Metacomputing
im Gigabit-Testbed Süd und Berlin

Projektabschlussbericht

März 2001

Beteiligte Einrichtungen:

Konrad-Zuse-Zentrum für Informationstechnik, Berlin (ZIB)
URL: http://www.zib.de
Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, München (LRZ)
URL: http://www.lrz.de
Max-Planck-Institut für Festkörperforschung, Stuttgart (MFK)
URL: http://www.mpi-stuttgart.mpg.de
Max-Planck-Institut für Astrophysik, Garching (MPA)
URL: http://www.mpa-garching.mpg.de
Max-Planck-Institut für Polymerforschung, Mainz (MPIP)
URL: http://www.mpip-mainz.mpg.de
Rechenzentrum der Universität Stuttgart (RUS)
URL: http://www.rus.uni-stuttgart.de
Rechenzentrum Garching der Max-Planck-Gesellschaft (RZG)
URL: http://www.rzg.mpg.de

Projektbeteiligte bei den verschiedenen Einrichtungen:

ZIB: H. Busch, Dr. T. Steinke, Dr. D. Reichardt, M. Schünemann
LRZ: Dr. V. Apostolescu, W. Schubring, Dr. R. Ebner, A. Völkl u.a. Mitarbeiter
MFK: Prof. Dr. Parrinello, Dr. W. Silvestri
MPA: Prof. Dr. Hillebrandt, Dr. A. Kercek
MPIP: Prof. Dr. Kremer, Dr. M. Pütz
RUS: M. Resch, M. Müller, P. Haas
RZG: Dr. H. Lederer, Dr. K. Desinger, Dr. A. Seitsonen

Projektdauer: 1. Jan. 1999 bis 31. Dez. 2000

Projektleitung:

Dr. Hermann Lederer, RZG
Max-Planck-Institut für Plasmaphysik, Boltzmannstr. 2, D-85748 Garching
E-Mail: Lederer@rzg.mpg.de

Dieses Vorhaben wurde vom Verein zur Förderung eines Deutschen Forschungsnetzes e.V. unter dem Kennzeichen TK 602 – NT 107 mit Mitteln des Bundesministeriums für Bildung und Forschung gefördert.

Inhaltsangabe

1. Projektübersicht 4

2. Netzverbindung im Gigabit-Testbed 4

2.1 Aufbau der Netzinfrastruktur 4

2.1.1 Aufbau der ATM-Infrastruktur 4
2.1.2 HIPPI-ATM OC12 Umsetzung 5
2.1.3 Native HIPPI-Verbindung: 6

2.2 Die Verbindung der Endgeräte mittels GTB 6

2.2.1 Test und erste Vermessung der funktionierenden Netzwerkverbindung 7
2.2.2 Optimierung der GTB-Endanbindungen 9
2.2.2.1 Optimierung der HIPPI-ATM-Umsetzung 9
2.2.2.2 Untersuchung der HIPPI-Performance von Cray T3E 14
2.2.2.3 Zum HIPPI-Anschluss des Fujitsu VPP-Systems 14
2.2.2.4 Verbindung zwischen VPP-System am LRZ und Cray T3E am RZG 15
2.2.3 Umstellungen am GTB-Backbone 16

3. Metacomputingszenarien mit wissenschaftlichen Anwendungen 17

3.1 Beschreibung der Anwendungen 17

3.1.1 Ab-initio Molekulardynamik 17
3.1.2 Potentialflächen von RNA-Fragmenten 18
3.1.3 Hydrodynamik der Verbrennung in Supernovae und in Motoren 18
3.1.4 Polymerforschung 19
3.1.5 Software zur Kopplung von Parallelrechnern 19

3.2 Homogenes Metacomputing 20

3.2.1 Kopplungs-Software PACX-MPI (RUS) 20
3.2.2 GAMESS-UK (ZIB): 22
3.2.3 Code CPMD (MFK, RZG) 26
3.2.4 Code PROMETHEUS (MPA) 29
3.2.4.1 Beschreibung und Implementierung des Codes 29
3.2.4.2 Einbettung von PROMETHEUS in das Meta-Computing Projekt: 29
3.2.4.3 Implementierung der Kommunikation in PROMETHEUS und daraus resultierende Herausforderungen für die GTB-Leitung: 30
3.2.4.4 Testrechnungen 33

3.3 Untersuchungen im Rahmen von heterogenem Metacomputing 36

3.3.1 Coupling Vector Parallel and Massively Parallel Computers: A Case Study from Polymer Physics (M. Pütz, MPIP) 36
3.3.1.1 Abstract 36
3.3.1.2 Introduction 37
3.3.1.3 Meta-computing 39
3.3.1.4 An Application from Polymer Science: Self-consistent PRISM Calculations 40
3.3.1.5 Coupling VPP and MPP architectures 46
3.3.1.6 Benchmarking and Evaluation 49
3.3.1.7 Conclusions 52
3.3.2 Untersuchungen mit PACX-MPI und Molekulardynamik-Code CPMD 53
3.3.2.1 Anpassungsarbeiten bei PACX-MPI und Test der Verbindung 54
3.3.2.2 Einsatz von CPMD (MFK) 56

4. Zusammenfassung der Arbeiten 59

5. Diskussion 62

Inhaltsangabe

Anhang: Vorarbeiten zur HIPPI-zu-ATM-Umsetzung 64

1. Projektübersicht

Das Projekt diente dem Ziel, im Vorfeld der generellen Einführung des Gigabit-Wissenschaftsnetzes in der BRD in einem geeigneten Gigabit-Testbed (GTB) verteilte Supercomputing-Szenarien mit ausgereiften und real für wissenschaftliche Problemlösungen eingesetzten Anwendungen zu erproben.

Für die verschiedenen Metacomputing-Szenarien wurden Hochleistungsrechner am Konrad-Zuse-Zentrum für Informationstechnik in Berlin, dem Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften in München und dem Rechenzentrum Garching der Max-Planck-Gesellschaft eingesetzt und eng gekoppelt.

Auf der Basis des *Backbones* der Gigabit-Teststrecke von München nach Berlin war eine Netzinfrastruktur zur End-zu-End-Verbindung dieser Rechner aufzubauen, die auch die Nutzungsgrundlage für weitere Gigabit-Testbed-Teilprojekte war.

Für die Erprobung der GTB-Infrastruktur für Metacomputingzwecke kamen Anwendungen aus der Materialforschung, der Astrophysik, der Chemie und der Polymerphysik zum Einsatz, für die Rechnerkopplung eine Softwarebibliothek der Universität Stuttgart.

Für homogene Metacomputing-Studien wurden die gleichartigen Rechnersysteme Cray T3E am RZG in Garching und am ZIB in Berlin eingesetzt, für heterogene Metacomputing-Studien wurde das Fujitsu VPP-System am LRZ mit einbezogen. Beim heterogenen Metacomputing wurde auch der Aspekt der Verteilung von zwei Aufgaben einer Gesamtrechnung auf zwei verschiedene Rechnerarchitekturen, die aufgrund ihrer Spezialeigenschaften für die jeweilige Aufgabe besonders geeignet sind oder sogar benötigt werden, betrachtet.

2. Netzverbindung im Gigabit-Testbed

2.1 Aufbau der Netzinfrastruktur

Der Aufbau der Netzinfrastruktur basierte vorgegeben auf ATM-Technologie. Da es für die im Metacomputing-Verbund einzusetzenden Rechner am RZG, LRZ und ZIB keine ATM OC12-Interfaces gibt, sondern die leistungsfähigsten Interfaces auf HIPPI-Technik beruhen, waren an den drei Rechenzentren HIPPI-ATM OC12-Umsetzungen zu realisieren, optional war eine native HIPPI-Verbindung vorgesehen. Weiterhin war an RZG und ZIB der Aufbau einer lokalen ATM-Infrastruktur nötig.

2.1.1 Aufbau der ATM-Infrastruktur

Am RZG war bereits vor Beginn dieses Projektes ein ATM-Switch von Fore Systems mit 2 OC12 Interfaces und einem 4-Port OC3-Interface beschafft worden, um einerseits den GTB-Anschluß der Cray T3E mit ATM OC3-Interface via ATM-Switch für das früher begonnene

Teilprojekt 1.5 (EM & HPCN) des MPI für Biochemie realisieren zu können, andererseits erste OC12-Testverbindungen zum GTB-Aufpunkt des LRZ in Garching schalten zu können.
Die beiden beschafften Multimode OC12 Interfaces wurden in die beiden ATM-Switches am RZG und LRZ in Garching eingebaut und die 700 m lange Glasfaserstrecke getestet, für die normalerweise wesentlich teurere Monomode Ports einzusetzen wären. Da die Verbindung jedoch einwandfrei funktionierte, wurde dem LRZ ein Multimode Port überlassen, für das RZG ein weiterer zum Anschluß der SUN Workstation für HIPPI-ATM-Umsetzung bestellt (siehe nächster Punkt). Für den Anschluss der Visualisierungsworkstation für das TIKSL-Projekt (TK 602 – NT 105), die Ende August 1999 geliefert wurde, war noch ein weiterer ATM OC12 Port zu beschaffen. Damit waren sämtliche Einschübe am ATM-Switch belegt.
Am ZIB erfolgte der Aufbau der ATM-Infrastruktur in analoger Weise. Allerdings wird hier nicht ein ATM-Switch von der Fa. Fore genutzt, sondern ein ATM-Switch CISCO 8540 MSR mit acht OC12-Ports, von denen für dieses Projekt die Außenverbindung zur Technischen Universität Berlin (TUB) und die Inhausverbindung über das SUN-Gateway zur Cray T3E genutzt werden (siehe nächster Absatz). Dieser Switch wird auch für das Projekt TIKSL sowie für regionale Verbindungen innerhalb der Region Berlin/Potsdam genutzt. Näheres hierzu wird im Bericht zu GÜTE ausgeführt. Die Außenverbindung über die TUB nach Erlangen und München konnte bereits im März 1999 anläßlich des Symposiums „Fortgeschrittene Kommunikationstechnik“ des DFN-Vereins mit einer Videokonferenz in das Klinikum „Rechts der Isar“ erfolgreich eingesetzt werden.

2.1.2 HIPPI-ATM OC12 Umsetzung

Für diese Umsetzung gab und gibt es immer noch keinen einwandfrei funktionierenden kommerziellen Switch. Im Mai 1998 war uns von der Fa. SGI eine funktionierende Lösung auf Workstation-Basis (Octane mit HIPPI und ATM OC12 Interface) versprochen worden.
Diese Lösung floss in den Projektantrag ein. SGI hatte jedoch Schwierigkeiten bei der Realisierung, so daß eine Alternative gefunden werden musste. In Zusammenarbeit mit Peter Haas vom Rechenzentrum der Uni Stuttgart wurde im Nov. 1998 versucht, eine solche Lösung auf der Basis einer SUN-Workstation zu testen. Der Test schlug jedoch fehl. Ein neuerlicher Test wurde für Anfang 1999 geplant. Nach Vorsprache bei SUN Microsystems Deutschland und Fore Systems wurde Herrn Haas kostenlos das benötigte Testequipment überlassen. Der darauf folgende Test im März 1999 verlief erfolgreich und zeigte ab Puffergrößen von 256 kB eine Durchsatzbandbreite von 38 MB/s ansteigend bis auf 65 MB/s bei einer Puffergröße von 4 MB (siehe Protokolle von Peter Haas/RUS im Anhang).
Daraufhin wurden von RZG, LRZ und ZIB identische Geräte bestellt und im Mai/Juni 1999 ausgeliefert. Die Workstations von RZG und LRZ wurden mit Interfaces installiert und (teilweise) mit HIPPI-Modem (für die Kupfer-Glasfaser-Umsetzung) an die HIPPI-Interfaces der jeweiligen Supercomputer angeschlossen, andererseits an die Endpunkte der GTB-Leitungen bei den Einrichtungen. Die Lieferung des HIPPI-Modems für das ZIB und damit die Anbindung der Cray T3E im ZIB erfolgte mit einigen Wochen Verzögerung.
Erste qualitative Tests mit Subnetzaddressen aus dem 10er-Netz ergaben zwischen RZG und LRZ bereits Übertragungsraten von über 40 MB/s zwischen den ATM-Interfaces. Fine-Tuning und Rechner-zu-Rechner-Tests konnten erst erfolgen, nachdem die von EANTC / TUB zugesagten IP-Adressen vorlagen.

2.1.3 Native HIPPI-Verbindung:

Wie im Technologiebericht von Dr. Apostolescu vom 30. 6. 1999 unter „Hippi-Connectivity“ dargelegt, wurde diese Option wegen Problemen bei einer Distanz von ca. 800 km und einer hierbei maximal zu erwartenden Durchsatzrate von 40 Mbit/s aufgegeben.

2.2 Die Verbindung der Endgeräte mittels GTB

Die Installation von Netzwerkkomponenten bei den beteiligten Einrichtungen mit Hochleistungsrechneranbindung (LRZ, RZG) war in der ersten Hälfte 1999 abgeschlossen worden, so dass in die Phase der Einrichtung der Verbindungen zwischen den Rechnern mittels GTB und in die sich anschließende Phase des Performance-Tunings eingetreten werden konnte. Die Anbindung der Cray T3E des ZIB an das GTB-Netz konnte nach verspäteter Lieferung des HiPPI-LWL-Kupfer-Umsetzers im September 1999 erfolgen.
Die de-facto Verbindung der Endgeräte via GTB erwies sich jedoch als komplexe und zeitraubende Aufgabe. Zunächst war die Zuständigkeit über die Vergabe von Class C Subnetzadressen zwischen DFN-Verein und den anderen GTB-Betreibern zu klären. Im Sept. 1999 wurde dann zunächst vom EANTC ein Subnetz für Classical IP bereitgestellt. Nach Bedarfsmeldung wurde ein weiteres Subnetz für HIPPI-Adressen bereitgestellt. Da es den GTB-Betreibern nicht gelang, ein funktionierendes CLIP-Netz mit dynamischem ARP zur Verfügung zu stellen, wurde von Anwenderseite auf die Einrichtung von PVCs gedrängt, um Mitte Nov. 1999 eine Demonstration auf der Supercomputer Conference 99 in Portland, Oregon, zu ermöglichen (E. Seidel).
Diese PVCs wurden dann von EANTC / Berlin geschaltet, und 8. November 1999 stand den Anwendern schließlich eine funktionierende GTB-Verbindung zwischen den Endsystemen zur Verfügung.

