaben in Mbit/s

Tabelle 14: berte → cros (2 Mbit/s-PVC)

Socketbuffersize	**Messagesize** 64 KByte	32 KByte	16 KByte	
64 KByte	1,31	1,31	1,31	Maximum
	$1,30 \pm 0,04$	$1,30 \pm 0,04$	$1,30 \pm 0,04$	Mittelwert
	1,21	1,29	1,29	Minimum
32 KByte	1,30	1,29	1,29	Maximum
	$1,29 \pm 0,04$	$1,28 \pm 0,02$	$1,27 \pm 0,04$	Mittelwert
	1,28	1,22	1,24	Minimum

Alle Angaben in Mbit/s

Tabelle 15: Latenzzeiten (2 Mbit/s-PVC)

	berte	cros	
cros	21,38	3,46	Minimum
	$26,10 \pm 1,33$	$5,36 \pm 0,36$	Mittelwert
	392,16	9,63	Maximum
berte	4,51	21,65	Minimum
	$6,93 \pm 0,55$	$25,69 \pm 0,99$	Mittelwert
	17,49	37,26	Maximum

Alle Angaben in ms

Tabelle 16: Latenzzeiten auf leerem 2 Mbit/s-PVC

	berte	cros	
cros	21,00	2,34	Minimum
	$21,33 \pm 4,43$	$2,46 \pm 0,52$	Mittelwert
	22,33	2,90	Maximum
berte	2,58	20,99	Minimum
	$3,11 \pm 0,73$	$21,53 \pm 4,49$	Mittelwert
	6,25	23,99	Maximum

Alle Angaben in ms

4 Die Anwendung

Matthias Linke

4.1 Beschreibung der Anwendung

Das Simulationsmodell LEGOMODEL ist eine Entwicklung von Michael Kleyer aus dem Fachbereich Landeskultur und Umweltschutz der Universität Rostock. Das Modell soll Pflanze und Umwelt so einfach wie möglich formalisieren. Ausgangspunkt ist ein leerer Raum, den unterschiedliche pflanzliche Wuchs- und Reproduktionstypen mit gleichen Anfangschancen besiedelt haben. Nach Ablauf der Simulation soll geprüft werden, welche Typen eine gegebene Kombination aus Eingriffen und Ressourcen überdauern. Ziel der Simulation ist die Ermittlung von spezifischen Überdauerungsbereichen von Pflanzen ähnlicher Biologie in einem Feld aus Eingriffs- und Ressourcengradienten. Diese Bereiche können als Grundlage für die Beurteilung der Frage dienen, ob Pflanzen unter einem bestimmten Standortregime überleben können, das im Rahmen einer Landschaftsplanung zu erwarten ist.
Modelltheoretisch kann das Programm als die Verknüpfung eines dreidimensionalen zellulären Automaten mit einem „i-state configuration model“ angesehen werden. Die Tabelle, in der in jedem Zeitschritt jeder Zustand jedes Individuums bilanziert wird, stellt die „i-state configuration“ dar. Im zellulären Automaten werden, regelbasiert und unter Berücksichtigung der lokalen Verhältnisse um jede Zelle, die Modellprozesse räumlich hochaufgelöst ausgehandelt.
Pflanzen sind modular aufgebaut. Jedes Blatt bildet mit dem zugehörigen Stengelglied ein Modul. Diese Module werden in artspezifischen Anordnungen zusammengesetzt und ergeben eine Pflanze. Neben der Möglichkeit, lange oder kurze Stengelglieder, große oder kleine Blätter auszubilden, kann die Pflanze auch die Architektur der Wurzel variieren, die Samenbildung ändern oder als annuelle bzw. perenne klonale Pflanze auftreten. Der modulare Aufbau der Pflanzen wird als Wuchs- und Reproduktionstyp realisiert, der nach Regeln angelegt wird.
Das Modell simuliert Pflanzen als diskrete räumliche Individuen mit einzelnen Blättern und Wurzeln. Die Daten eines jeden Individuums werden in einer Tabelle verwaltet.

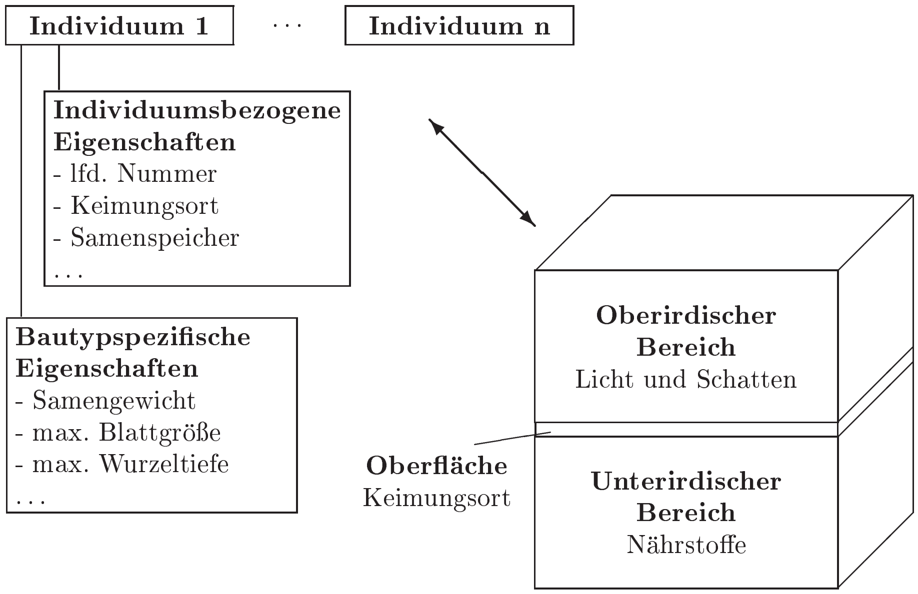

Abbildung 9: Individuentabelle und Simulationsraum

Jedes Individuum beginnt als Same. Die Keimorte der Samen auf der simulierten Bodenoberfläche werden durch Ziehung von Zufallszahlen bestimmt („lottery for living space“). Der Same kann je nach Umwelt- und Konkurrenzbedingungen keimen und vegetativ aufwachsen. Das Wachstum wird durch verschiedene Parameter bestimmt. Zum einen definieren bautypspezifische Parameter die generellen Wachstumsmöglichkeiten. Diese Werte sind für alle Individuen dieses Bautyps identisch. Auf der anderen Seite bestimmen zeitabhängige individuenspezifische Parameter die Entwicklung der Pflanze. Diese zeitabhängigen Werte ergeben sich direkt aus den Konkurrenzbedingungen im Simulationsraum.
Der Simulationsraum wird durch ein dreidimensionales Feld aufgespannt. Dieser Kubus stellt den oberirdischen und unterirdischen Raum dar, in dem die Pflanzen wachsen und interagieren. Die Zellen des Kubus nehmen einen der drei Zustände

- Pflanze
- Ressource : oberirdisch = Licht
- keine Ressource : oberirdisch = Schatten

an. Am Anfang einer Simulation sind alle Zellen des Kubus auf „Ressourcen“ gesetzt. Die Randzellen werden mit „keine Ressourcen“ initialisiert. Damit sind die Randbedingungen für die Simulation vorgegeben.
Ein Zeitschritt der Simulation sei als Tag bezeichnet. 30 Tage ergeben einen Monat, 7 Monate eine Vegetationsperiode. Die Simulationszeit beträgt 100 Vegetationsperioden. Im ersten Zeitschritt werden die Keimorte der Samen auf der Bodenoberfläche ausgelost. In jedem weiteren Zeitschritt erfolgt der Bau der Pflanze. Dabei werden in einer definierten Reihenfolge der Stengel, die Blätter und die Wurzel gebaut. Die Regeln zum Aufbau der einzelnen Pflanzenteile sind bautypspezifisch. Der Aufbau der Pflanzenteile erfordert eine Interaktion mit dem Simulationsraum. Nur wenn Ressourcen in einer Zelle vorhanden sind, kann eine Pflanze diese Zelle belegen. Das Konkurrenzverhalten ist daher stark lokalisiert. Die Anzahl belegter Zellen spiegelt sich in verschiedenen individuenspezifischen Parametern wieder. Diese Parameter wiederum bestimmen die Überlebensfähigkeit eines Individuums. Ein Parameter definiert die Fähigkeit, Samen zu erzeugen und zu verstreuen bzw. Ausläufer anzulegen.
Das Prinzip der „lottery for living space“ bringt eine stochastische Komponente in die Simulation. Daraus folgt eine Wiederholung der Simulation, um statistisch auswertbare Ergebnisse zu erhalten.
Detailliertere Aussagen zum LEGOMODEL können in [Kleyer1] und [Kleyer2] nachgelesen werden.

4.2 Parallelisierungsebenen

4.2.1 Vorbemerkungen

Die Anwendung war ursprünglich unter PASCAL auf einer SUN-Workstation implementiert. In dem oben erwähnten Literaturverweis sind Simulationsergebnisse enthalten. Es wurden für einen Zeitraum von 30 Vegetationsperioden 12 Wiederholungen aller Kombinationen aus Eingriffsintensität und Ressourcenangebot durchgeführt. Der Simulationsraum besteht aus $200 \times 250 \times 250$ Zellen. Es werden 16 Wuchstypen gebildet, die eine Analogie zu den im Gelände gefundenen Gruppen ähnlicher Biologie aufweisen. Jeder Simulationslauf wird mit 50 Samen der 16 Wuchstypen initialisiert. Die abiotischen Umweltbedingungen sind räumlich absolut homogen. Das ursprüngliche LEGOMODEL simuliert einen Pflanzenbestand als völlig isolierten Simulationsraum (*patch*), ohne Austauschbeziehungen nach außen.
Der Schritt zur Parallelisierung ergab sich aus der unbefriedigenden Anzahl Wiederholungen der Simulationsläufe. Statistisch verwertbare Aussagen sind damit nur bedingt machbar. Eine größere Anzahl an Wiederholungen hätte jedoch die Workstation überfordert. Daneben ist die Anzahl simulierter Vegetationsperioden nicht ausreichend.
Als Zielrechner wurde die CRAY T3E des Rechenzentrums der Universität Rostock ausgewählt. Das parallele Programm sollte möglichst portabel sein.

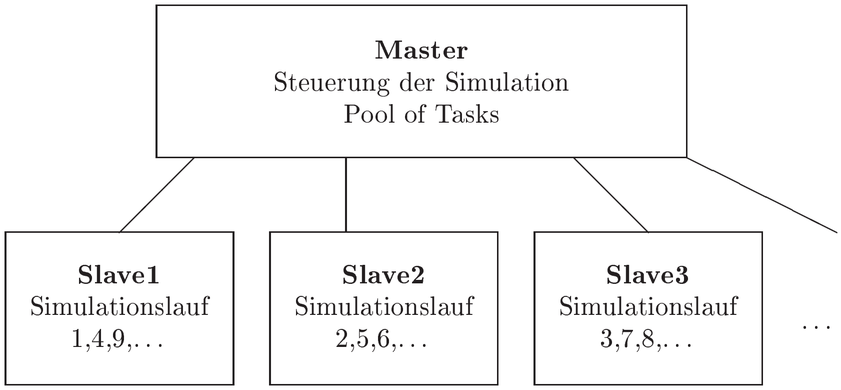

Abbildung 10: 1.Parallelisierungsebene

4.2.2 1. Parallelisierungsebene

Die wesentlichen Ansatzpunkte der Simulation wurden beibehalten. Ziel war es, die einzelnen Simulationsläufe zu verteilen. Da Programm und Daten in den Hauptspeicher eines PE passen, reduzierte sich die Parallelisierung auf das Abarbeiten einer sogenannten „Pool of Tasks“ bzw. „Streichliste“. Aus Portabilitätsgründen wurde eine erste MPI[3]-Version implementiert. Die Realisierung mit MPI hat allerdings einige programmtechnische und damit auch laufzeitrelevante Nachteile. Deshalb wurde ebenso eine shmem[4]-Implementierung realisiert. Um eine portable Version mit dem Verhalten einer shmem-Realisierung zu erhalten, wurde die Streichliste auch in PVM[5] realisiert. Nähere Informationen zu Realisierungen einer Streichliste sind unter [Linke1] zu finden. Ein Vergleich der verschiedenen Realisierungen ist in [Linke2] enthalten.
In diese Realisierungsphase fiel der Beginn der aktiven Mitarbeit am Metacomputing-Projekt zwischen dem Rechenzentrum der Universität Rostock und dem ZIB in Berlin. Die PVM-Realisierung der Anwendung ist ideal geeignet zur Verteilung auf die PE beider CRAY T3E-Systeme. Ein einzelner Master-Prozeß wird auf einem Command-PE gestartet. Dieser Prozeß legt die geforderte Anzahl Slave-Prozesse an und versorgt sie mit Arbeit (Abb.10). Da er keine eigentliche Arbeit ausführt, verbraucht der Master auch kaum Ressourcen auf dem Command-PE. Die Slave-Prozesse können nun sowohl auf dem lokalen System des Masters, als auch auf dem remote-System laufen. Besondere Anforderungen an das Netz werden nicht gestellt, da während der Simulationsläufe keine Kommunikation stattfindet. Nur wenn ein Slave-Prozeß das Ende eines Simulationslaufes meldet, werden Ergebnisdaten vom Slave an den Master gesendet. Der Master startet dann den nächsten Simulationslauf.

Im Rahmen des Projektes wurde eine Verteilung auf beide T3E-Systeme realisiert. Programmtechnisch stellt das kein Problem dar. Dabei ist es egal, auf welcher Maschine der Master-Prozeß läuft. Es müssen lediglich folgende Punkte sichergestellt werden:

- Die Systeme sind in der virtuellen Maschine enthalten. Die Einbindung eines Systems über die PVM-Konsole funktioniert nicht. Es wird also der PVM-Dämon mit einem entsprechenden Hostfile gestartet.

- Jeder Prozeß muß mit dem Master kommunizieren dürfen. Standardmäßig wird nur Prozeß 0 mit dem Master durch einen Socket verbunden. Daher ist es notwendig, die Umgebungsvariable PVM_PE_LIST auf den Wert **all** zu setzen. Das Setzen der Variable muß vor dem Starten des Dämons erfolgen.

- Das Modul *mpt* muß geladen sein.

[3] Message Passing Interface
[4] logischer shared, distributed memory access
[5] Parallel Virtual Machine

In das Arbeitsverzeichnis von PVM wird eine Datei *pvmd* abgelegt, die erst das Modul lädt, dann die Umgebungsvariable setzt und anschließend den PVM-Dämonen startet.

```
#!/bin/sh
. /opt/modules/modules/init/sh
module load mpt
export PVM_PE_LIST=all
export PVM_AUTO_ERR=0
exec pvmd3 $*
```

Das Hostfile für den Start des Masters auf `cros` und `berte` als remote System sieht dann wie folgt aus:

```
berte.zib.de lo=buvrlink dx=/home/buvrlink/pvm3/bin/CRAY/pvmd
```

Es wird also *pvmd* ausgeführt und nicht direkt der Dämon gestartet. Über die PVM-Konsole kann man sich nun die korrekte Konfiguration der virtuellen Maschine ansehen.
Der Master wird jetzt wie gewöhnlich gestartet. Über Parameter wird die Anzahl der zu startenden Prozesse auf den einzelnen Systemen gesteuert.
Bezüglich der Anwendung wurde im Rahmen des Projektes nur die Machbarkeit des Ansatzes getestet. Ein eigentlicher Produktionslauf fand aus zwei Gründen nicht statt. Zum einen lagen die Ergebnisse für diesen Lauf bereits vor. Zum anderen ist ein derartiger Ansatz keine tatsächliche Herausforderung an die Kopplung zweier Parallelrechner, da er keine Kommunikationsanforderungen stellt.