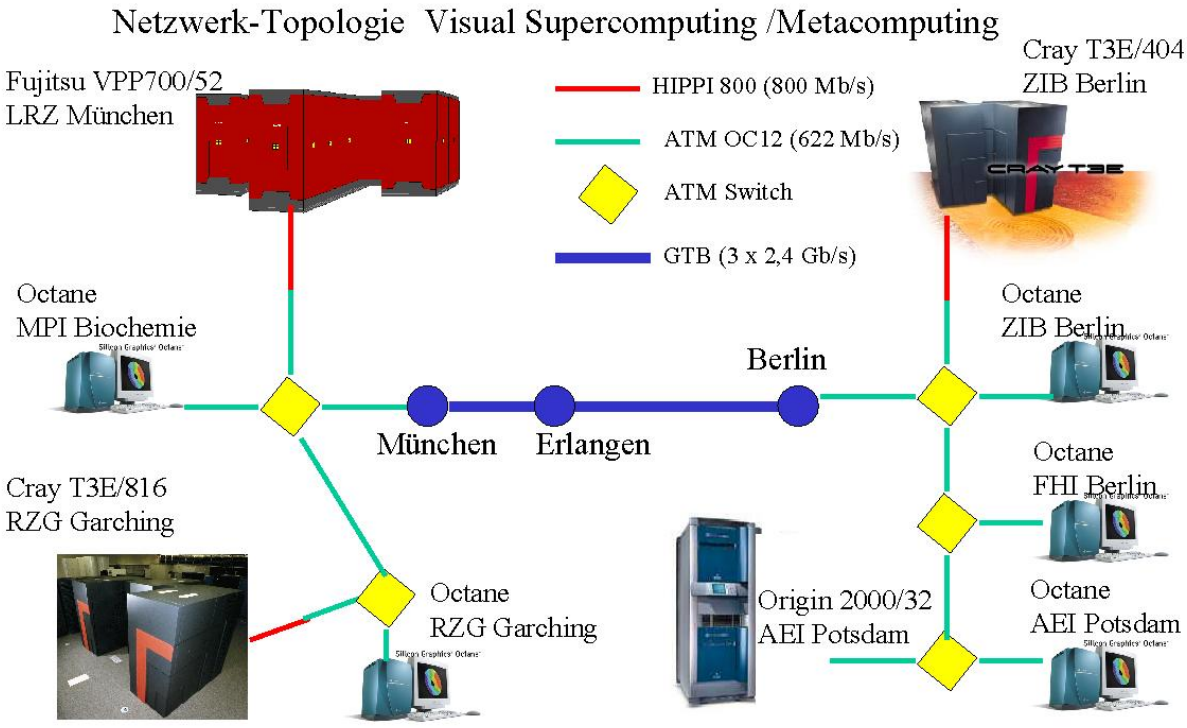

Abbildung N 1
Verbindung von Endgeräten in Garching, München, Berlin (und Potsdam) via GTB

2.2.1 Test und erste Vermessung der funktionierenden Netzwerkverbindung

Erste Messungen zur erzielbaren Bandbreite (anhand von großen Datenpaketen) zwischen den Cray T3E-Systemen in Berlin und Garching verliefen enttäuschend: Nur ca. 95 Mbit/s wurden erzielt, d.h. weniger als 15% der theoretischen Peakperformance von 622 Mbit/s.
Nun begann eine komplexe Analyse der Rolle der verschiedenen Netzwerkkomponenten und ihrer Konfigurationen, mit einer ganzen Reihe von Messungen von Punkt-zu-Punkt-Verbindungen (K. Desinger).

Zum Verständnis des nachfolgenden Diagramms erfolgt eine Erläuterung der verwendeten Hostnames der eingesetzten Rechner anhand des vorhergehenden Bildes:

pc:	T3E am RZG / Garching	(HIPPI-Endpunkt)
berte:	T3E am ZIB / Berlin	(HIPPI-Endpunkt)
gigue:	ATM-to-HIPPI Router am RZG	(SUN WS mit HIPPI und OC12 Interfaces)
zib-sun:	ATM-to-HIPPI Router am RZG	(SUN WS mit HIPPI und OC12 Interfaces)
spinat:	Octane am RZG	(ATM OC12-Endpunkt)

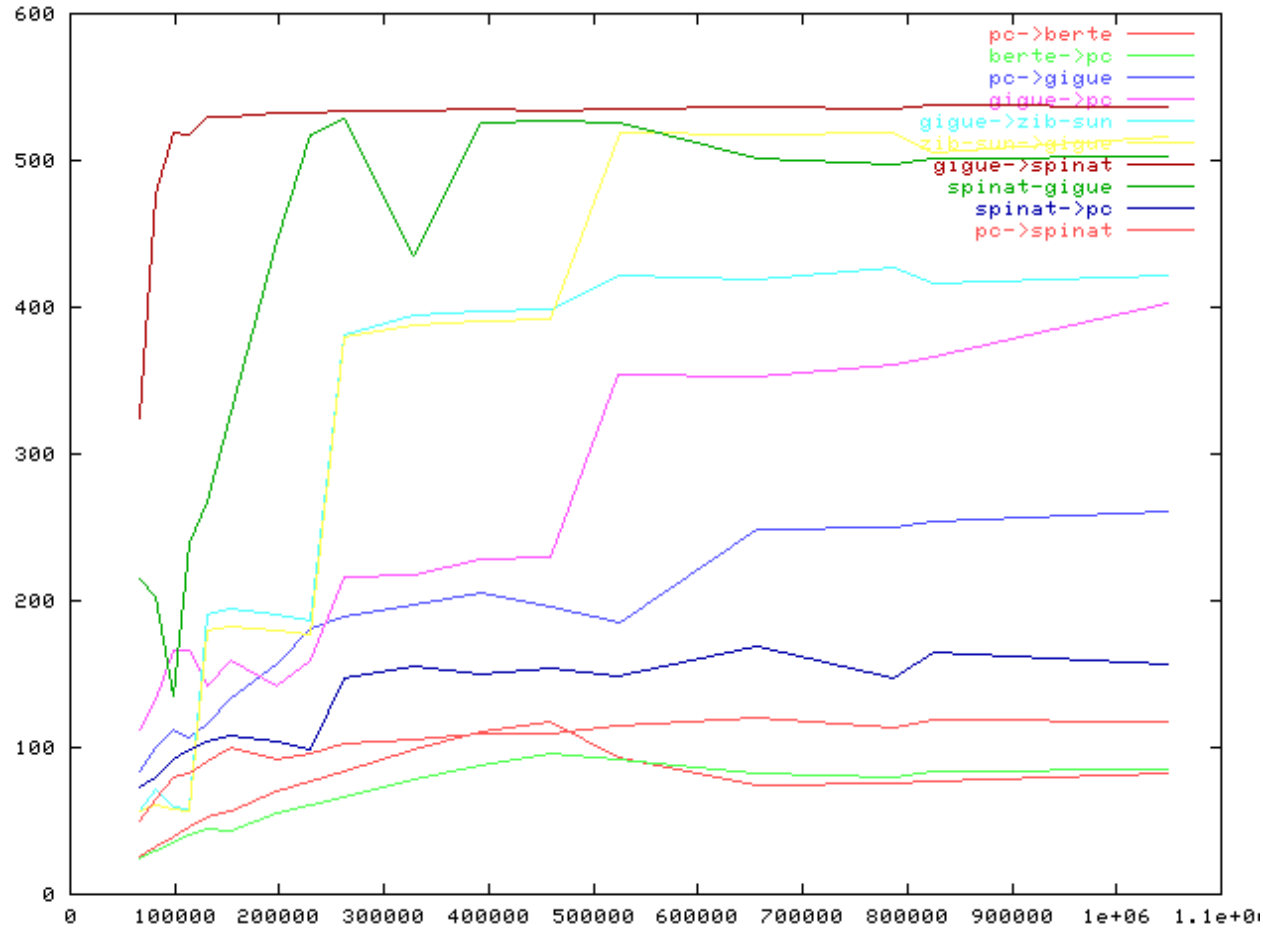

Abbildung N 2

Gemessene TCP-Transferraten zwischen beteiligten Komponenten zum Zeitpunkt der ersten Funktionsfähigkeit des GTB für die Endanwender (Nov. 1999) (Messprogramm: netperf)

Abszisse: *Puffergröße in Byte*

Ordinate: *Transferrate in Mbit/s*

Die Kontrollmessung zwischen den Maschinen gigue und spinat mit zwei ATM OC12 Interfaces lokal am ATM-Switch am RZG mit 500 bis 530 (je nach Richtung) von theoretischen 622 Mbit/s zeigt das prinzipiell korrekte ATM-Setup.
Zwischen den beiden SUNs am RZG und ZIB zeigt sich noch eine Asymmetrie bei der reinen ATM-Verbindung. Mit einer Rate von 400 bzw. 500 Mbit/s ist jedoch die erwartete Funktionalität des GTB nachgewiesen.
Auf der lokalen HIPPI-Verbindung gigue-pc wurden 350 Mbit/s für 512 kB Päckchen mit SUN (gigue) als Sender erreicht, mit T3E (pc) als Sender jedoch nur 240 Mbit/s.
Auf einer lokalen RZG-Verbindung spinat-pc von ATM auf Octane (über den Sun-Umsetzer) auf HIPPI auf T3E wurden nur noch bis zu 140 Mbit/s gemessen, in der Gegenrichtung (pc-spinat) mit T3E als Sender gar nur bis zu 110 Mbit/s. Für die T3E-Verbindungen zwischen Garching und Berlin (pc-berte und berte-pc) wurden nur noch enttäuschende 90 Mbit/s erzielt.

Aus dieser Messserie ergeben sich folgende Schlussfolgerungen:

a) Der GTB-Backbone ist nahezu verlustfrei nutzbar.
b) Es gibt ein Problem mit der HIPPI-zu-ATM Umsetzung in den SUN-WS.
c) Die Cray T3Es zeigen starke Performance-Unterschiede beim Senden und Empfangen, die HIPPI-Performance lässt ingesamt zu wünschen übrig.

2.2.2 Optimierung der GTB-Endanbindungen

2.2.2.1 Optimierung der HIPPI-ATM-Umsetzung

Aufgrund der Performance-Probleme bei der HIPPI-ATM-Umsetzung wurde die Konfiguration der Interfaces analysiert.
Als einer der Gründe für die schlechte End-zu-End-Leistung erwies sich die stark unterschiedliche MTU-Größe bei HIPPI (64 kB) und Standard-CLIP (9 kB). Eine HIPPI-Übertragungseinheit besteht aus maximal 68 „Bursts“ zu je 1 kB. Laut Spezifikation können zwar auch weniger „Bursts“ gesendet werden, aber bei den Implementationen ergeben sich offensichtlich erhebliche Geschwindigkeitseinbußen, wenn die volle MTU nicht ausgenutzt wird.

Abschalten der „Path-MTU-discovery“, die in diesem Fall die verfügbare MTU auf 9 kB begrenzt, löst das Problem nicht: Auf der lokalen HIPPI-Strecke wird zwar mit voller MTU übertragen, und die SUN-Workstations haben auch keine Performance-Probleme mit der Fragmentierung, aber die entfernte HIPPI-Strecke wird durch die kleinen Fragmente nicht optimal ausgenutzt, da das „Re-Assembly“ erst auf dem Ziel-Rechner erfolgt.

Eine deutliche Steigerung des Durchsatzes wurde dagegen durch die Konfiguration einer höheren MTU auf der CLIP-Strecke erreicht, so dass durchgängig die volle HIPPI-Paketgröße genutzt werden kann. Diese Einstellung war bereits von Herrn Peter Haas vom RUS vorgeschlagen worden. Obwohl sie im Testbed funktioniert, ist davon auszugehen, dass sie in einer „best-effort“ Umgebung eher von Nachteil ist, da durch den Verlust nur einer einzigen ATM-Zelle gleich 64 kB Daten verloren gehen.

Nunmehr stellten die HIPPI-Netzanbindungen auf den T3E-Systemen die leistungsbegrenzenden Faktoren dar: der Datenstrom von T3E zum HIPPI-Interface der SUN-WS für ATM-Umsetzung wurde ohne weitere Einbuße zu ATM-Empfängern weitergeleitet. Auf diese Weise konnte eine Transferrate bis zu 240 Mbit/s, also 30 MByte/s für große Datenpakete (ab 512 kB) zwischen den T3Es in Garching und Berlin erzielt werden, allerdings mit einem künstlichen Testprogramm (netperf), nicht mit einer realen Supercomputeranwendung. Dies entspricht ca. 1/3 der HIPPI 800-Peakleistung.

Im ZIB wurden ebenfalls Messungen zur GTB Durchsatzleistung vorgenommen (M. Schünemann).
Nach Übernahme der MTU-Sizes der Garchinger SUN-WS auf zib-sun wurde zunächst eine Messreihe mit dem Messtool nettest durchgeführt. Die gemessenen Bandbreiten (in Mbit/s) sind in der nachfolgenden Tabelle N 1 dargestellt.

	Empfänger			
	pc	**gigue**	**zib-sun**	**berte**
Sender				
pc	-	262	250	203
gigue	336	-	339	291
zib-sun	338	345	-	-
berte	198	283	292	-

Tabelle N 1 *Am ZIB (mit nettest) gemessene Transferraten im GTB in Mbit/s*

Diese Werte lassen ebenfalls eine Verbesserung der Performance erkennen, sind allerdings doch noch weit entfernt von den ca. 540 Mbit/s, die man rein theoretisch nach Abzug eines Protokoll-Overheads von der 622 Mbit/s Strecke erwartet. Zu dieser Zeit wurde noch geprüft, ob ein Update des Betriebssystems der Crays (von unicos/mk 2.0.4. auf 2.0.5) zu einer Verbesserung der Performance führt. Unicos/mk 2.0.5 erlaubt die Verteilung der Netzwerkanwendungen auf mehrere PEs (Processing Elements), bisher liefen alle Socket-Verbindungen nur über einen speziellen PE, was zu Performance-Einbußen geführt haben könnte.
Eine Untersuchung der Latenzzeiten ergab gute Werte zwischen den SUN-WS, die mit 9 ms den vom WIN-Labor in Erlangen gemessenen reinen ATM-Werten von 7,77 ms recht nahe kommen. Für die Direktverbindung zwischen den beiden T3Es wurden leicht vergrößerte Latenzzeiten von ca. 13,5 ms gemessen. Allerdings lagen diese Zeiten noch deutlich unter den Latenzzeiten von ca. 20 ms, die über das B-Win erreicht werden konnten.
Für weitere Analysen unter Einbeziehung der beiden SUN-Vorrechner wurde dann das von Hewlett-Packard entwickelte Programm netperf für Tests auf TCP/IP-Ebene eingesetzt. Gemessen wurde die Bandbreite in Abhängigkeit von der Socketbuffergröße, die dem maximalen TCP-Window entspricht, sowie die Latenzzeit (round-trip-time).

Die damit im GTB erzielbaren Durchsatzbandbreiten wurden am ZIB projektbegleitend mitverfolgt. Da die Messungen nicht in einer isolierten Umgebung stattfanden, sind die Ergebnisse sowohl von der durch andere Anwender erzeugten System- und Netzlast beeinflusst, als auch von den Konfigurationseinstellungen bzw. -änderungen (z.B. Größe der Maximum Transfer Unit (MTU)) der Administratoren. Dieser Umstand erschwert die Interpretation der Messergebnisse, da sich die Konfiguration der Umgebung während des Projekts häufiger änderte.
Die für ein Netzwerk optimale TCP-Window-Größe läßt sich theoretisch durch Multiplikation der minimalen Bandbreite (bottleneck) mit der Latenzzeit bestimmen. Für das vorliegende Netzwerk haben wir eine minimale Bandbreite auf der Strecke zwischen dem Sun-Vorrechner im ZIB und dem EANTC in der TU-Berlin von 622 Mbit/s, multipliziert mit einer Latenzzeit von 15 ms zwischen den beiden T3Es ergibt sich ein Rechenwert von ca. 1 Mbyte für das TCP-Fenster. Auf diesen Wert wurden die maximalen Send- und Receivebuffer und die MTU von den Administratoren gesetzt. Mit dem Programm netperf kann man die Messungen mit verschiedenen Socketbuffergrößen durchführen. Es wurden Größen bis zu 1 Mbyte gewählt. Wie erwartet wurde der höchste Durchsatz mit Socketbuffergrößen von 1 Mbyte erzielt. In der folgenden Abbildung N 3 ist der Einfluss der Socketbuffergrößen auf den Durchsatz zwischen den SUN-Vorrechnern dargestellt.