4.2.3 2. Parallelisierungsebene

Eine erste Erweiterung des ursprünglichen Simulationsprogrammes ist die Aufhebung der Einschränkung auf ein isoliertes Patch. Es sollte damit ermöglicht werden, daß Bautypen, die auf dem lokalen Patch kaum eine Überlebenschance haben, durch Sameneintrag von außerhalb gestärkt werden. Dadurch ändert sich die Struktur der Abarbeitung der Simulationsläufe. Im Gegensatz zum ersten Ansatz, in dem jeder Prozeß genau ein Patch bearbeitet, werden jetzt *npatch*[6] Prozesse zur Bearbeitung eines Simulationslaufes benötigt (Abbildung 11). Die Interaktion der Patches findet an genau einer Stelle im Programm statt, der Samenverteilung. Der Samen kann jetzt auch außerhalb des lokalen Patches abgelegt werden. Das führt zu einer zusätzlichen Kommunikationsebene. Neben der Kommunikation zwischen Master und Slaves tauschen nun auch die Slaves, die interagierende Patches bearbeiten, Daten aus. Zusätzlich müssen sich diese Prozesse auf zwei Ebenen synchronisieren.

- Synchronisation während des Simulationslaufes
  Der Austausch der Samen kann zu unterschiedlichen Zeitpunkten der Simulation erfolgen. Zu diesen Zeitpunkten muß der korrekte Eintrag der Samen gewährleistet werden. Das bedeutet, daß sich die Simulation beider Patches im selben Zustand befinden muß.

- Synchronisation der Simulationsläufe
  Die einzelnen Simulationen der interagierenden Patches müssen natürlich zur selben Zeit starten.

Die Realisierung dieses Ansatzes baut auf der oben beschriebenen Implementierung auf. Die Kommunikation zwischen Master und Slaves erfolgt über PVM. Beim Datenaustausch zwischen den PE wird unterschieden, wo die einzelnen Prozesse gestartet wurden. Dabei gibt es zwei Möglichkeiten. Die Prozesse zur Simulation interagierender Patches laufen auf demselben Host. Eine effektive Kommunikation ist dann nur durch den Einsatz der crayspezifischen libsma-Bibliothek möglich. Damit wird zwar der Anspruch der Portabilität aufgegeben, aber die Gesamtabarbeitungszeit reduziert. Laufen die Prozesse zur Simulation interagierender Patches

[6] Anzahl interagierender Patches

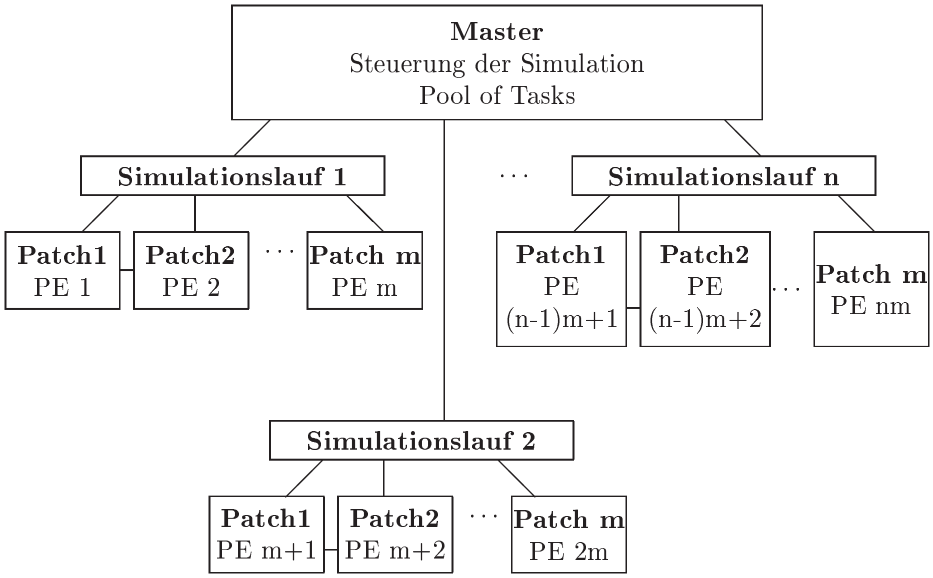

Abbildung 11: 2.Parallelisierungsebene

jedoch auf unterschiedlichen Hosts, muß die Kommunikation zwangsläufig über PVM realisiert werden.
Dieser Ansatz ist bezüglich der Kopplung mehrerer Rechner wesentlich interessanter. Die Anforderungen an das dazwischen liegende Netz steigen. Verschiedene Parameter bestimmen die Anzahl und Größe der zu übertragenden Nachrichten. Eine Kopplung zweier Systeme auf dieser Parallelisierungsebene ist sicherlich für das vorliegende Projekt der einzig mögliche Ansatz. Daher werden die wesentlichen Steuerparamter und Realisierungsmöglichkeiten später näher vorgestellt.

4.2.4 3. Parallelisierungsebene

Es wurden entscheidende Änderungen bezüglich der biologischen Vorgaben vorgenommen. Die wesentlichste ist dabei sicherlich der Übergang von fest vorgegebenen Bautypen zu zufällig gewählten Modulparametern für jedes Individuum. Aus den gegebenen Werten für jeden Parameter wird für jedes zu initialisierende Individuum ein Wert zufällig ausgewählt und in die Tabelle eingetragen. Große oder kleine Blätter werden nach dem Zufallsprinzip mit langen oder kurzen Stengelgliedern, mit langen oder kurzen Ausläufern, mit großen oder kleinen Samen zu abstrakten Pflanzen kombiniert. Diese werden dann in Konkurrenz zueinander in der Simulation selektiven Wettläufen unterzogen. Die Anzahl der Bautypen wird daher die ursprüngliche Anzahl von 16 weit übersteigen. Um statistisch relevante Aussagen machen zu können, müssen eine Mindestanzahl von Individuen jeden Bautyps initialisiert werden. Das führt sicherlich zu der Notwendigkeit eines größeren Simulationsraumes, um nicht von vornherein die Konkurrenz durch Platzmangel zu steuern. Ein größerer Simulationsraum wird nicht mehr in den Speicher eines PE passen, so daß eine Aufteilung des Raumes auf mehrere PE notwendig ist. Das führt zu einer weiteren Kommunikationsebene in der Anwendung.
Die resultierende Anwendung beinhaltet drei Kommunikationsebenen. Die erste Kommunikationsebene entspricht dem ersten Parallelisierungsansatz. Die zweite Ebene resultiert aus dem zweiten Ansatz. Die oben gemachten Aussagen bleiben vollständig gültig. Die Kommunikation in der dritten Ebene ist die anspruchvollste bezüglich einer effektiven Realisierung. Durch die Aufteilung des Simulationsraumes muß beim Wachstum der Blätter und Wurzeln der Übergang von einem Teilraum in einen anderen beachtet werden. Dieser Übergang bedeutet Kommunikation. Die auszutauschenden Daten bestehen in der derzeitigen Realisierung aus einem INTEGER-Wert. Der zu übertragende Wert ist entweder der aktuelle Zustand einer Zelle des Si-

mulationsraumes oder deren neuer Zustand. Es ist mit einer häufigen Kommunikationstätigkeit, und damit einer hohen Netzlast zu rechnen. Eine effektive Umsetzung dieser Kommunikationsebene ist daher nur mit der libsma-Bibliothek zu gewährleisten. Der Overhead eines Message Passing-Systems wie MPI oder PVM verhindert eine performante Realisierung. Daraus folgt, daß die Aufteilung eines Patches über die gekoppelten Systeme nicht effizient ist.
Bezüglich der Verteilung der Simulationsläufe auf verschiedene Rechner ändert sich gegenüber dem zweiten Parallelisierungsansatz nichts. Es werden nur vollständige Patches auf jedem Host abgelegt. Der Grund liegt natürlich in der dritten Kommunikationsebene. Die Synchronisation der verteilten Prozesse kann sich aber entscheidend erschweren, wenn die Anzahl der Teilräume der interagierenden Patches auf den verschiedenen Rechnern unterschiedlich ist.

4.3 Metacomputing in der zweiten Parallelisierungsebene

4.3.1 Steuerparameter

Die zweite Parallelisierungsebene stellt bezüglich einer Kopplung die interessanteste Variante dar. Eine Kopplung in der ersten Ebene ist trivial, der Kommunikationsaufwand der dritten Ebene zu groß. Die Kopplung der Prozesse erfolgt daher über den Samenaustausch. Individuen erzeugen zu gegebenen Zeitpunkten in Abhängigkeit verschiedener individuenspezifischer Parameter Samen. Diese Samen werden auf die vorhandenen Patches verteilt. Wird nur ein Patch betrachtet, bleiben alle Samen im lokalen Patch. Die Verteilung auf mehrere interagierende Patches wird durch zwei Parameter gesteuert. Für alle Samen wird der Anteil angegeben, der lokal im Patch verbleibt. Ein weiterer Parameter gibt für jeden Samentyp an, wieviel Patches er „wandern“ kann. Es wird dabei unterschieden zwischen *leichten* und *schweren* Samen.
Die Verteilung der Samen erfolgt in zwei Schleifen. In der ersten Schleife wird der lokale Anteil der Samen im Simulationsraum abgelegt. Es wird eine zufällige Position auf der Oberfläche gezogen. Eine Kommunikation findet nur dann statt, wenn das Patch selber aufgeteilt ist. In der zweiten Schleife wird der Samen auf andere Patches verteilt. Ein Patch wird zufällig ausgewählt. Dieses Patch muß in der Reichweite des jeweiligen Samentyp liegen.
Die zu übertragenden Daten können zwischen vielen kurzen Nachrichten bis zu wenigen langen Nachrichten schwanken. Wesentlicher Aspekt ist dabei der Zeitpunkt der Samenübertragung.

1. Senden jedes einzelnen Samens
   Sobald das Zielpatch des Samens ermittelt ist, wird der Samen an das entsprechende PE gesendet. Die zu sendende Nachricht besteht dann genau aus den Daten eines Individuums. Die Anzahl solcher kleinen Nachrichten hängt von der Anzahl Individuen, die zu einem bestimmten Simulationszeitpunkt Samen erzeugen können, und deren Anzahl erzeugter Samen ab. Mit fortlaufender Simulation werden immer mehr Individuen diesen Zustand erreichen, sodaß die Anforderungen an die Kommunikation steigen. Das Senden der Samen stellt dabei das kleinere Problem dar, da es asynchron zur Simulation erfolgen kann. Der Empfänger der Samen muß seine Simulation tatsächlich unterbrechen, um die Samen sowohl im Simulationsraum, als auch in der Individuentabelle abzulegen.

2. Senden der Samen eines Individuums
   Die Zielpatches der Samen eines Individuums werden gesammelt. Die Anzahl der zu sendenden Nachrichten reduziert sich für einen Sendevorgang maximal auf die Anzahl interagierender Patches. Es werden also wesentlich weniger Nachrichten versendet.

Ein zweiter wesentlicher Aspekt ist der Synchronisationszeitpunkt während der Simulation. Die Prozesse interagierender Patches müssen zu dedizierten Zeitpunkten den gleichen Status der Simulation gewährleisten. Ein solcher Zeitpunkt ist die Ausgabe des Simulationsraumes. Ein zweiter Zeitpunkt muß so gewählt werden, daß ein korrekter Sameneintrag gewährleistet ist. Der sicherste Weg ist die **tageweise** Synchronisation. Die Prozesse synchronisieren sich jeweils nach der Simulation eines Zeitschrittes. Damit sind alle Prozesse im selben Status der Simulation. Der resultierende hohe Kommmunikationsaufwand ist für die verteilte Simulation jedoch von entscheidendem Nachteil. Außerdem müssen die Prozesse nur zum Zeitpunkt des Samenaustausches im selben Zustand sein. Ein Samenaustausch findet jedoch nur zu dedizierten Zeitpunkten

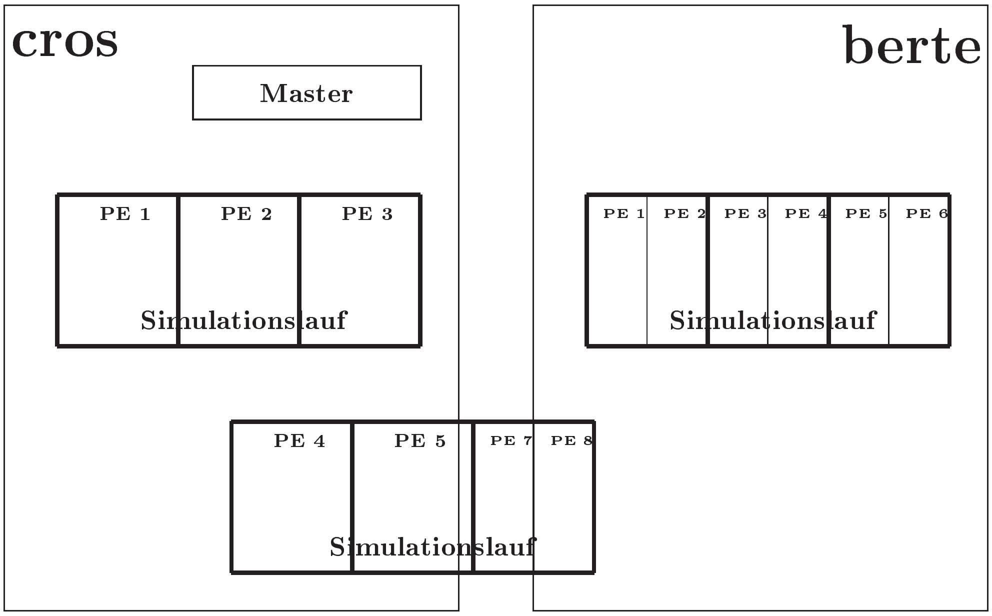

Abbildung 12: Prozeßverteilung bei gekoppelter Simulation

statt, entweder wenn *monat % 3 = 0* oder *monat % 5 = 0* gilt. Der Synchronisations- und damit Kommunikationsaufwand reduziert sich erheblich.

4.3.2 Verteilung der Prozesse

Die interagierenden Patches werden unterschieden in *lokale* und *remote* Patches. Die Unterscheidung erfolgt bezüglich jeden Prozesses. Alle Prozesse, die interagierende Patches bearbeiten, werden in eine logische Partition eingeordnet. Innerhalb einer logischen Partition ist die Aufteilung in lokale bzw. remote Patches möglich. Abbildung 12 veranschaulicht dies. Es laufen gleichzeitig drei Simulationen. Jeder Simulationslauf bearbeitet drei interagierende Patches. Jeweils eine Simulation läuft ausschließlich auf einem Host. Ein Simulationslauf ist verteilt auf beide Hosts. Die für eine Kopplung interessante Kommunikation spielt sich zwischen den PE 4 und 5 auf `cros` und den PE 7 und 8 auf `berte` ab. Für die Prozesse 4 und 5 sind die Prozesse 7 und 8 remote und umgekehrt.
Es wurde eine statische Verteilung der Prozesse implementiert. Die Prozesse werden zu logischen Gruppen zusammengefasst. In einer Gruppe liegen alle die Prozesse, die denselben Simulationsraum und interagierende Patches bearbeiten. In Abbildung 12 bestehen die einzelnen Gruppen demnach aus den PE 1,2 und 3 auf `cros`, den PE 1-6 auf `berte` sowie den PE 4 und 5 auf `cros` und PE 7 und 8 auf `berte`. Die Anzahl der PE pro Gruppe wird zum einen durch die Vorgaben der Größe des Simulationsraumes und der Anzahl interagierender Patches, zum anderen durch die Größe des Hauptspeichers der PE bestimmt.