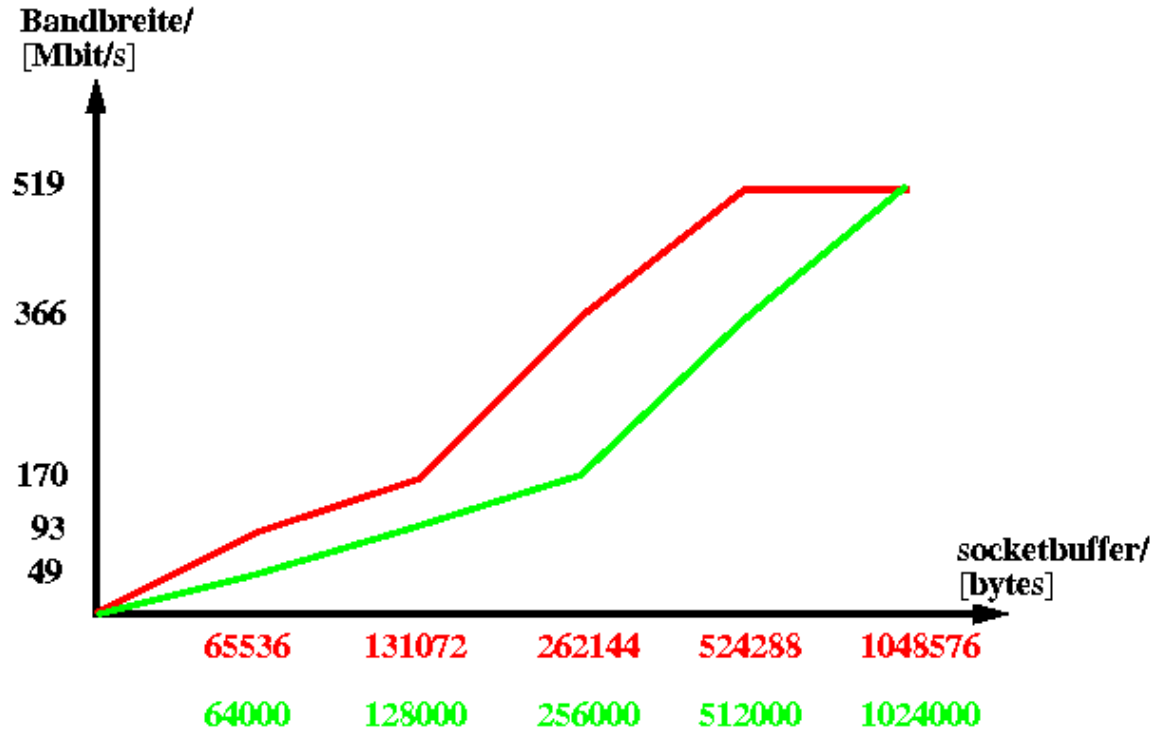

Abbildung N 3
Am ZIB gemessene Transferraten zwischen den SUN-Vorrechnern gigue (am RZG) und HIPPI (= zib-sun am ZIB) in Abhängigkeit von Socketbuffergrößen

Die folgende Tabelle gibt die während der Projektlaufzeit maximal gemessenen Bandbreiten zwischen den T3Es und den Sun-Vorrechnern bei RZG und ZIB wieder (mit z.T. unterschiedlich groß gewählten Socketbuffern).

Sender	***Empfänger***			
	berte	***pc***	***zib-sun***	***gigue***
berte		272 Mbit/s	433 Mbit/s	286 Mbit/s
pc	282 Mbit/s		270 Mbit/s	282 Mbit/s
zib-sun	386 Mbit/s	294 Mbit/s		519 Mbit/s
gigue	279 Mbit/s	364 Mbit/s	516 Mbit/s	

Tabelle N 2 *Am ZIB während der Projektlaufzeit gemessene maximale Bandbreiten zwischen den T3Es pc (RZG) und berte (ZIB) sowie den Sun-Vorrechnern gigue (RZG) und zib-sun (ZIB)*

Der Durchsatz zwischen den T3Es fällt geringer aus als zwischen den Suns.

Der Wert von ca. 282 bzw. 272 Mbit/s wurde im Mai 2000 ermittelt und liegt allerdings deutlich höher als zu Beginn der Messungen, der Wert im November 1999 lag bei ca. 90 Mbit/s. Die folgende Abbildung zeigt den Verlauf des, mit einer Socketbuffergröße von 1 Mbyte, ermittelten Durchsatzes im Verlauf des Projektes (die Messung im Dezember wurde mit dem Craytool nettest durchgeführt statt mit netperf).

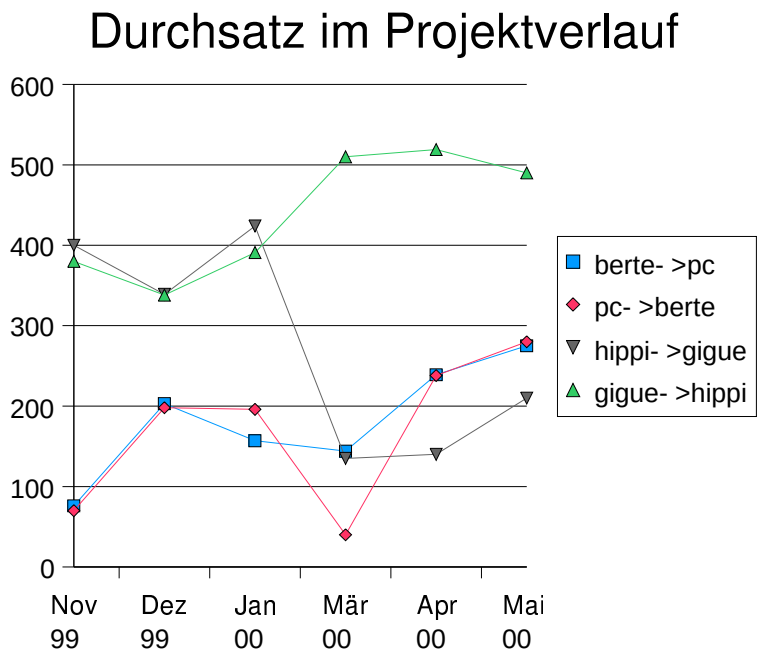

Abbildung N 4
Am ZIB gemessenr GTB-Durchsatz im Projektverlauf
(Transferraten in Mbit(s) zwischen den Cray T3Es pc (RZG) und berte (ZIB) sowie den SUN-Vorrechnern gigue (am RZG) und hippi (= zib-sun am ZIB)

Aufällig bei den Messergebnissen im März 2000 ist, dass in der Richtung von zib-sun (hippi) nach gigue zwar mit ca. 519 Mbit/s eine sehr gute Performance erzielt wurde, in der umgekehrten Richtung von gigue nach zib-sun (hippi) mit ca. 150 Mbit/s hingegen eine sehr niedrige Leistung. Gemessen wurde mit einer Socketbuffergröße von 1 Mbyte.
Weitere Tests ergaben dann, daß für diese Richtung ein optimaler Durchsatz erzielt wird, wenn für die Socketbuffergröße 524288 bytes und für die Größe der Message 512000 bytes gewählt wird, der Durchsatz lag dann bei 516 Mbit/s. Diese Tatsache ließ sich leider nicht aufklären, war allerdings auch für das Projekt nicht relevant, da für die Anwendungen eine TCP-Verbindung von T3E zu T3E wichtig war und nicht zwischen den Suns.
Der ermittelte Durchsatz von T3E zu T3E war in beide Richtungen unabhängig von der Wahl der Socketbuffergrößen und Messagegrößen symmetrisch.
Die gemessenen Latenzzeiten sind in der nachfolgenden Tabelle dargestellt.Die Latenzzeit zwischen den Suns beträgt ca. 9 ms, zwischen den T3Es ca. 15 ms. Dies zeigt, daß durch die Umsetzung auf den Suns und evtl. auf den T3Es eine zusätzliche Verzögerung entsteht.

Sender	***Empfänger***				Wert in
	berte	***pc***	***zib-sun***	***gigue***	[ms]
berte	2,28	15,44	3,61	12,47	Minimum
	2,97	16,53	4,03	12,91	Mittelwert
	7,46	20,75	5,02	15,19	Maximum
pc	15,33	7,75	12,74	3,97	Minimum
	17,23	9,74	13,38	4,48	Mittelwert
	26,41	33,47	23,62	7,53	Maximum
zib-sun	3,35	12,76	0,10	9,21	Minimum
	3,73	13,63	0,11	9,23	Mittelwert
	4,62	29,03	0,11	9,25	Maximum
gigue	12,19	3,92	9,21	0,09	Minimum
	12,71	4,63	9,23	0,10	Mittelwert
	14,88	10,83	9,25	0,10	Maximum

Tabelle N 3 *Am ZIB gemessene GTB-Latenzzeiten zwischen den T3Es pc (RZG) und berte (ZIB) sowie den Sun-Vorrechnern gigue (RZG) und zib-sun (ZIB)*

Zum Schluss wurde noch ein Vergleich gezogen zwischen einer Datenübertragung mittels ftp über das B-WIN und über das GTB. Die Dauer der jeweiligen Übertragung von einer 24 Mbyte großen Nachricht ist in der nachstehenden Tabelle aufgeführt.

	B-WIN	***Gigabit-Testbed***
berte->pc	23 ms	1,7 ms
berte->zib-sun	8,9 ms	3,7 ms
pc->berte	23 ms	1,7 ms
pc->gigue	6,8 ms	3,8 ms
zib-sun>berte	4,8 ms	1,3 ms
zib-sun->gigue	-	4,6 ms
gigue->pc	11 ms	1,2 ms
gigue->zib-sun	-	5,2 ms

Tabelle N 4 *Am ZIB durchgeführter Vergleich zwischen B-WiN und GTB Datentransferraten mit ftp (Datenmenge 24 MB).*
Rechner: T3Es pc (RZG) und berte (ZIB) sowie Sun-Vorrechner gigue (RZG) und zib-sun (ZIB)

Wie erwartet lassen sich Daten im Gigabit-Testbed sehr viel schneller übertragen als im B-WIN. Auffällig ist, daß die Übertragung zwischen den T3Es schneller ist als zwischen den Suns. Vermutlich ist die Verarbeitung im Protokollstack auf der Sun-Workstation langsamer als auf der T3E, auch unterschiedliche Diskgeschwindigkeiten mögen eine Rolle spielen.

2.2.2.2 Untersuchung der HIPPI-Performance von Cray T3E

Es wurden unabhängig voneinander alle in Frage kommenden, das HIPPI-Interface betreffenden System-Einstellungsparameter auf den Cray T3E-System am RZG und ZIB untersucht und gegenseitig abgeglichen, auch unter Zuhilfenahme von Cray System-spezialisten an den jeweiligen Zentren. Leider war durch diese Maßnahmen keine Performance-Verbesserung erzielbar. Vom HIPPI-Interface eines IBM SP2-Knotens konnte hingegen zum selben HIPPI-SUN-Interface unmittelbar eine Datenrate von 450 Mbit/s erzielt werden, ohne jeglichen Optimierungsaufwand.
Um die Ursache für den immer noch nicht zufriedenstellenden Durchsatz zu finden, wurde ein „Trace“ der TCP-Pakete eingeschaltet und die Sequenz-Nummern von Daten- und Acknowledge-Paketen zeitaufgelöst untersucht. Es zeigte sich, dass

- die sendende T3E auch auf mikroskopischer Zeitskala die Pakete nicht in der maximal möglichen Rate abschickt.
- immer wieder "Pausen" von ca. 16 ms vorkommen, die offensichtlich durch die TCP-Flusskontrolle bedingt sind. Es konnte noch nicht eindeutig geklärt werden, ob die „ACK“-Pakete verzögert abgeschickt, unterwegs aufgehalten oder von der sendenden T3E verzögert zur Kenntnis genommen werden.

Zur Klärung beider Punkte wurden Fachleute von Cray/SGI aus USA mit einbezogen.
Auch dadurch konnte keine weitere Verbesserung erzielt werden. Die Fachleute behaupteten zwar, 50 MB/s bereits realisiert zu haben, waren aber nicht in der Lage, diesen Wert auf dem RZG-System oder dem ZIB-System zu erzielen. Somit stellt die HIPPI-Performance der Cray T3E-Systeme den Engpass bei der GTB-Kommunikation dar. Die HIPPI-Performance der Cray T3E-Systeme lag bei maximal 30 MB/s, wohingegen ATM OC12 die doppelte Bandbreite zuließe.

2.2.2.3 Zum HIPPI-Anschluss des Fujitsu VPP-Systems

Die Messungen zum Anschluss des am Projekt beteiligten LRZ-Hochleistungsrechners Fujitsu VPP700 wurden am LRZ vorgenommen (A. Völkl).
Die Verbindung des SUN Vorrechners für die HIPPI-ATM-Umsetzung zu den entsprechenden SUNs am RZG und am ZIB funktionierte quantitativ auf demselben Level wie zwischen den SUN-Rechnern am RZG und ZIB nach der Optimierungsphase. Alle drei SUN-Rechner erhielten identische Konfigurationen zu den ATM Netz-Interfaces. Untersuchungen konzentrierten sich nun auf die Leistung der HIPPI-800 Verbindung zwischen dem Fujitsu VPP700-System zum SUN-Vorrechner.
Hierzu erfolgte zunächst eine Optimierung auf der SUN-Seite, dann auf der VPP-Seite. Auf der SUN-Seite wurden Leistungen von 400 Mbit/s mit TCPIP möglich. Auf VPP-Seite war es nicht möglich, so hohe Leistungswerte zu erzielen.
Die Tests wurden mit einem kleinen C Programm von Hitachi durchgeführt, das im Rahmen von Tests zum HIPPI-Anschluss des neuen Hitachi SR8000 Rechners geschrieben worden war. Umfangreiche Zusatzmessungen wurden mit dem Programm Netperf durchgeführt, die die Messungen mit dem C Programm bestätigten.

Send und Receive Socket Size auf VPP (kB)	Durchsatz (Mbit/s) LRZ-SUN -> VPP	Durchsatz (Mbit/s) VPP -> LRZ-SUN
8 (Standardeinstellung)	24	120
100	64	144
1000	160	184

Tabelle N 5
Am LRZ gemessener HIPPI-Durchsatz zwischen VPP-System und SUN-Rechner in Mbit/s

Die Nachrichtengröße war jeweils 10 MB, die Send und Receive Socket Size auf der LRZ-SUN war jeweils 1024 Kbytes.
Auf der VPP empfahl es sich, die Send und Receive Socket Size auf 1024 Kbyte zu erhöhen. Dies könnte allerdings mit dem Nachteil verbunden sein, dass zu viel Speicher beansprucht wird. Denn fast alle Messungen und Optimierungen betrachten den Durchsatz, im realen Betrieb muss dies aber nicht zu wirklichen Verbesserungen führen. Durch die hohen Puffergrößen könnten evtl. einzelne Prozesse zuviel Speicher beanspruchen und dadurch das Gesamtsystem "bremsen".