4.3.3 Realisierung der Kommunikation

Die Kommunikation wird entsprechend der Unterscheidung in lokale und remote Patches realisiert. Eine lokale Kommunikation findet immer über die shmem-Bibliotheken statt. Der Sendeprozeß zieht zufällig eine Position auf der Oberfläche des Zielpatches. Dann holt er sich per *shmem_get* alle relevanten Informationen zur Ablage des Samens. Ist eine Samung möglich, werden die entsprechenden Änderungen im Simulationsraum und der Individuenliste durch *shmem_put*-Aufrufe vorgenommen. Damit es nicht zu Kollisionen mit anderen Prozessen kommt, wird dieser Programmabschnitt durch *shmem_set_lock* bzw. *shmem_clear_lock* geschützt. Die Verantwortung der Samenablage liegt demnach nur auf Seiten des Samenerzeuger-Prozesses.
Die Verteilung von Samen an remote Patches erfolgt mit PVM-Kommunikationsroutinen. Alle

relevanten Informationen werden in einen Sendepuffer gepackt und an den Zielprozeß gesendet. Damit ist die Arbeit des Samenerzeuger-Prozesses erledigt. Der Zielprozeß muß dafür sorgen, daß an ihn gerichtete Nachrichten vom PVM abgeholt werden. Die entsprechenden Informationen werden aus dem Empfangspuffer ausgepackt und die Samung angestoßen. Die Verantwortung der Samenablage liegt hier also ganz auf Seiten des Empfängerprozesses.
Die Synchronisation der Prozesse findet in zwei Schritten statt. Zuerst synchronisieren sich die lokalen Prozesse durch *shmem_barrier*. Danach müssen sich die remote Prozesse über PVM synchronisieren. Da die PVM-Implementation auf der CRAY keine dynamischen Gruppen über verschiedenen Partitionen zuläßt [CrayPVM], muß die Synchronisation der remote Prozesse über eine explizite Kommunikation erfolgen. Realisiert ist diese Kommunikation durch einen *pvm_mcast*-Aufruf mit dazugehörigen Empfangsroutinen.

4.4 Ergebnisse

4.4.1 Voraussetzungen und Probleme

Der Ablauf der Simulation kann durch eine Vielzahl von Parametern gesteuert werden. Auf der einen Seite stehen die Parameter zur Definition der zu simulierenden Bautypen. Für die vorliegenden Ergebnisse wurden die Parameter so gesetzt, daß pro Patch jeweils ein Individuum eines Bautyps initialisiert wurde. Der Bautyp wurde so gewählt, daß die Individuen nur Samen aber keine Ausläufer bilden. Durch diese Wahl wird die Abarbeitung auf die Samenbildung und -verteilung konzentriert. Gleichzeitig kann die Ausbreitung jedes Bautyps vom Ausgangspatch ausgehend und damit die Kommunikation zwischen den Prozessen beobachtet werden. Die Größe eines Patches ist beschränkt worden. Es wurde daher kein eigentlicher Produktionslauf der Simulation durchgeführt.
Alle Samen können maximal ein Patch wandern. Von den erzeugten Samen bleiben 90% im eigenen Patch. Die restlichen Samen werden auf die lokalen und remote Patches verteilt.
Die Synchronisation erfolgt immer monatsweise. Die Samen werden sowohl einzeln als auch individuumsweise verteilt.
Die Simulationsläufe wurden für verschiedene Anzahl Patches durchgeführt. Je nach Anzahl wurden verschiedene Aufteilungen der Patches realisiert. Desweiteren wurde die Aufteilung der Prozesse auf die beiden Cray-Systeme variiert.
Ein großes Problem stellt die lange Laufzeit einer Simulation dar. Obwohl die Bautyp-Parameter so gewählt wurden, daß möglichst wenig Laufzeit im Wachstum der Pflanzen oder Routinen zur Behandlung von Ausläufern verbraucht wird, liegt die Laufzeit bei 16 Prozessen zwischen zwei und drei Stunden. LEGOMODEL beinhaltet jedoch einige stochastische Prozesse, wie z.B. die Samenverteilung. Daher ist eine Wiederholung desselben Simulationslaufes für gesicherte statistische Aussagen notwendig. Die oben genannte Zahl von 100 Wiederholungen ist für den angestrebten Machbarkeitstest aufgrund der langen Laufzeiten nicht durchführbar. Insbesondere stellte der vorgegebene Zeitrahmen des PVC ein Problem dar. Um die Tests zu einem Ende in vernünftigem Zeitrahmen zu bringen, beschränkten sich die Läufe über das B-Win im Wesentlichen auf die über den PVC getesteten Parameterkombinationen.
Die verteilten Simulationsläufe wurden im interaktiven Modus ausgeführt. Eine Batch-Verarbeitung ist nur unter großem administrativem Aufwand möglich, da die angeforderten Prozessoren zum gleichen Zeitpunkt auf beiden Systemen frei sein müssen. Sonst würden die Laufzeiten schon durch das Warten eines Teils der Anwendung verfälscht werden.
Ein nicht zu unterschätzendes Problem ist die administrative Seite, die sich aus der rein interaktiven Arbeit ergibt. Die Limits für interaktive Prozesse sind meist wesentlich geringer als die der Batch-Jobs. Im vorliegenden Fall mußten diese Limits auf beiden Systemen hochgesetzt werden. Ein weiterer Aspekt ist die Begrenzung der PE, die für interaktive Jobs vorhanden ist. In der Regel stehen wesentlich weniger Prozessoren zur Verfügung als für Batch-Jobs. Diese Prozessoren sind dann auch eher zum Testen als für Produktionsläufe gedacht. Für dieses Testbeispiel wurden auf der Cray T3E in Rostock extra 8 PE reserviert, um die PVC-Läufe ungestört durchführen zu können. Die Läufe über das B-Win fanden jedoch unter normalen Arbeitsbedingungen auf beiden Systemen statt. Das bedeutet z.B., daß auf dem Berliner System eine interaktive Arbeit normalerweise nur von 8 - 18 Uhr möglich ist. Um dennoch die vorliegen-

den Test schnell durchführen zu können, wurden speziell für die Wochenenden Reservierungen von PE vorgenommen.

4.4.2 Lokale Simulation

Bevor die Verteilung der Simulation auf verschiedene Rechner getestet werden konnte, mußte sichergestellt sein, daß die verschiedenen Parameter für die Simulation zum gleichen biologischen Ergebnis führen. Das heißt, am Ende eines Simulationslaufes sollten die einzelnen Bautypen im Rahmen der Stochastik in den gleichen Patches vorkommen. Wesentliche Aspekte waren dabei die Aufteilung der Patches sowie der Synchronisationspunkt. Die Samenverteilung erfolgt immer über direkten Speicherzugriff per *shmem*-Routinen und für jeden Samen einzeln.
In der folgenden Tabelle sind die Laufzeiten (s) für die einzelnen Parameter dargestellt. Zusätzlich werden die Differenzen der Laufzeiten (s) angegeben.

Anzahl	Aufteilung	Synchronisation			
Patches		tageweise		monatsweise	
4	4 x 1	920	37	752	269
	2 x 2	914	42	675	69
8	8 x 1	1209	67	952	74
	4 x 2	1201	78	962	49
16	16 x 1	1818	220	1917	422
	8 x 2	1748	292	1536	206
	4 x 4	1741	177	1911	500

Die Laufzeiten bei einer tageweisen Synchronisation unterliegen im wesentlichen keinen großen Schwankungen. Mit zunehmender Anzahl Patches und damit Prozessoren steigt natürlich die Differenz zwischen der minimalen und der maximalen Abarbeitungszeit. Die Laufzeitschwankungen bei einer monatlichen Synchronisation hängen von der Arbeit der einzelnen Prozessoren im letzten Monat ab. Hier kann es offensichtlich zu beträchtlichen Unterschieden kommen. Die Aufteilung der Patches scheint dabei eine Rolle zu spielen.
Für die verteilte Simulation wurden die Zeitmessungen mit der monatlichen Synchronisation durchgeführt.

4.4.3 Kommunikation über B-Win

Im Gegensatz zur rein lokalen Abarbeitung spielt der Zeitpunkt des Sendens des Samens eine Rolle. Es wurden zwei grundsätzliche Varianten getestet.

- **klmv** - Samen werden einzeln gesendet.

- **kl41mv** - Samen werden pro Individuum gesendet.

Das Problem bei einer Kommunikation über das B-Win ist die ständig schwankende Netzlast. Die Simulationsläufe müßten daher zu verschiedenen Tageszeiten bzw. Wochentagen wiederholt werden, um eventuelle Abhängigkeiten beobachten zu können. Die Tabelle stellt die mittleren Laufzeiten (s) und die Differenzen zwischen der minimalen und der maximalen Laufzeit (s) für die verschiedenen Parameter dar. Die einzelnen Simulationsläufe fanden zu unterschiedlichen Tageszeiten und an verschiedenen Wochentagen statt.

	Anzahl Patches	Aufteilung Patches	Prozesse	klmv		kl41mv	
1	4	4 x 1	2-2	3760	678	3758	454
2		2 x 2	2-2	4091	387	4290	1733
3	8	8 x 1	4-4	5263	2786	4312	830
4			2-6	6311	2497		
5		4 x 2	4-4	5224	393	4895	1009
6		16 x 1	8-8	6476	1403	5705	571
7			5-11	6661	1697	8747	6003
8	16	8 x 2	8-8	6665	1826	6292	2154
9			5-11	8084	4006	8098	3261
10		4 x 4	8-8	7439	1404	6790	1722

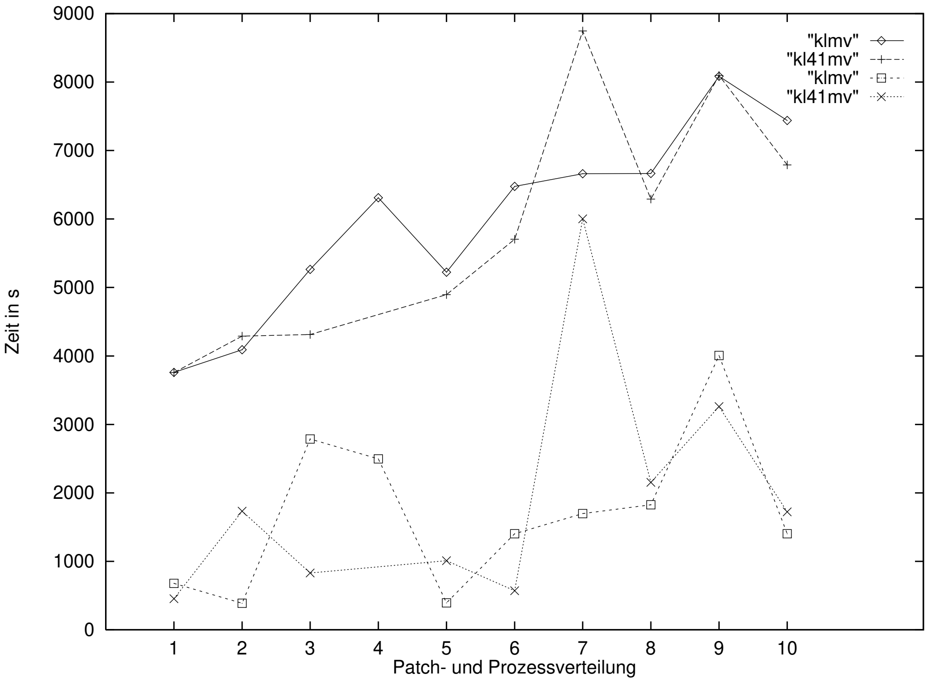

Abbildung 13: Laufzeiten und Differenzen über B-Win

Die dargestellten Laufzeiten zeigen, daß ein Senden der Samen pro Individuum in den meisten Fällen bessere Laufzeiten bringt. Der Kommunikationsoverhead wird reduziert. Abbildung 13 zeigt jedoch auch deutlich den Einfluß der Dimension des Patchverbundes bzw. der Verteilung der Prozesse. Offensichtlich laufen vektorartige Patchanordnungen am schnellsten. Es kommunizieren dann nur zwei Prozesse über das Netz miteinander. Die gesamte andere Kommunikation findet lokal statt. Daneben scheint eine Gleichverteilung der Prozesse auf beide Systeme zu besseren Laufzeiten zu führen. Der Grund hierfür dürften die damit verbundenen identischen Kommunikationsanforderungen zur Synchronisation auf beiden Systemen sein.
Die Laufzeitdifferenzen unterliegen offenbar denselben Gesetzmäßigkeiten.

4.4.4 Kommunikation über PVC

Der PVC stand für die Anwendung eine Woche zur Verfügung. Es wurde versucht, möglichst viele verschiedene Parameter, insbesondere bzgl. der Verteilung der Patches und Prozesse zu untersuchen. Die folgende Tabelle stellt für alle erfolgten Simulationsläufe die untersuchten Parameter, die Anzahl Simulationsläufe(#) und die erzielten Laufzeiten(s) sowie die Laufzeitdifferenzen(s) dar. Die Laufzeiten sind die Durchschnittswerte über die Simulationsläufe.

	Anzahl	Aufteilung		klmv			kl41mv		
	Patches	Patches	Prozesse	#			#		
1	4	4 x 1	2-2	9	3324	730	9	2996	686
2		2 x 2	2-2						
3	8	8 x 1	4-4	5	4611	558	5	4673	314
4		4 x 2	4-4						
5			8-8	4	7678	3168	4	6798	1038
6		16 x 1	5-11	2	7209	984			
7	16		2-14	3	7710	505	2	7198	216
8		4 x 4	8-8	4	7442	1465	4	6674	856

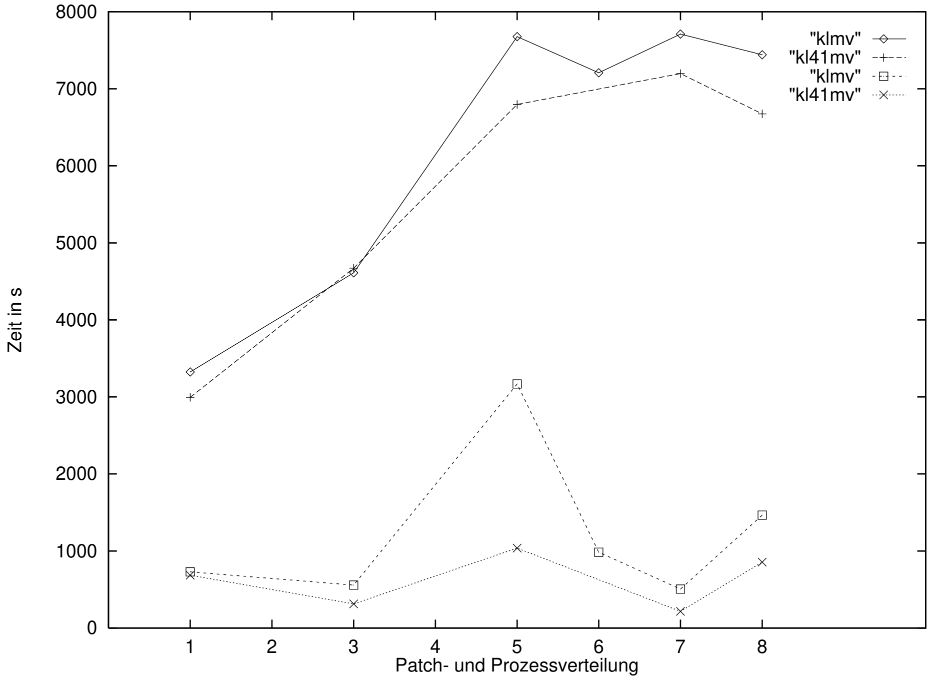

Abbildung 14: Laufzeiten und Differenzen über PVC

Es ist zu sehen, daß die Anzahl der gelaufenen Simulationen sehr gering ist. Statistische Aussagen sind daher nur mit äußerster Vorsicht möglich. Neben der erwähnten langen Laufzeit einer Simulation und der notwendigen Wiederholungsrate begrenzten aufgetretene PVM-Fehler und Hardware-Probleme beim Berliner System die zur Verfügung stehende Nutzungszeit und damit die Anzahl der durchgeführten Simulationsläufe. Insbesondere bei Läufen, die über Nacht bzw. am Wochenende ausgeführt wurden, fallen die aufgetretenen PVM-Fehler stark ins Gewicht, da keine direkte Kontrolle über die Ausführung stattfand. Zusätzlich lag an einigen Tagen eine hohe Beanspruchung des lokalen Zuganges beim Berliner System vor.
Die Kurvenverläufe in Abbildung 14 spiegeln die wesentlichen Aussagen der Abarbeitung übers B-Win wieder. Die Kurven weisen jedoch weniger Schwankungen bezüglich der Dimension und Verteilung der Patches auf. Ausschlaggebend dafür dürfte die stets identische Verbindung zwischen den Systemen sein.