2.2.2.4 Verbindung zwischen VPP-System am LRZ und Cray T3E am RZG

Die Leistungsfähigkeit der GTB-Verbindung zwischen dem VPP-System am LRZ und dem T3E-System am RZG wurde ebenfalls mit Netperf gemessen (A. Seitsonen). Beide Richtungen wurden vermessen. Bei VPP als Sender wurden zwei verschiedene Knoten untersucht: Knoten VPP00, der das relevante HIPPI-Interface beherbergt, und Knoten VPP01, der mit VPP00 über das leistungsfähige Fujitsu Crossbar-Netzwerk mit deutlich über HIPPI liegender Leistung verbunden ist. Die erzielten Ergebnisse sind der folgenden Tabelle N 6 zu entnehmen.

	Datenpaket-größe 0.5 MB	Datenpaket-größe 1 MB	Datenpaket-größe 2 MB	Datenpaket-größe 10 MB
T3E - > VPP0	182	160	181	183
VPP0 -> T3E	79	65	65	63
VPP1 -> T3E	135	131	133	130

Tabelle N 6
Mit Netperf gemessene Transferraten in Mbit/s zwischen Cray T3E am RZG und Fujitsu VPP-System am LRZ via GTB.
(VPP0: VPP-Knoten mit HIPPI-Interface; VPP1: zweiter VPP-Knoten, mit Fujitsu Crossbar an VPP0 mit höherer als HIPPI-Leistung angeschlossen;
Auf T3E und VPP gewählte Send/Receive Socket Size: 1 MB)

Der Unterschied vom Faktor 2, mit dem VPP01 besser als VPP00 mit dem GTB-Netzinterface Daten zur T3E senden kann, lässt sich auf den Umstand zurückführen, dass der

Knoten VPP00 wesentliche Systemaufgaben für das VPP-Gesamtsystem übernehmen muss und deshalb mit einer wesentlich höheren Grundlast belastet ist. Auf VPP01 mit geringerer Grundbelastung, auf der aber wegen der guten Netzanbindung zu VPP00 keine Verschlechterung bei der Übertragung zu erwarten ist, sieht man deshalb auch bessere Werte von ca. 130 Mbit/s. Von T3E zu VPP00 wurden ca. 180 Mbit/s erzielt.

Aufgrund dieser Messungen wird deutlich, dass die tatsächlich zu erzielenden Übertragungsleistungen stark von der momentanen Auslastung des VPP00 Knotens abhängen und bei symmetrischen Anforderungen ein Wert zwischen 65 und 130 Mbit/s erwartet werden kann, auf einer lastfreien Maschine ein Wert bis zu 180 Mbit/s.

2.2.3 Umstellungen am GTB-Backbone

Im Frühjahr 2000 wurden anlässlich der Cebit-Veranstaltung in Hannover die statischen PVCs von EANTC/Berlin durch dynamische SVCs erfolgreich ersetzt. Allerdings waren an verschiedenen Orten in Switches Einstellungen zu ändern, bis alle Verbindungen wieder voll funktionsfähig waren.

3. Metacomputingszenarien mit wissenschaftlichen Anwendungen

Im ersten Schritt wurden mit drei wissenschaftlichen Anwendungen verteilte Rechnungen über das GTB zwischen Garching und Berlin im Rahmen von homogenen Metacomputingstudien durchgeführt. Im zweiten Schritt wurde das Vektorsystem VPP700 am LRZ für heterogene Metacomputingstudien mit einbezogen.

3.1 Beschreibung der Anwendungen

Die wissenschaftlichen Anwendungen entstammen den Bereichen Astrophysik, Polymerphysik, Chemie und Materialwissenschaften.
In fast allen Fällen lagen bereits parallele Codes vor, die für massiv-parallelen Einsatz geeignet sind und auf den jeweiligen Hochleistungsrechnern im Produktionseinsatz sind. Diese Codes dienten als Ausgangsbasis für die zu bearbeitenden Fragestellungen.

3.1.1 Ab-initio Molekulardynamik

Anwendung des MPIs für Festkörperforschung, Stuttgart, Abt. Prof. M. Parrinello

Ab-initio Molekulardynamik erlaubt die Simulation von komplexen, dynamischen chemischen Prozessen. Sie wird heute in Deutschland an Universitäten, Forschungseinrichtungen und in der chemischen Industrie häufig eingesetzt. Fortschritte in der Effizienz und der Verarbeitung der Resultate der Simulationen haben direkte Konsequenzen in Wissenschaft und Praxis. In der näheren Zukunft werden neue Methoden entwickelt werden, die es erlauben werden, große biologische Systeme zu simulieren.

Die kürzliche Einführung der Pfadintegralmethode ermöglicht die Simulation von Systemen, bei denen Quanteneffekte der Atomkerne eine wichtige Rolle spielen. Solche Systeme sind z.B. flexible Moleküle, fester Wasserstoff und die sehr wichtige Klasse chemischer und biologischer Protonentransfersysteme. Diese Anwendungen erzeugen eine Fülle detaillierter Informationen und wertvoller neuer Einsichten.
Sie sind aber auch extrem aufwendig bezüglich ihrer Anforderungen an Rechenzeit und Speicherbedarf.
Vom technischen Standpunkt ist diese Anwendung besonders für verteiltes Rechnen geeignet, da die hierarchische Struktur des Codes ausgenützt werden kann. So wird auf einer tieferen Stufe separat voller Nutzen aus verteilten Rechnern in Garching, Berlin und München gezogen, während auf einer höheren Ebene die Parallelität der Maschinen genutzt wird. Dies reduziert die Datenmenge, die zwischen den Computern ausgetauscht werden muss, beträchtlich. Trotzdem ist für eine effiziente Nutzung von zwei Systemen das Vorhandensein der Gigabitleitung entscheidend, um die nötige Bandbreite zu erreichen.

3.1.2 Potentialflächen von RNA-Fragmenten

Anwendung des Konrad-Zuse-Zentrums für Informationstechnik, Berlin, Dr. T. Steinke

Zum Verständnis der spezifischen Wechelwirkung von RNA mit anderen Molekülen sind energetische und strukturelle Informationen relevanter Ausschnitte der Potentialhyperfläche von RNA-Fragmenten notwendig. Verlässliche Daten zur Energetik verschiedener Konformationen molekularer Systeme sind nur mit nichtempirischen quantenchemischen Methoden erhältlich. Dichtefunktionalmethoden sind in der Lage, kosteneffektiv auch bei sehr großen Molekülen hinreichend brauchbare energetische und strukturelle Informationen zu liefern. Mit dem quantenchemischen Programm GAMESS-UK (Version 6.0 für CRAY T3E) sollen unter Anwendung der implementierten Dichtefunktionale Ausschnitte der Potentialfläche von RNA-Fragmenten berechnet und stationäre Punkte durch das IR-Frequenzspektrum charakterisiert werden. Bedingt durch die Problemgröße (einige Hundert Atome) sind derartige Rechnungen aufgrund von Speicherengpässen noch nicht routinemäßig durchführbar.
Mit diesem Projekt wurde der sogenannte "heroische" Ansatz verfolgt, d. h., dass die geplante Weiterentwicklung von GAMESS-UK weniger auf die Ausnutzung einer etwaigen grob-körnigen Parallelität beruhte, sondern es sollte versucht werden, die ursprüngliche Funktionalität des Programmes, insbesondere die Nutzung des parallelen Eigensolvers mit den Global Arrays, an die Metacomputing-Umgebung anzupassen bzw. weiterzuentwickeln.

3.1.3 Hydrodynamik der Verbrennung in Supernovae und in Motoren

Anwendung des Max-Planck-Instituts für Astrophysik, Garching, Abt. Prof. Hillebrandt

Es wird der Code PROMETHEUS eingesetzt, der ursprünglich zur Simulation von Supernova-Explosions-Ereignissen entwickelt wurde, aber inzwischen auch zur Simulation von Verbrennungsvorgängen in Verbrennungsmotoren mit Relevanz für die Autoindustrie eingesetzt werden soll.
Der Code enthält einen Hydrodynamikteil, der auf dem PPM (Piecewise Parabolic Method) Verfahren basiert und der Simulation der Gasdynamik Rechnung trägt. Thermodynamische Eigenschaften der Gase werden mittels einer Zustandsgleichungsroutine berücksichtigt. Ferner ist ein Reaktionsnetzwerk integriert, das derzeit thermonukleare Reaktionen zwischen den Fluidkomponenten beschreibt, das man aber *straight-forward* für chemische Reaktionen adaptieren kann.
Bei der aktuellen Projektdurchführung von Nova-Simulationen auf 512 T3E PEs werden pro Zeitschritt etwa alle 10 s ca. 320 MB Daten zwischen allen Prozessoren ausgetauscht. Alle 50 Zeitschritte wird ein Zwischenergebnis von ca. 1 GB auf Disk geschrieben. Insgesamt sind pro Simulation etwa 200000 Zeitschritte abzuarbeiten. Damit stellt PROMETHEUS hohe Anforderungen an die Kommunikationsleistung.

3.1.4 Polymerforschung

Anwendung des Max-Planck-Instituts für Polymerforschung, Mainz, Abt. Prof. Kremer

Die Arbeitsgruppe "Theorie der Polymersysteme" am Max-Planck-Institut für Polymerforschung, Mainz, untersucht schwerpunktmäßig komplexe molekulare Systeme mit statistisch-mechanischen Methoden der Computersimulation. Die Berechnung von strukturellen und dynamischen Eigenschaften von Polymersystemen erfordert große Rechenressourcen. Für das Studium sehr langer Polymerketten sind Standardtechniken wie Molekulardynamik-Simulationen wegen des immensen Rechenaufwandes überfordert. Aus diesem Grund sind theoretische Ansätze zur Strukturvorhersage auf qualitativer Ebene mit geringerem Rechenaufwand entwickelt worden. Eine solche Theorie ist das *Polymer Reference Interaction Site Model* (PRISM). In diesem Modell werden gut parallelisierbare Monte Carlo Simulationen mit gut vektorisierbaren Näherungen zur molekularen Flüssigkeitsverteilung kombiniert. Eine Teilaufgabe könnte effizient auf einem skalaren MPP-System, die andere Teilaufgabe effizient auf einem Vektorsystem gelöst werden. Damit können Untersuchungen im Rahmen des *Polymer Reference Interaction Site Model* von der engen Kopplung eines Vektorrrechners und eines massiv parallelen Rechners über ein leistungsstarkes Netzwerk profitieren.
Die in den letzten Jahren entwickelte PRISM Theorie für Polymere gestattet es, allein aus der molekularen Struktur von Polymeren deren Packungseigenschaften zu berechnen. Die molekulare Struktur kann aus Monte-Carlo(MC)-Simulationen von einzelnen Polymeren berechnet werden. Für thermodynamische Eigenschaften reicht die Genauigkeit von PRISM nahezu an die von wesentlich aufwendigeren MD-Simulationen heran. Typischerweise sind PRISM Rechnungen für Polymere etwa 2 Zehnerpotenzen schneller als MD und sind daher eine attraktive Alternative für Fragestellungen von qualitativer Natur. Solche Berechnungen teilen sich natürlicherweise auf in einen leicht zu parallelisierenden, aber nicht vektorisierenden MC-Anteil und einen leicht vektorisierenden, aber nur schwierig zu parallelisierenden Anteil zur Lösung der PRISM Gleichungen, weshalb auch hier ein Metacomputing Ansatz vielversprechend erschien. Ein bestehender serieller Programmcode wurde hierfür erfolgreich parallelisiert, im PRISM-Teil vektorisiert, und beide Teile können mittels TCP Nachrichten auf verschiedenen Rechnerarchitekturen verteilt abgearbeitet werden. Auf geeigneten Rechnerplattformen lassen sich somit viele Anwendungsfälle mit dem im Rahmen dieses Projektes modifizierten Programm interaktiv bearbeiten.

3.1.5 Software zur Kopplung von Parallelrechnern

Entwicklung durch Rechenzentrum der Universität Stuttgart

Parallelisierung mit Message Passing basiert im allgemeinen auf dem Standard MPI (Message Passing Interface), der auf allen derzeit verfügbaren Parallelrechnern effizient implementiert wurde und der daher Portabilität und Performance garantiert. Die herstellerspezifischen Implementierungen von MPI bieten jedoch nicht die Möglichkeit, mehr als einen parallelen Rechner zu benutzen. Ansätze mehrere Rechner auf der Basis von MPI zu verbinden, greifen meist auf PVM oder Sockets zurück und erfordern daher Veränderungen im Code.

Deshalb wurde vom Rechenzentrum der Universität Stuttgart (RUS) die PACX-MPI Bibliothek entwickelt, mit deren Hilfe interoperables MPI realisiert werden kann (PACX steht für: PArallel Computer eXtension).

Aufbauend auf einem Mehrschichtenmodell wurde der MPI 1.2 Standard implementiert, so dass mehrere Rechner als ein einziger Rechner für eine Anwendung verfügbar gemacht werden können. Bis zur Implementierung interoperablen MPIs durch Hersteller ist PACX-MPI somit derzeit die beste Methode, MPI Programme unverändert in einem Meta-Computingszenario einzusetzen.

Obwohl PACX-MPI auf TCP/IP basiert und eine zusätzliche Schicht zwischen Anwendung und MPI darstellt, wird die Latenzzeit innerhalb der Maschine nur von 13 Microsekunden auf etwa 16 erhöht. Die Latenzzeit zwischen zwei Cray T3Es, die TCP/IP verbunden sind, beträgt 4.5 Millisekunden.

3.2 Homogenes Metacomputing

Für sogenannte homogene Metacomputingstudien wurden gleichartige Cray T3E-Systeme am ZIB in Berlin und am RZG in Garching eingesetzt und mittels GTB verbunden. Nach entsprechenden Code-Anpassungen wurden dann die Codes CPMD für ab-initio Molekulardynamikrechnungen, PROMETHEUS für astrophysikalische Berechnungen und GAMESS UK für RNA-Potentialflächenberechnungen für verteilte Rechnungen über das GTB eingesetzt. In allen Fällen wurde die Kopplungs-Software PACX-MPI verwendet.

3.2.1 Kopplungs-Software PACX-MPI (RUS)

Die PACX-MPI-Bibliothek wurde auf den T3Es am RZG und am ZIB installiert und die Grundfunktionalität überprüft. Die PACX-Entwickler am RUS erhielten auch frühzeitig Zugang zum VPP-System am LRZ, um mit der Portierung auf das Vektorsystem zu beginnen.

Da bei den VPP-Systemen jeder Knoten eine eigene IP-Adresse besitzt, und die Zuordnung auf die Knoten erst zur Laufzeit bekannt ist, mußte der Startmechanismus für die Metacomputing-Anwendungen neu erstellt werden. Im Gegensatz dazu ist auf der Cray T3E jeder Knoten unter derselben IP-Adresse ansprechbar. Aus diesem Grund war PACX-MPI für die T3E früher verfügbar. Die Fujitsu VPP wurde erst ab PACX-MPI Version 3.7.4 incl. neuem Startmechanismus unterstützt.