4.4.5 Vergleich der Laufzeiten

Aufgrund der genannten Probleme bei der Nutzung des PVCs sind Vergleiche zwischen der Ausführung der verteilten Simulation über das B-Win bzw. den PVC eigentlich nicht realistisch. Eine nähere Betrachtung der Abbildung 13 bzw. Abbildung 14 zeigt, daß nicht generell gesagt werden kann, daß die Abarbeitung über den PVC in jedem Fall schneller ist. Es kann jedoch festgestellt werden, daß die konkrete Verteilung der Prozesse auf beide Systeme eine

weniger große Rolle spielt als beim B-Win.
Eine weitaus wichtigere Erkenntnis liefert Abbildung 15. Die verteilte Abarbeitung ist drei- bis sechsmal langsamer als eine lokale Simulation. Der Faktor nimmt mit steigender Anzahl Prozesse ab. Liegt er für vier und acht Prozesse noch im Bereich 4 - 6, sinkt er für 16 Prozesse in den Bereich 3 - 4. Die besten Laufzeiten werden offenbar erreicht, wenn eine vektorartige Patchanordnung (4x1,8x1,16x1) so verteilt wird, daß auf beiden Systemen die gleiche Anzahl Prozesse läuft. Dann findet die Kommunikation nur zwischen zwei Prozessen statt. Gleichzeitig ist der Kommunikationsaufwand zur Synchronisation der Prozesse identisch.
Neben den mittleren Laufzeiten sind die Schwankungsbereiche der Laufzeiten interessant. Die

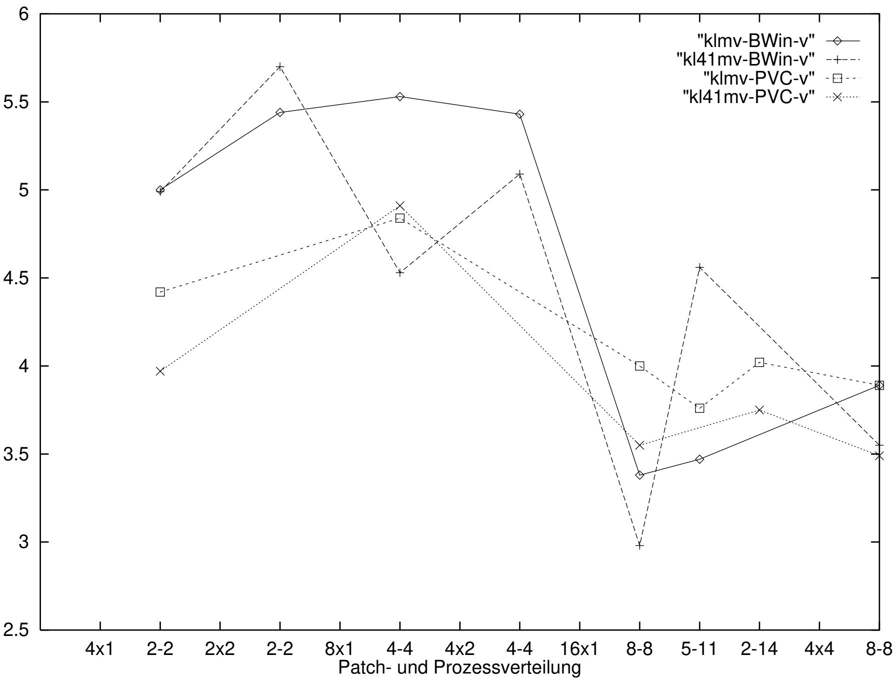

Abbildung 15: Vergleich der Laufzeiten relativ zur lokalen Laufzeit

Ausführungszeiten unterliegen beträchtlichen Schwankungen, die im Wesentlichen vom Algorithmus und der Anzahl der simulierten Individuen abhängen. In den Abbildungen 16 und 17 sind die Differenzen zwischen den gemessenen Minima und Maxima der Laufzeiten aufgezeigt.
Auffällig an den Abbildungen sind die Spitzen, die genau an den Stellen auftreten, für die die besten Laufzeiten diagnostiziert wurden. Das gilt insbesondere für die B-Win-Abarbeitung. Offensichtlich schlägt sich hier die wechselnde Netzlast auf dem B-Win nieder.
Dieser Fakt wird besonders deutlich in der prozentualen Abweichung der Laufzeiten. Mit Ausnahme des Falles des Sendens aller Samen einzeln bewegen sich die Werte für den PVC im Bereich bis 20%. Bei einer Abarbeitung über das B-Win liegen die Abweichungen zum Teil wesentlicher darüber. Daneben unterliegen diese Werte wesentlich größeren dimensions- bzw. verteilungsabhängigen Schwankungen.
Es kann also abschließend festgestellt werden, daß der Einsatz eines PVC nicht zu einer wesentlich schnelleren Abarbeitung der Simulation geführt hat. Das liegt insbesondere an der Spezifik des vorliegenden Algorithmus. Es werden relativ kleine Nachrichten übertragen, demzufolge spielt die Bandbreite nicht die wesentliche Rolle. Offensichtlich ist die aktuell vorliegende Bandbreite des B-Win ausreichend für die getestete Anwendung. Der Vorteil einer Abarbeitung über einen PVC liegt jedoch in einem anderen Aspekt der Anwendung. Aufgrund der Stochastik in LEGOMODEL muß die Simulation mit denselben Parametern wiederholt werden. Die Schwankungen der Laufzeiten sind dann über einen PVC geringer als über das B-Win.

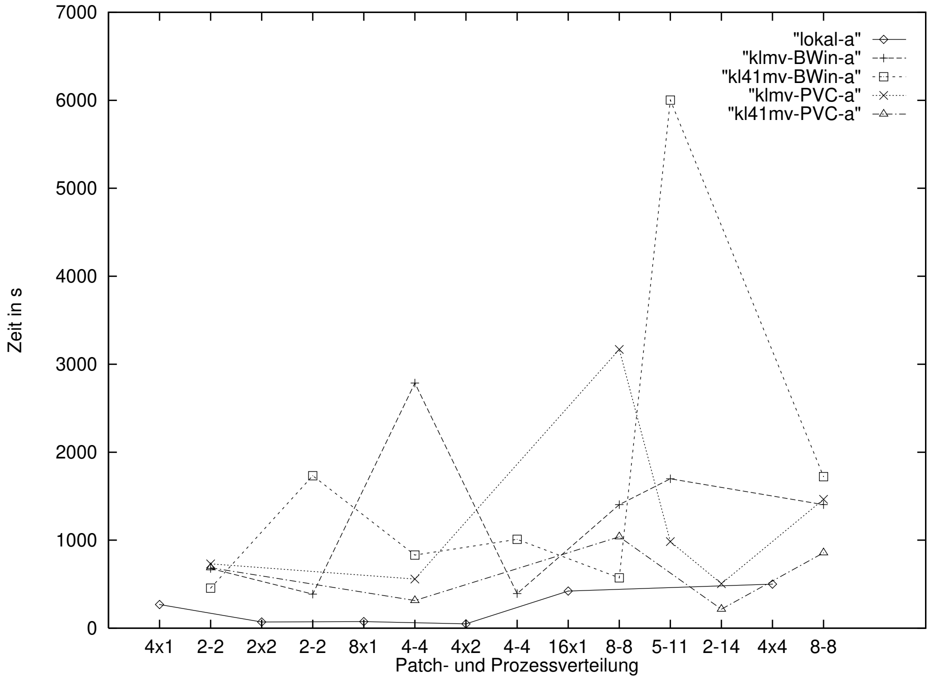

Abbildung 16: Absolute Differenzen der Laufzeiten

4.5 Zusammenfassung und Ausblick

Ziel des Projektes war die Verteilung einer parallelen Anwendung aus dem Bereich der Umweltsimulation auf die CRAY T3E-Systeme in Rostock und Berlin. Die Parallelisierung der Anwendung LEGOMODEL resultiert in drei Parallelisierungsebenen. Für ein Metacomputing auf den Systemen ist die zweite Ebene, die Ebene des Samenaustausches interessant. Die Kommunikation zwischen den simulierten Patches findet durch den Austausch von Samen statt. Dazu müssen die Simulationsprozesse denselben Stand der Simulation erreicht haben. Eine Synchronisation ist daher unumgänglich.
Ein Metacomputing erfordert die Implementierung der Anwendung mit PVM-Kommunikationsroutinen. Die Kommunikation findet daher systemübergreifend über PVM statt. Systemintern erfolgt die Kommunikation über die schnelleren crayspezifischen shmem-Routinen. Die Prozesse werden in logische Partitionen zusammengefaßt. Dadurch wird eine flexiblere Verteilung und Steuerung der gesamten Simulation erreicht.
Verschiedene Parameter steuern die Verteilung der Patches auf die Systeme und bestimmen den Kommunikationsaufwand der Anwendung. Diese Parameter haben mitunter großen Einfluß auf die Laufzeit der verteilten Simulation. Für den vorliegenden Fall haben sich vektorartige Patchdimensionen bei einer Gleichverteilung auf beide Systeme als günstig erwiesen.
Die Kopplung der Cray-Systeme in Berlin und Rostock erfolgte sowohl über das B-Win als auch über einen gemieteten PVC. Ein Vergleich der Laufzeiten zeigt, daß das B-Win durchaus als Grundlage eines Metacomputings dienen kann. Der Vorteil eines PVC liegt in den beobachteten konstanteren Laufzeiten. Für die vorliegende Anwendung spielt das eine große Rolle, da die Simulation mit denselben Parametern öfter wiederholt werden muß, um statistisch korrekte Aussagen zu erhalten.
Es konnte nachgewiesen werden, daß für die betrachtete Anwendung eine Verteilung durchaus sinnvoll sein kann. Ein wesentlicher Hauptgrund wäre zum Beispiel die stark beschränkte Anzahl Prozessoren des Rostocker Systems. Durch die Verteilung auf entfernte Prozessoren können dann auch größere Probleme simuliert werden. Allerdings müssen insbesondere administrative Probleme im Vorfeld gelöst werden.
Die eingeschränkte Nutzungsdauer insbesondere des PVC und die langen Laufzeiten der betrachteten Anwendung lassen noch einige Fragen offen. Wie entwickeln sich die Laufzeiten

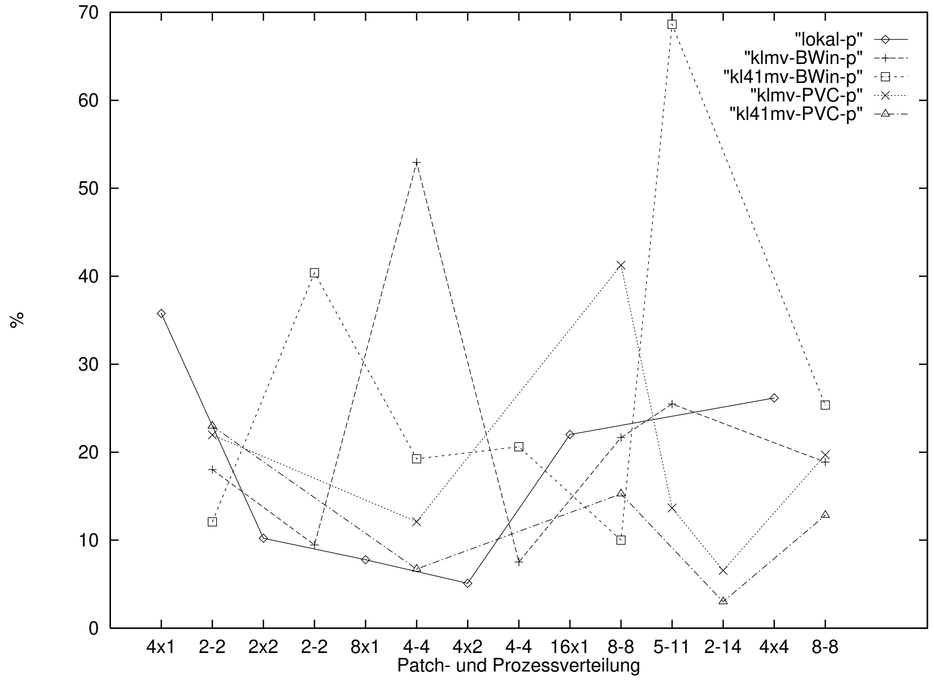

Abbildung 17: Prozentuale Differenzen der Laufzeiten

bei zunehmender Anzahl Prozesse? Welchen Einfluß haben die Aufteilungen der Patches bzw. Prozesse tatsächlich? Hierzu müßten wesentlich mehr Simulationsläufe durchgeführt werden. Die vorliegenden Aussagen können daher als Basis für zukünftige Arbeiten genutzt werden.

A Konfiguration der Rechnersysteme

	berte	cros
Betreiber	Konrad-Zuse-Zentrum für Informationstechnik Berlin	Universitaet Rostock
Hersteller	CRAY Research	CRAY Research
IP-Name	berte.zib.de	cros.rz.uni-rostock.de
Internet Adr.	130.73.206.21	139.30.19.10
Typ	T3E-900 LC 256-128	T3E-900 LC 40-128
Anzahl Proz.	272	44
Taktzeit	2,22 ns	2,22 ns
CPU-Frequenz	450 MHz	450 MHz
Linpack 1000x1000	366 je CPU	366 je CPU
Linpack 100x100	125 je CPU	125 je CPU
Hauptspeicher	128 MByte je PE	128 MByte je PE
Theor. Gesamtleistung	230 GFlops	36 GFlops
ATM-Karte	Firma Interphase (bis 16.03.98) Fore Runner 200E (ab 16.03.98)	Firma Interphase (bis 21.10.98) Fore Runner 200E (ab 21.10.98)
Betriebssystem	UNICOS/mk	UNICOS/mk

Literatur

[netperf] Hewlett-Packard Corp., *Netperf Homepage*: `http://www.netperf.org`

[PVM] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, Vaidy Sunderan, *PVM: Parallel Virtual Machine, A Users' Guide and Tutorial for Networked Parallel Computing* The MIT Press, Cambridge, Massachusetts, London, England

[ATM] Cheryl D. Krivda, *Analyzing ATM Adapater Performance, The Real-World Meaning of Benchmarks*
`http://www.cs.ucl.ac.uk/research/misa/biblio/atm/EM.ps`

[RFC1483] Juha Heinanen, Telecom Finland, *RFC1483: Multiprotocol Encapsulation over ATM Adaptation Layer 5*

[WIN] WiN-Labor Erlangen `http://www-win.rrze.uni-erlangen.de/`

[Kleyer1] Michael Kleyer. Individuenbasierte Modellierung von Sukzessionen pflanzlicher Wuchstypen bei unterschiedlichen Störungsintensitäten und Ressourcenangeboten. Verhandlungen der Ges. f. Ökologie 28, 1998

[Kleyer2] Michael Kleyer. Vergleichende Untersuchungen zur Ökologie von Pflanzengemeinschaften. Eine Grundlage zur Beurteilung der Ersetzbarkeit in der naturschutzfachlichen Planung am Beispiel einer Agrar- und einer Stadtlandschaft. Diss. Bot. 286 (1997). Bornträger: Stuttgart 202 S.

[Kleyer3] Michael Kleyer. Biological traits of vascular plants. A database. Arbeitsberichte Institut für Landschaftsplanung und Ökologie, Universität Stuttgart, N.F. 2: 23 S., mit Diskette, 1995

[Linke1] Matthias Linke. Programmierung einer Lastverteilungsstrategie für Probleme mit mehreren voneinander unabhängigen Rechenläufen. http://www.uni-rostock.de/rz/hware/cray/bspstr.html

[Linke2] Matthias Linke. Implementation Aspects and Performance of Some Common Problems on the CRAY T3E. Vortrag im Rahmen des 4. European CRAY-SGI MPP Workshop, September 1998, IPP Garching

[CrayPVM] Message Passing Toolkit: PVM Programmer's Manual. SR-2196 1.1. CRAY Research Inc. 1996