Im der ersten Projektphase wurden die nötigen Erweiterungen vorgenommen, um den komplexen Funktionsumfang der bereits eingesetzten Codes PROMETHEUS, CPMD und GAMESS UK zu unterstützen.
Neben der Beseitigung von kleineren Fehlern, die beim Einsatz von PACX-MPI mit den Applikationen auftraten und der Unterstützung der Anwender, stand die Verbesserung der Performance im Vordergrund. Das GTS bot hier die Gelegenheit unter reproduzierbaren Bedingungen zu testen. Die verfügbare hohe Bandbreite bei relativ hoher Latenz stellte dabei neue Anforderungen an PACX-MPI.

Die Optimierungen wurden in Zusammenarbeit mit Netzwerkexperten und Anwendungsprogrammierern (GAMESS-UK, Detlef Reichardt) des ZIBs durchgeführt. Um einen direkten Vergleich mit den verschiedener Programme wie netperf und nettest einerseits und PACX-MPI andererseits zu ermöglichen, wurde ein entsprechendes Benchmark-Programm entwickelt. Es handelt sich dabei um ein MPI Programm, das dieselben Verkehrsmuster erzeugt wie nettest/netperf. Die Umsetzung von MPI nach TCP/IP wird dabei von PACX-MPI vorgenommen.
Erste Tests ergaben, daß PACX-MPI viel empfindlicher auf die beobachtete Asymmetrie und damit der Fehlkonfiguration reagierte als die TCP/IP Benchmarks. Die Resultate schwankten viel stärker. Eine mögliche Ursache ist die Verwendung von relativ großen TCP/IP Pakete durch PACX-MPI. Dies bedingt eine größere Empfindlichkeit gegenüber ATM Zellverlusten. Mangels fehlender Tools auf der ATM Ebene konnte dies allerdings nicht endgültig verifiziert werden. Nach Beseitigung der Asymmetrie zwischen ZIB und RZG wurden die Ergebnisse reproduzierbar.

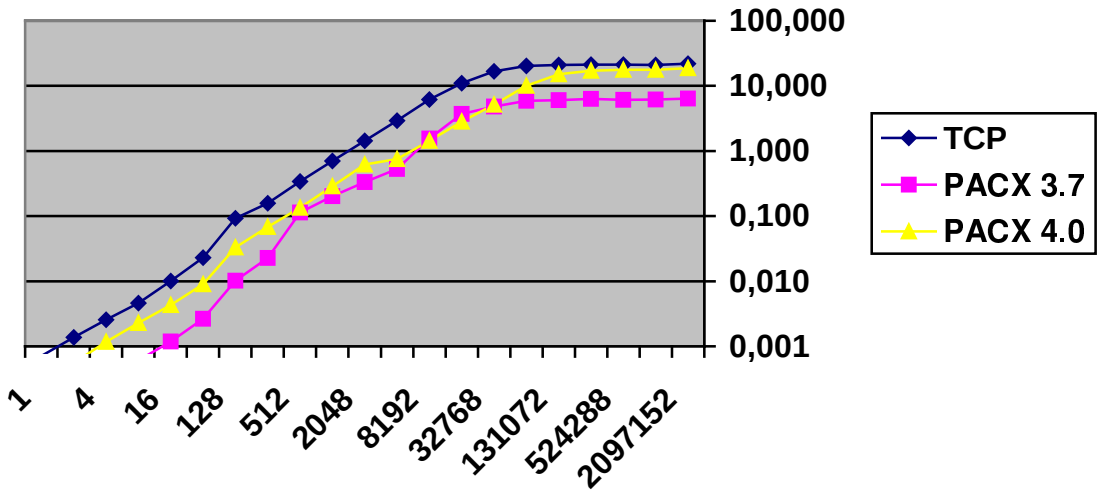

Abbildung PACX 1:
Leistungssteigerung von PACX-MPI um den Faktor 3 von Version 3.7 zu Version 4 Aufgetragen sind erzielte Übertragungsraten in MB/s in Abhängigkeit von der Datenpaketgröße in Byte

Die Abbildung zeigt die Ergebnisse der Verbesserungen. PACX-MPI 3.7 erreichte einen maximalen Durchsatz von 6,4 Mbyte/s (51 Mbit/s). Bei PACX-MPI 4.0 wurde der Wert auf 18,7 Mbyte/s (150Mbit/s) verbessert. Die Gründe für die schlechte Performance bei PACX-MPI 3.7 waren einerseits ein nicht optimaler Wert für das TCP Window Shifting, und andererseits ein ungewöhnliches Verhalten der Sockets auf der T3E bei der Übernahme der Konfiguration eines Sockets im Rahmen des „accept“-calls. Durch Vergrößern des TCP-Buffers konnte die Performance inzwischen auf 25 Mbyte/s (200 Mbit/s) gesteigert werden. Die Bandbreite, die für verteile MPI-Applikationen zur Verfügung steht, entspricht damit der TCP/IP-Bandbreite. Die Vergrößerung der Puffer war möglich geworden, da neuere Versionen des UNICOSmk Betriebssystems höhere Werte unterstützen.

In der weiteren Folge wurde die Zuverlässigkeit und Performance von PACX-MPI durch weitere Tests verbessert. Außerdem wurde die Implementierung des MPI Standards

vorangetrieben, so daß jetzt nahezu eine vollständige MPI 1.2 konforme Implementierung zur Verfügung steht.

Insgesamt wurde die PACX-MPI Bibliothek im Rahmen des Gigabit Testbed Süds deutlich verbessert. Dies betrifft die Bereiche Funktionsumfang, Zuverlässigkeit und Performance. Ermöglicht wurde dies einerseits durch die guten Testumgebungen im Gigabit Testbed Süd (schnelle Leitung und Reproduzierbarkeit) und andererseits durch die enge Zusammenarbeit mit den Entwicklern der verschiedenen Anwendungen.

3.2.2 GAMESS-UK (ZIB):

Die in der Originalversion von GAMESS-UK benutzte Message-Passing-Bibliothek TCGMSG war Ausgangspunkt der Implementierungsarbeiten. Unter Beibehaltung der Aufrufschnittstelle in GAMESS-UK wurde die im Programmpaket enthaltene MPI-Version (TCGMSG-MPI) verwendet und erweitert. In enger Zusammenarbeit mit den Entwicklern der PACX-Bibliothek (Arbeitsbesuch von E. Gabriel und M. Müller im März 2000 im ZIB) wurden Untersuchungen der Netzperformance durchgeführt, die zu einer wesentlichen Verbesserung der PACX-Performance bezüglich Bandbreite und Latenz beitrugen (siehe Abschnitt 3.2.1).

Die auf der Grundlage von TCGMSG-MPI erstellte MPI-Version von GAMESS-UK zeigt für mittlere Systemgrößen (ca. 400 Basisfunktionen) und mittlere Knotenzahlen (bis ca. 100) bei Dichtefunktionalrechnungen (DFT) ein gutes Skalierungsverhalten. Anzumerken ist, dass die MPI-Version nur eine algorithmisch eingeschränkte Funktionalität bietet: wenn die Lösung des Eigenwertproblems zeitlich dominiert, was bei sehr grossen molekularen Systemen mit deutlich mehr als 500 Basisfunktionen auftritt, kann nur eine begrenzte Skalierung erwartet werden. Umgangen kann diese Einschränkung durch Einsatz des parallelen Eigenwertlösers PeIGS, der jedoch eine Implementierung der Global Arrays (s.u.) voraussetzt.

Zur Erstellung der PACX-Version war zunächst die Implementierung WAN-verteilter globaler Taskzähler erforderlich, um eine dynamische Lastbalancierung in der Metacomputing-Umgebung zu ermöglichen. Globale Taskzähler werden in GAMESS-UK zum Lastausgleich beim verteilten Aufbau der Fock-Matrix und der Berechnung der entsprechenden Zweielektronen-Integrale eingesetzt.
Das folgende Beispiel von DFT-Rechnungen für Morphin mit 410 Basisfunktionen (s. Abb. G 1) für 32 bis 128 PEs auf der Garchinger CRAY T3E (300 MHz) im Vergleich zu verteilten Rechnungen zwischen Garching und Berlin (16+16 bis 64+64 Computing PEs) zeigt einen Leistungsabfall der PACX-Version bezüglich der (lokal laufenden) MPI-Version von nur 8%. Bei einer derartigen Rechnung werden typischerweise 300 MB an Daten über das Netz übertragen, wobei fast ausschließlich Broadcast-Operationen und globale Summen verwendet werden. Als weitere Gründe für den geringen Leistungsabfall lassen sich die verteilte Lastbalancierung und ein sparsamer Einsatz globaler Barrieren anführen.

Die Implementierungsarbeiten konzentrierten sich sodann auf die Erweiterung der Funktionalität der Schnittstelle zwischen TCGMSG- und PACX-Bibliothek (*libtcgmsg-pacx.a)*. Gegenstand waren:

- eine Berücksichtigung der Hardware-Umgebung bei der dynamischen Lastbalancierung zur Laufzeit (Anpassung der globalen Taskzähler), sowie
- die Bereitstellung einer Umgebung für einseitige asynchrone Kommunikation über das WAN, um die Funktionalität der Global Arrays (GA) innerhalb der Metacomputing-Umgebung zur Verfügung zu stellen.

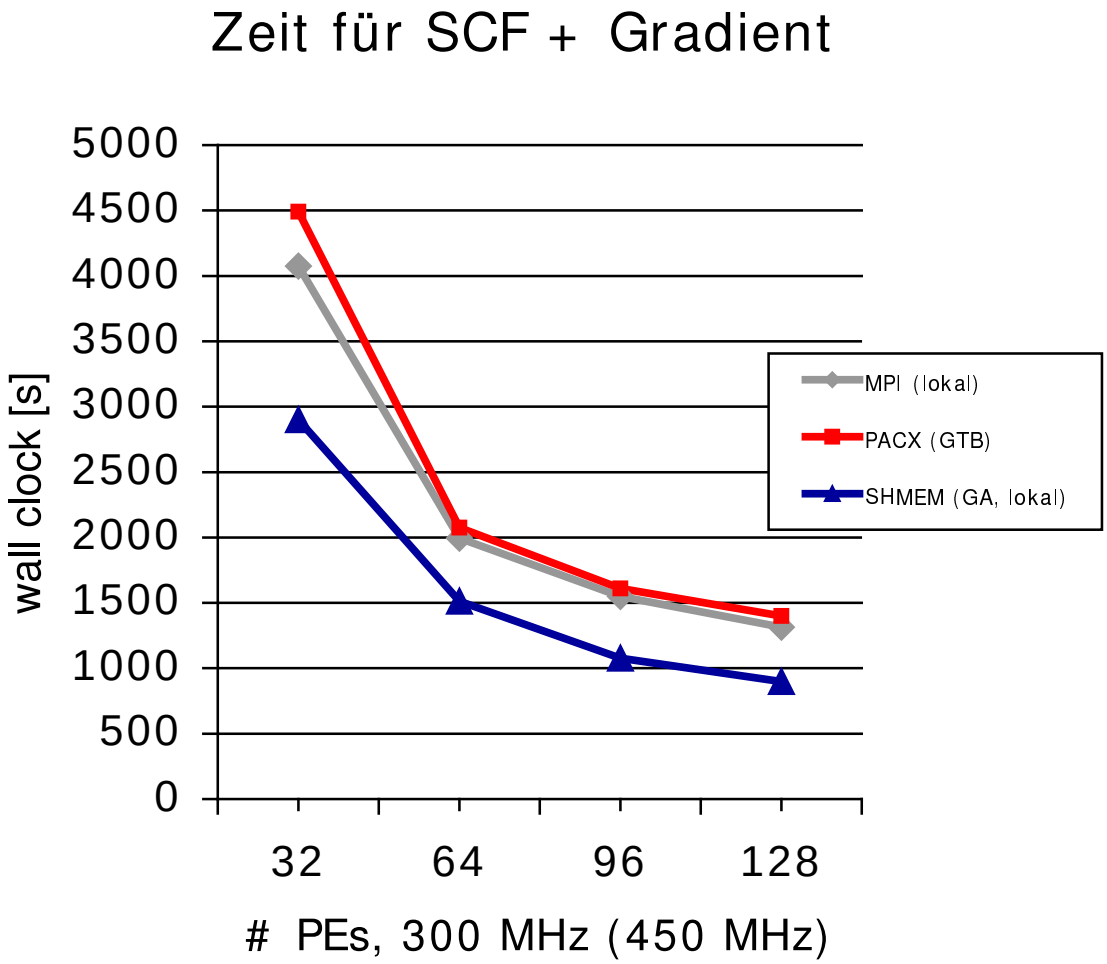

Abbildung G 1:
GAMESS-UK Programmlaufzeiten bei verteilter Verarbeitung auf T3Es am RZG und ZIB (PACX-MPI, GTB) und Abarbeitung auf einem einzigen T3E-System am RZG (MPI bzw. SHMEM mit GA)

Die Lastbalancierung in der homogenen Metacomputing-Umgebung sollte zur Laufzeit die heterogene Hardware-Konfigurationen der beteiligten Komponenten des virtuellen Metacomputers berücksichtigen. Neben den unterschiedlichen Taktfrequenzen der CPU-Boards der hier beteiligten CRAY T3E in Berlin und Garching können es auch unterschiedliche Partitionsgrössen (Anzahl der jeweils lokal benutzten Compute PES) sein. Eine Berücksichtigung derartiger Unterschiede wird durch Bereitstellung disjunkter Folgen von Taskzählern in einem vorgegebenen Mengenverhältnis ihrer Elemente erreicht, die je nach Anforderung feingranular erfolgen kann. Die Auswirkung der verbesserten Lastbalancierung wird am Beispiel des Testsystems Morphin demonstriert (s. Abb. G 2).

DFT-Rechnungen auf jeweils 2 Partitionen der T3E des ZIB bei konstanter Gesamtzahl der Computing PES, aber mit unterschiedlichem Verhältnis der PE-Anzahl pro Partition zeigen eine annähernde Konstanz der Gesamtlaufzeit.

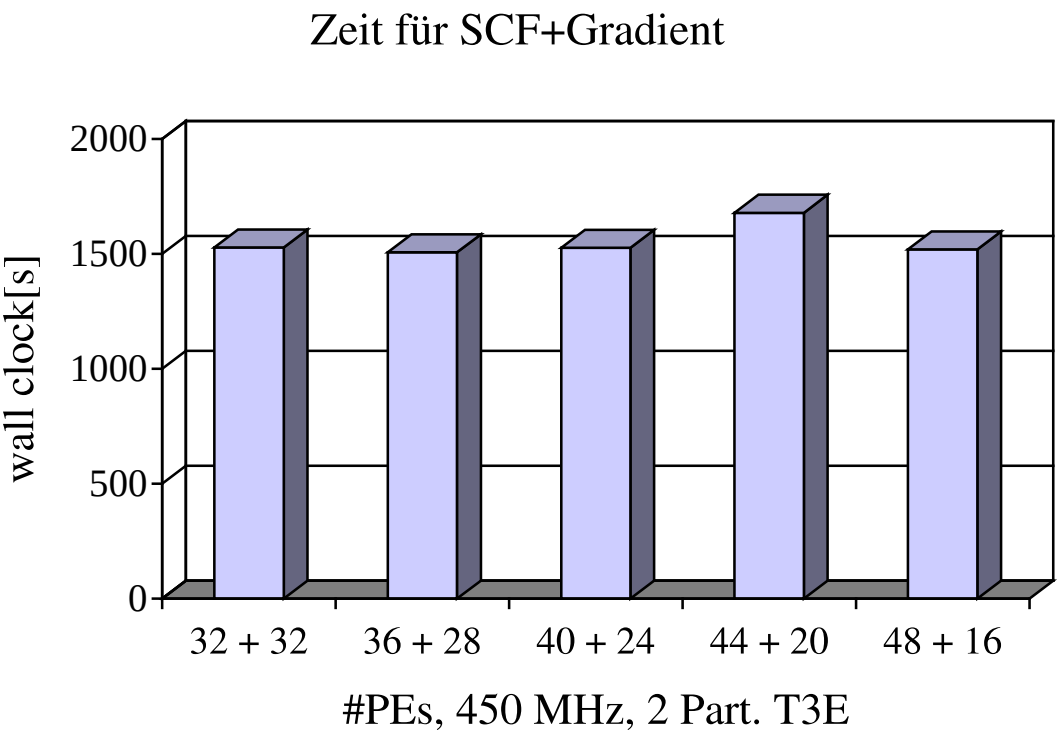

Abbildung G 2:
Abhängigkeit der GAMESS-UK Programmlaufzeiten für DFT-Rechnungen von Morphin bei Verarbeitung auf 2 Partitionen verschiedener Größe auf T3E am ZIB.

Wie bei der Diskussion der MPI-PACX-Version erwähnt, bietet erst die Einbeziehung des Global Array Toolkit (GA) in die GAMESS-UK-Implementierung die Voraussetzung, ein für hier interessierende Molekülgrössen adäquate Skalierung zu erreichen. Die Global Arrays bieten dem Programmierer eine portable Schnittstelle einer abstrakten NUMA-Architektur an und ermöglichen den Einsatz des parallelen Eigenwertlöser PeIGS sowie von verbesserten I/O-Strategien (z. B. Caching). Das verbessert das Skalierungsverhalten und die Ausführungszeit deutlich für Moleküle ab 500 Basisfunktionen (letzteres deutet sich schon beim kleineren Beispiel in Abb. G 1, SHMEM-Kurve, an).

Vorbereitend für den Einbau der GA mussten Erweiterungen an der TCGMSG-PACX-Schnittstelle sowie WAN-spezifische Anpassungen in den Global Arrays vorgenommen werden.
Die TCGMSG-PACX-Schnittstelle wurde um lokale Kommunikatoren erweitert, da hier eine transparente Verwendung von PACX nicht mehr möglich ist. Durch redundante Implementierung der GA auf den jeweils lokalen Partionen des Metacomputers und der optionalen Spiegelung der Daten für den Eigensolver soll der notwendige Datentransfer reduziert werden. Der damit verbundene Mehraufwand für die redundante Diagonalisierung

auf den lokalen Partitionen wird bei den angestrebten Problemgrößen von 1000-2000 Basisfunktionen als akzeptabel angesehen.

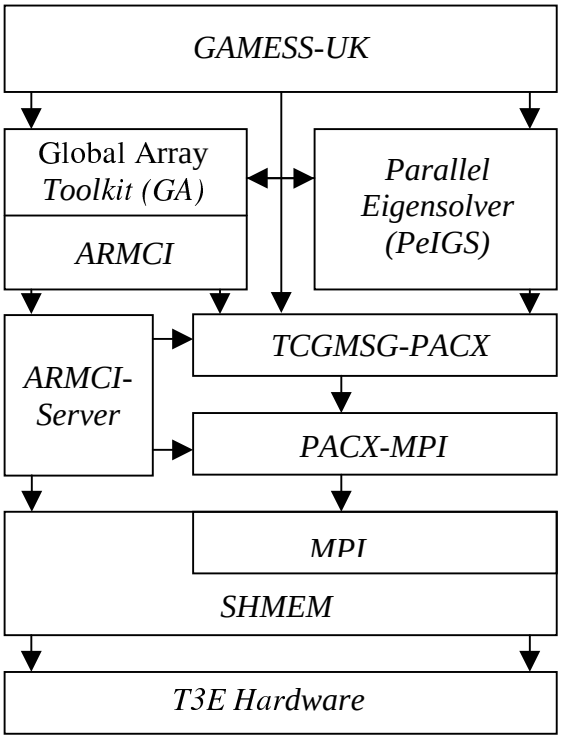

Abbildung G 3:
Parallelisierungsschema der Metacomputing-Version von GAMESS-UK unter Nutzung der Global Array Toolkit (GA).

Die Anpassung der Global Array an die Metacomputing-Umgebung unter Verwendung von PACX wurde mittels der aktuellen Version GA 3.0 vorgenommen. Diese enthält mit ARMCI (Aggregate Remote Memory Copy Interface) eine neue zusätzliche Softwareschicht zur Trennung von Datenverwaltung und Datenverteilung innerhalb der GA. ARMCI stellt als portable Schnittstelle zur Datenverteilung die Funktionalität der einseitigen asynchronen Kommunikation (*ARMCI_get*/*ARMCI_put*) zur Verfügung. Auf CRAY T3E nutzt ARMCI die entsprechenden Funktionen der SHMEM-Library (*shmem_get*/*shmem_put*).
Für eine Realisierung in einer Metacomputing-Umgebung wurde das Konzept des ARMCI-Servers entwickelt (s. Abb. G 3). Pro Partition wird auf jeweils einem Knoten ein Serverprozess gestartet, der die Aufgaben von *ARMCI_get* und *ARMCI_put* lokal übernimmt und die Kommunikation mit den Knoten der Remote-Partition über MPI-PACX, beispielsweise *MPI_Send* und *MPI_Recv*, abwickelt. Dieses vollständig neu zu implementierende Konzept erforderte eine komplette Reorganisation der TCGMSG-PACX-Bibliothek, da die Bereitstellung neuer Kommunikatoren notwendig war.

Die Tragfähigkeit des ARMCI-Server–Konzeptes wurde erfolgreich mittels einer Testumgebung für die ARMCI-Bibliothek in der Metacomputing-Umgebung zwischen Garching und Berlin nachgewiesen. Ein Einbau in die Metacomputing-Version von GAMESS-UK war jedoch aus Zeitgründen vor Ablauf des Projekts nicht mehr möglich.

Die bislang in der Metacomputing-Umgebung mit GAMESS-UK erreichten Leistungszahlen sowie das ARMCI-Server-Konzept wurden auf dem *3. Metacomputing-Workshop* des Höchstleistungsrechenzentrums Stuttgart (Juni 2000) vorgestellt.
(s. http://www.hlrs.de/news-events/events/2000/meta2000/metaGAMESS.ppt)

3.2.3 Code CPMD (MFK, RZG)

Der ab-initio Molekulardynamik-Code CPMD wurde auf den T3E-Systemen am RZG und am ZIB implementiert. Durch Einbeziehung der Kopplungssoftware PACX-MPI gelang es, erste verteilte Läufe zwischen den Cray-T3Es in ZIB (Berlin, 450 MHz Prozessoren) und RZG (Garching, 300 MHz Prozessoren) durchzuführen. Aus den unterschiedlichen Taktfrequenzen der Prozessoren wurden dabei keine Vorteile gezogen.
Zum Studium des Einflusses der verschiedenen beteiligten Komponenten wurde stufenweise vorgegangen. Folgende Hauptkomponenten sind beteiligt:

CPMD + internes MPI, PACX-MPI, TCP/IP, GTB.

Als Referenzlauf diente ein 16-Prozessorlauf auf dem T3E-System am RZG (CPMD + internes MPI, Simplexlauf). Bei Einsatz der Pfadintegralmethode kann eine Gruppenbildung von Prozessoren vorgenommen werden, und es wurden die drei Varianten von einer Gruppe mit 16 Prozessoren, zwei Gruppen mit 8 Prozessoren, und 4 Gruppen mit 4 Prozessoren untersucht.

Im nächsten Schritt wurde der Einfluss von PACX-MPI als weiterer Softwareschicht untersucht, unter Ausschluss zusätzlicher Elemente. Solche Simplexrechnungen mit PACX-MPI wurden durch Angabe eines Netzinterfaces auf der T3E ermöglicht, der TCP/IP Stack wurde hierbei nicht durchlaufen. Auch hier wurde der Einfluss der Gruppenbildung betrachtet.

Durch Einbeziehung zweier Netzinterfaces auf dem T3E-System (HIPPI-Interface zum GTB und HIPPI-Interface zum Archivsystem) konnte eine Duplexrechnung auf nur einem Rechner simuliert werden. Der TCP/IP-Overhead kam jetzt zusätzlich zum PACX-MPI Einfluss hinzu, ohne dass ein Netzwerk benutzt wurde und damit das GTB noch ohne Einfluss war. Auch hier wurde die Gruppenbildung vorgenommen.
Für diese Konstellation wurde nun der Einfluss der in der PACX-MPI-Bibliothek verwendeten minimalen Paketgrößen untersucht. Dabei handelt es sich um einen internen Protokollparameter von PACX-MPI. Die Erhöhung vom Default 4 kB auf 32 kB führte dabei zu einer deutlichen Verbesserung bei der Kommunikationsleistung.

Im letzten Schritt wurden dann echte Duplexrechnungen zwischen den T3Es in Garching und in Berlin über das GTB durchgeführt. Die Vergleichbarkeit der Messergebnisse mit den vorherigen Messungen war dadurch gegeben, dass die mit 450 MHz schneller getakteten T3E-Prozessoren am ZIB an den Synchronisationsstellen einfach auf die langsameren 300 MHz Prozessoren am RZG warteten, also auch nur die Leistung von 300 MHz Prozessoren erbrachten. Es wurde eine Rechnung mit zwei Gruppen zu je 8 Prozessoren und eine Rechnung mit 4 Gruppen zu je 4 Prozessoren durchgeführt. Für den PACX-Kontrollparameter "minimale Paketgröße" wurde die günstigere Einstellung von 32 kB gewählt.

Die Ergebnisse zu den beschriebenen Messungen sind in der Abb. C 1 dargestellt.

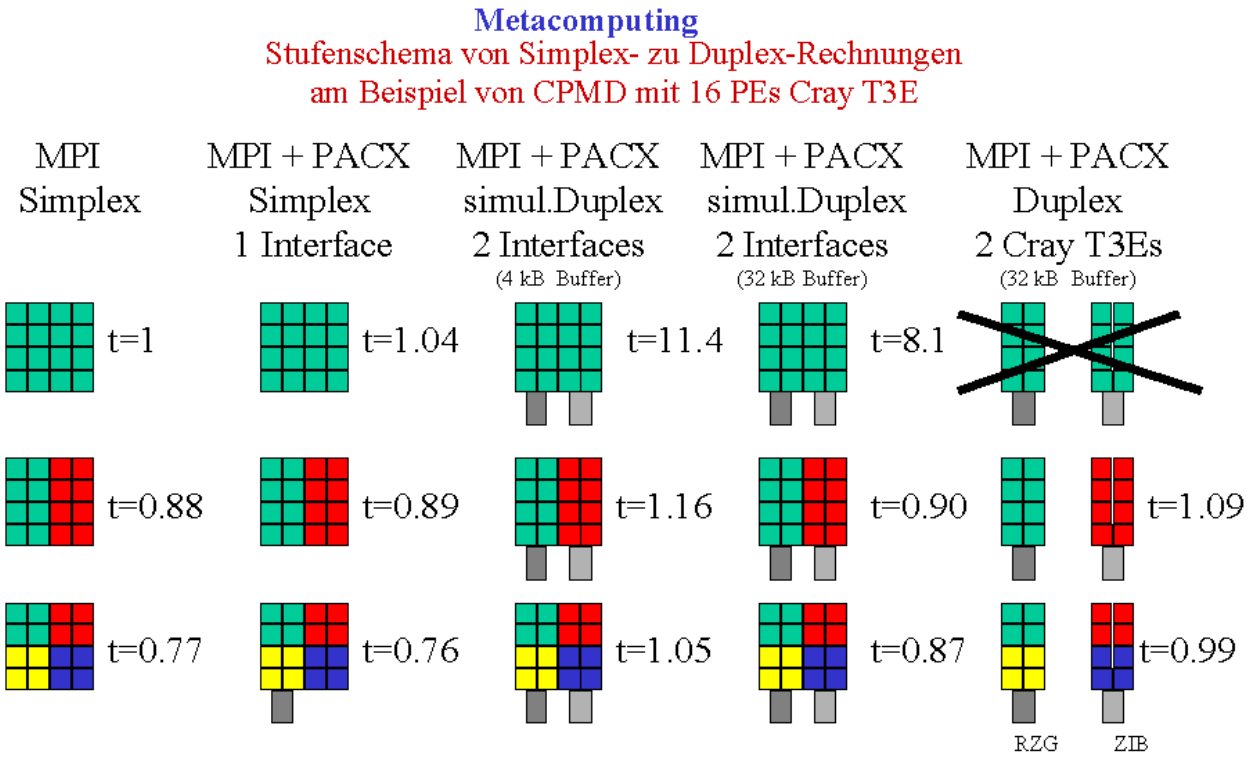

Abbildung C 1
Stufenschema von Simplex- zu Duplexrechnungen mit CPMD unter Einsatz von Gruppenbildung nach der Pfad-Integral-Methode
Quadrate gleicher Farbe (grün, rot, gelb, blau) repräsentieren Prozessoren derselben Prozessorgruppe. Graue Kästchen repräsentieren Netz-Interfaces.
Der Parameter t beschreibt die ermittelten Programmlaufzeiten relativ zum Simplex-Lauf auf einem Rechnersystem mit allen Prozessoren in derselben Prozessorgruppe (t = 1).

Bei Betrachtung der Messreihe mit nur einer Prozessorgruppe von 16 Prozessoren von links nach rechts (oberste Reihe in Abb. C 1), von der Simplex- bis zur vollen Duplexrechnung, sieht man auf einen Blick, dass Hinzufügen der PACX-Bibliothek nur einen minimalen Overhead von 4% erzeugt, der zusätzliche TCP/IP-Overhead in der simulierten Duplexrechnung (mit 4 kB PACX Buffer Size) aber zu einer gewaltigen Programmverlangsamung um mehr als den Faktor 11 führt. Dieser Overhead kann zwar durch eine PACX Buffer Size von 32 kB auf den Faktor 8 abgemildert werden, es liegt aber immer noch eine Laufzeitverschlechterung um nahezu eine Größenordnung vor.

Wesentlich günstigere Verhältnisse ergeben sich bei einer Gruppenbildung nach der Pfadintegralmethode (mittlere und untere Reihe in Abb. C 1).

Durch Bildung von 2 Gruppen zu je acht Prozessoren erfolgt bereits eine Verbesserung um 12% auch beim Simplexlauf. Der PACX-Overhead wird ganz vernachlässigbar. Und der TCP/IP Overhead wird deutlich reduziert. Bei 32 kB PACX Buffer Size liegt die entsprechende Laufzeit mit t = 0,90 nur noch um ca. 2% höher als beim entsprechenden Simplexlauf mit t = 0,88. Der echte Duplexlauf über das GTB (t = 1,09), in dem zusätzlich das GTB "bremsend" wirkte, erfuhr nur eine Verlangsamung um 24% relativ zum entsprechenden Simplexlauf mit t = 0,88.

Noch günstigere Verhältnisse ergaben sich bei der Gruppenbildung von 4 Gruppen mit je 4 Prozessoren (unterste Reihe in Abb. C 1). Auch hier profitiert man bereits beim Simplexlauf von der Gruppenbildung (t = 0,77). Die Verlangsamung beim echten Duplexlauf über das GTB betrug mit t = 0,99 zwar 29% relativ zum entsprechenden Simplexlauf, insgesamt hat sich aber die verteilte Abarbeitung über das GTB von t = 1,09 bei nur 2 Gruppen noch einmal auf t = 0,99 bei Nutzung von 4 Gruppen verbessert.

Dies zeigt, dass der Einsatz der Pfad-Integral-Methode für verteilte Rechnungen gute strategische Vorteile bietet.

Im weiteren Verlauf der Studie wurde nun die Zahl der eingesetzten Prozessoren erhöht.
Während der dedizierten Messzeiten, zu denen RZG und ZIB größere Prozessorgruppen für simultane Nutzung im Rahmen des Metacomputing-Projekts reservierten, wurden Rechnungen bis zu je 64 Prozessoren auf beiden T3Es durchgeführt ("2x64"). Die gemessenen Zeiten werden relativ zu einem 128-Prozessorlauf auf dem T3E-System am RZG angegeben.

Als Beispiel wurde eine Zelle mit 31 Wassermolekülen und einer Hydroxylgruppe verwendet. Die sog. Trotter-Dimension wurde auf 8 gelegt, so dass maximale Prozessorgruppengrößen ermöglicht wurden.

Ein Referenzlauf auf 128 Prozessoren des T3E-Systems am RZG benötigte 903 Sekunden.
Bei einer Bildung von 4 Prozessorgruppen zu je 16 Prozessoren auf jedem der beiden T3Es in Berlin und Garching, wobei insgesamt ebenfalls 128 Prozessoren eingesetzt wurden, dauerte dieselbe Rechnung über das GTB 1092 Sekunden, was einen Verlust von nur 21 Prozent durch die Verteilung auf die beiden entfernten Rechner bedeutet.
Durch eine Optimierung der Socket-Parameter konnte die erzielte Effizienz weiter gesteigert werden, so dass die Verluste nur noch bei 17 Prozent lagen.
In einer Vergleichsmessung mit der Bildung von nur zwei statt vier Prozessorgruppen je Rechner erhöhten sich die Verluste bereits auf 57%.

Wie bei dem kleineren Testbeispiel ist es auch bei größeren Läufen vorteilhaft, möglichst große Prozessorgruppen zu bilden, um so die Kommunikation zwischen den schwach gekoppelten Pfadintegralgruppen zu minimieren. Diese Beobachtung wurde durch eine weitere Berechnung des 31*Wasser+OH-Beispiels mit einer kleineren Prozessorzahl von insgesamt 2 mal 32 Prozessoren bestätigt.

Da es wichtige wissenschaftliche Problemstellungen gibt, die effizient mit der Pfadintegralmethode bearbeitet werden könne, kann zusammenfassend gesagt werden, dass der ab-initio Molekulardynamik-Code CPMD in der Pfadintegral-Variante sehr gut für

verteiltes Rechnen mittels Metacomputing in einer homogenen Rechnerumgebung eingesetzt werden kann.

Literatur:

D. Marx u. J. Hutter: *Ab initio molecular dynamics: Theory and Implementation*, in: Modern Methods and Algorithms of Quantum Chemistry, Proceedings Series Volume 1 (Ed. J. Grotendorst), Jülich 2000, ISBN 3-00-005618-1
(http://www.fz-juelich.de/nic-series/Volume1/)

3.2.4 Code PROMETHEUS (MPA)

Für dieses Teilprojekt wurde der Code PROMETHEUS eingesetzt. Dies ist ein Code, der zur Simulation von Brennfronten in Supernovae eingesetzt wird.

3.2.4.1 Beschreibung und Implementierung des Codes

Es werden auf einem Rechengitter die hydrodynamischen Gleichungen mit der sogenannten "piecewise parabolic method" für mehrere Gaskomponenten gelöst. Reaktionen zwischen den Komponenten werden mit Hilfe eines Reaktionsnetzwerks und einem Newton-Raphson Solver berechnet. Die thermodynamischen Eigenschaften des Gases werden mit einer Zustandsgleichung modelliert.

Der Code wurde so parallelisiert, dass das gesamte Rechengitter in einzelne unabhängige Blöcke zerlegt wurde. Jedem PE wird ein Block zugeordnet (domain decomposition). Zur Berechnung eines Zeitschritts muss Information zwischen benachbarten Blöcken ausgetauscht werden. Das wird im Laufe dieses Berichts noch genauer spezifiziert werden.

3.2.4.2 Einbettung von PROMETHEUS in das Meta-Computing Projekt:

Von der Art der Parallelisierung und vom Aufbau der Datenstruktur im Code PROMETHEUS sowie der Forderung nach maximaler load balance eignet sich PROMETHEUS für Tests im Bereich des homogenen Metacomputings.
Mit einer Verteilung der Anwendung auf zwei CRAY T3E-Systeme am RZG, Garching und am ZIB, Berlin. sollten Duplexrechnungen ausgeführt und mit Simplexrechnungen verglichen werden.
Die notwendigen Vorarbeiten konnten bis September 1999 durchgeführt, getestet und abgeschlossen worden. Sie bestanden im Wesentlichen aus.

- Der Vervollständigung der Konversion von CRAY-shmem zu Standard-MPI Kommunikation
- Verteilte Implementierung der neuen Code-Version auf den T3Es in Garching und Berlin.
- Einbindung und Test der PACX-MPI-Bibliothek zum verteilten Rechnen auf mehreren Maschinen vom Rechenzentrum der Uni Stuttgart.
- Implementierung fehlender MPI-Aufrufe in PACX-MPI durch RUS

- Definieren, Aufsetzen und Testen von Testproblemen für das Meta-Computing.

3.2.4.3 Implementierung der Kommunikation in PROMETHEUS und daraus resultierende Herausforderungen für die GTB-Leitung:

Der Hydrodynamikteil von PROMETHEUS löst die hydrodynamischen Gleichungen mit dem ppm (piecewise parabolic method) Verfahren. Die hydrodynamischen Größen werden in den Zonen durch Parabelstücke approximiert. Die dadurch entstehenden Diskontinuitäten an den Zonenrändern sind als Riemannprobleme bekannt und werden mittels eines Riemannsolvers gelöst. Dieser Riemannsolver ist eindimensional und wird nach dem "directional splitting" Verfahren in jeder der drei Raumrichtungen getrennt angewendet. Die Anwendung des Riemannsolvers in einer Raumrichtung wird als "Sweep" bezeichnet.
Für einen Hydrodynamik-Zeitschritt in einem dreidimensionalen Rechengebiet werden demnach drei Sweeps benötigt. Die Kommunikation zwischen den benachbarten PEs erfolgt nach j e d e m Sweep. Das heißt, innerhalb eines Hydrozeitschritts kommunizieren die benachbarten PEs d r e i m a l.

Zur Erläuterung der Kommunikation muss an dieser Stelle die Behandlung von Randbedingungen in PROMETHEUS betrachtet werden. PROMETHEUS behandelt Randwertprobleme in der Weise, dass an Gitterenden vier zusätzliche sogenannte "Geisterzellen" angehängt werden. In ihnen werden keine Gleichungen gelöst, sondern sie werden lediglich mit Werten gefüllt, um eine entsprechende Randbedingung zu simulieren (z.B. reflektierend, durchlässig, periodisch,...)

In der Parallelversion von PROMETHEUS werden die Geisterzellen eines Blocks, der einem PE zugeordnet ist, dazu benutzt, Information von benachbarten Blöcken ("benachbarten" PEs) vor jedem Sweep zu acquirieren. Das gilt für jeden PE. Folglich muss vor jedem Sweep jeder PE mit seinen 6 direkten Nachbarn bidirektional kommunizieren. Dabei entstehen unabhängige Randwertprobleme in den einzelnen Blöcken, deren Randbedingungen einfach durch den Zustand der Ränder der benachbarten Blöcke definiert ist.

Technisch geschieht das so, dass an jedem der 6 Seiten der eigentlich gerechneten Blöcke ein Geisterzellenblock von der Fläche der jeweiligen Seite und der Höhe 4 (4 Geisterzellen sind zur Festlegung von Randbedingungen in PROMETHEUS notwendig) mit gespeichert werden. Das wird in folgender Abbildung verdeutlicht.

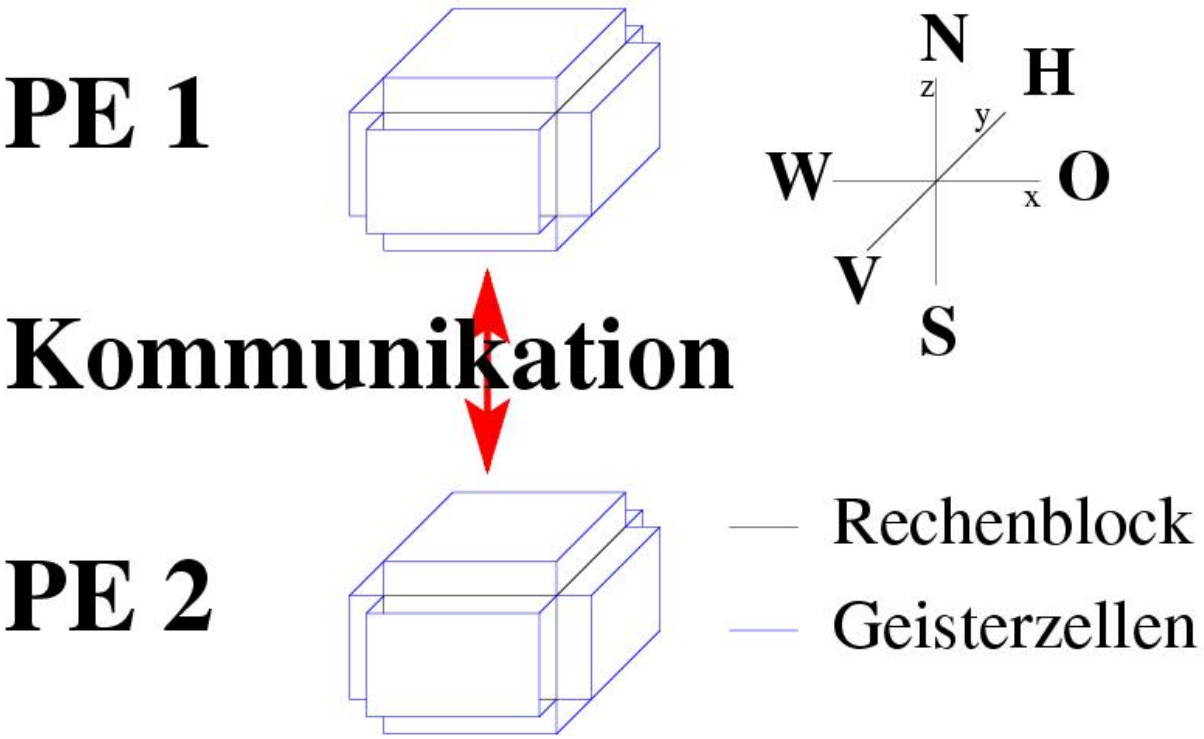

Abbildung P 1
Kommunikation zum Austausch der Geisterzellen

Pro Kommunikationvorgang in einer Richtung werden damit

n1 * n2 * 4 * 8 * qn

Bytes verschickt. Dabei sind n1 und n2 die Anzahl der Zellen auf der jeweiligen Blockoberfläche und 4 die Höhe eines der "Geisterzellenblöcke". Die Zahl 8 bedeutet Real*8 Repräsentation für ein CRAY-Wort und qn ist die Anzahl der zu kommunizierenden Variablen.

Diese Kommunikation wird im Code von einer Routine namens mpi_com.f ausgeführt.
Das Ablaufschema soll hier kurz zusammengefasst werden. Ein PE sammelt die zu verschickenden Daten in einem Puffer und schickt diese per mpi_send an das entsprechende Nachbar-PE, das seinerseits die Daten mit mpi_recv empfängt und den Inhalt des Puffers wieder in die richtigen Variablen schreibt. Dieser Vorgang wird von jedem PE in allen sechs Richtungen und bidirektional durchgeführt.

Worin besteht die Herausforderung für eine Gigabit Leitung wie der zwischen dem ZIB und dem RZG?

Liegt eine Anwendung von genügend hoher Zahl an Zellen (n_x*n_y*n_z) und eine genügende Anzahl von Prozessoren (N) vor und sind die Prozessoren zu gleichen Teilen auf zwei Maschinen verteilt, so ergibt sich folgende Situation für die Verbindung der beiden Rechner:

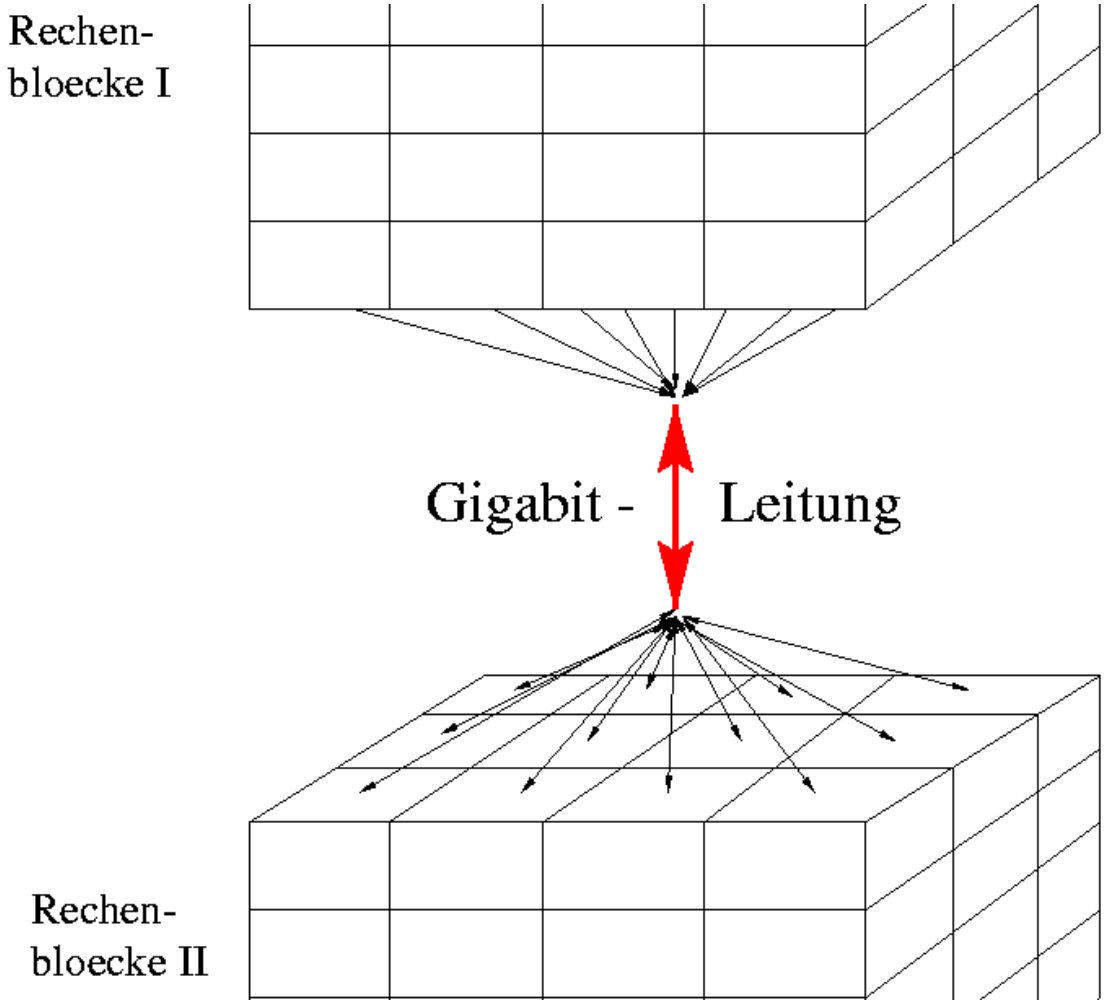

Abbildung P 2
Einbettung der Gigabit-Leitung bei Verteilung der Rechenblöcke

Das Rechengitter teilt sich in zwei gleich große Gebiete auf den jeweiligen Maschinen. Im günstigsten Fall sind die beiden Gebiete nur durch eine Grenzwand voneinander getrennt. In ungünstigeren Faellen vergrößert sich die "Kommunikationsoberfläche", die einfach durch die Anzahl der Seitenflächen definiert ist, die über die Gigabit-Leitung verbunden sind. Jede andere Situation ist ungünstiger. Wir wollen uns in diesem Bericht nur auf die günstigste Realisierung beschränken.
Um einen Kommunikationsschritt durchzuführen, müssen durch die Gigabit Leitung np_x * np_y Pakete sowohl in eine Richtung, als auch in die andere Richtung verschickt werden. Dabei ist np_x die Anzahl der Blöcke (PEs) in x-Richtung und np_y das Entsprechende für die y-Richtung (Es ist eine Teilung senkrecht zur z-Achse angenommen. Die Größe eines solchen Pakets ist nun, wie oben erwähnt,

n1 * n2 * 4 * 8 * qn

Die Herausforderung für die Gigabit Leitung liegt damit auf der Hand. Je nach Größe des Grids, der Anzahl der zu kommunizierenden Variablen und der Anzahl der PEs muss sie mehr oder weniger umfangreiche Daten in mehr oder weniger zahlreichen Paketen verteilen. Diese Tatsache unterscheidet "günstige" Probleme von "ungeeigneten" zur verteilten Rechnung. Bei "günstigen" Problemen übertrifft die Rechenzeit auf den einzelnen PEs die Kommunikationszeit. Das wird entweder dadurch realisiert, dass man kompliziertere lokale Physik (Zustandsgleichung, Kernreaktionsnetzwerk,...) einbaut oder bei gleicher Anzahl an PEs die Anzahl der Gitterzellen erhöht. Es bleibt für jeden Fall zu prüfen, ob eine Anwendung in diesem Sinne günstig ist, oder ob sie sinnvoll günstig gemacht werden kann.

Zusammenfassend gilt für den Kommunikationsbedarf bei der Durchführung eines Hydro-Zeitschritts:

Angenommen, ein dreidimensionales Rechengebiet teilt sich auf np_x * np_y * np_z Blöcke (PEs) auf (np_x: Anzahl der Blöcke in x-Richtung,...). Die Grenzwand der beiden Teilgebiete auf den beiden Maschinen ist senkrecht zur z-Achse. Die Größe eines Blocks sei n1*n2*n3 Zellen (n1: Anzahl der Zellen eines Blocks in x-Richtung,...).
Pro Zeitschritt sind 3 Sweeps und damit 3 Kommunikationsschritte notwendig.
Pro Kommunikationsschritt werden np_x * np_y Pakete a n1 * n2 * 4 * 8 * qn Bytes bidirektional verschickt, also 2 * np_x * np_y Pakete pro Kommunikationsschritt.

Für einen Hydrozeitschritt werden demnach

3 * 2 * np_x * np_y Pakete a n1 * n2 * 4 * 8 * qn Bytes verschickt.

3.2.4.4 Testrechnungen

3.2.4.4.1 Simulation einer thermonuklearen Brennfront in einem Gas aus zwei Komponenten (Reaktionsedukt und -produkt).

Das 3D-Problem besteht aus 50 x 50 x 100 Gitterzellen, einer einfachen Gamma-Zustandsgleichung und einer einfachen analytischen Lösung für die Kernreaktionen. Gerechnet wurde dies auf 128 PEs. Das ergibt eine Aufteilung in 4 x 4 x 8 (np_x*np_y*np_z) Blöcke, a 12 x 12 x 12 (max 13 x 13 x 13) Gitterzellen (n1*n2*n3).

Die Rechnung wurde sowohl lokal auf der Garchinger T3E durchgeführt, als auch verteilt am RZG, Garching und am ZIB, Berlin. Für die verteilte Rechnung befanden sich ein Teilgebiet mit 4 x 4 x 4 Blöcken auf 64 PEs der T3E am RZG und ein gleich großes Rechengebiet mit gleicher Prozessorenzahl auf Seiten der T3E am ZIB.
Die Grenzwand zwischen den beiden Teilgebieten, die über die Gigabit Leitung verbunden sind, besteht aus den äußeren Seiten von jeweils 4 x 4 Blöcken auf beiden Seiten.

Es werden pro Hydro-Zeitschritt also 3*2*np_x*np_y = 3*2*4*4 Pakete a

4 * (n1 * n2) * (8 + qn) * 8 = 4 * (12 * 12) * 10 * 8 = 46 kB

Das sind 96 Pakete a 46kB in einem Hydro-Zeitschritt.

Das ergibt für diesen Fall eine kommunizierte Datenmenge pro Zeitschritt von unter 4.4 MB. Würde man das als ein Paket durch die Leitung schicken, bei einer angenommenen Transferrate von 200 Mbits/sec, dann dauert das 180 ms. Da aber 96 Versendungen organisiert werden müssen, ist 180ms eher eine zeitliche Untergrenze.
Der Vergleich der verteilten mit der lokalen Rechnung ergab für diesen Fall:

Verteilte Rechnung 64+64 PEs, 10 Zeitschritte.

Kommunikationsteil:

Gesamtzeit für Komm.:	42.1 s
Durchschn. Zeit für Komm./Zeitschritt:	4.21 s
Durchschn. Versendezeit für ein Paket:	44 ms

Gesamtzeit für diesen Lauf:

timing information

initialization	9.397 seconds
timestepping total	50.116 seconds
time for hydrodynamics	46.756 seconds (93.30%)
time for nuclear reactions	2.397 seconds (4.78%)

Lokale Rechnung mit MPI, 128 PEs, 50 Zeitschritte.

Kommunikationsteil:

Gesamtzeit für Komm. :	5.73 s
Durchschn. Zeit für Komm./Zeitschritt:	115 ms
Durchschn. Versendezeit für ein Paket:	1.2 ms

Gesamtzeit für diesen Lauf:

timing information

initialization	6.648 seconds
timestepping total	20.623 seconds
time for hydrodynamics	19.981 seconds (96.89%)
time for nuclear reactions	0.641 seconds (3.11%)

In diesem Fall handelt es sich um eine ungünstige Anwendung, da die Zeit für die Kommunikation bei der verteilten Rechnung etwa 90% der Zeit zur Durchführung des Zeitschritts benötigt.
Im Vergleich hierzu benötigt die Kommunikation bei der lokalen Rechnung auf der T3E in Garching nur etwa 29% der Zeit eines Zeitschritts. Die Kommunikationszeit für einen Zeitschritt verhält sich demnach wie folgt für die beiden Fälle:

(Kommunikationszeit verteilte Rechnung) : (Kommunikationszeit lokale Rechnung) = 37

Interessant ist, dass die Versendung eines Pakets (inkl. Signallaufzeiten) im Schnitt 44 ms dauert, obwohl nur etwa 46 kB versendet werden. Bei einer angenommenen Übertragungsrate von 200 Mbit/s würde die reine Übertragung von 46 kB etwa 2 ms dauern. Demnach dominiert zeitlich offensichtlich der Overhead inkl. Signallaufzeiten, der zur Versendung eines Pakets notwendig ist. Sollte dieser Overhead nicht von der Datenmenge im Paket abhängen, dann ist die Versendezeit für ein Paket in diesem Bereich nahezu unabhängig von der Menge der zu versendenden Daten im Paket. Eine günstigere Situation wird dann erreicht, wenn man die Anzahl der Gitterzellen erhöht, zumal der Aufwand für die Hydrodynamik pro Zeitschritt mit der 3. Potenz der Zellenanzahl ansteigt.

Dieses Ergebnis hängt nicht stark von der WINSHIFT-Einstellung ab:

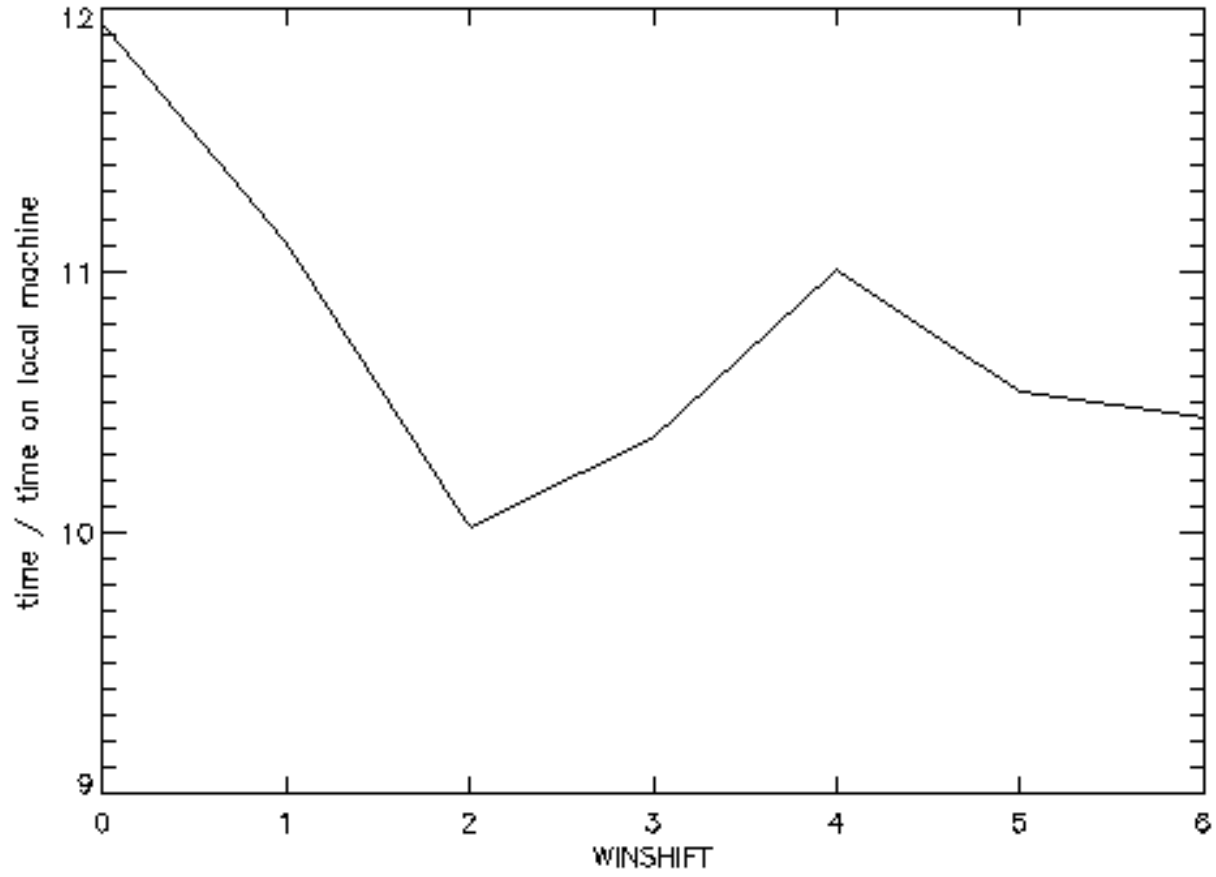

Abbildung P 3
Gesamtrechenzeit für 10 PROMETHEUS Zeitschritte in Einheiten der Rechenzeit für die entsprechende lokale Rechnung

3.2.4.4.2 Simulationen mit höherer Anzahl an Gitterzellen bzw komplizierterer lokaler Physik

Aufgrund der detaillierten Analysen zum vorhergehenden Fall ergibt sich, dass für Metacomputing-Berechnungen wesentlich günstigere Situationen durch Erhöhung der Gesamtzahl an Gitterzellen einerseits, durch kompliziertere lokale Physik mit aufwendigem Kernreaktionsnetzwerk und aufwendiger Zustandsgleichung andererseits, erzielt werden.

Bei der Vergrößerung der Gitterzellen von 50 x 50 x 100 auf 400 x 200 x 200 Gitterzonen erhält man ein deutlich günstigeres Verhältnis von Rechenleistung zu Kommunikation. Solche Anforderungen ergeben sich typischerweise bei der Berechnung von nur Typ 1A Supernovae mit einer etwas komplexerer Zustandsgleichung, aber einem einfachen Kernreaktionsschema.

Der andere Fall mit komplizierter lokaler Physik liegt beim Nova-Problem vor. Im Fall einer Supernova-Simulation bzw. einer Simulation von Verbrennungsmotoren.liegen Probleme mit komplexen Reaktionsnetzwerken für Kern- oder chemische Reaktionen vor, die ebenfalls günstigen Fall für verteilte Metacomputing-Berechnungen darstellen.

3.3 Untersuchungen im Rahmen von heterogenem Metacomputing

Im Rahmen von heterogenen Metacomputingstudien wurde das Fujitsu Vektorsystem VPP700 am LRZ mit dem Cray T3E-System am RZG über das GTB gekoppelt. Es wurden zwei Kopplungstypen mit zwei verschiedenen Anwendungsprogrammen untersucht.
Für eine Anwendung aus der Polymerforschung erfolgte eine Kopplung mit einer selbst implementierten TCP/IP-Schnittstelle im Code. Die Anwendung selbst bestand aus zwei eng gekoppelten, unterschiedlichen Anwendungsteilen.
Für eine zweite Untersuchung zur Verteilung einer homogen strukturierten Anwendung auf zwei verschiedene Rechnerarchitekturen wurden die Kopplungssoftware PAXC-MPI und der ab-initio Molekulardynamik-Code CPMD eingesetzt, der bereits für homogene Metacomputingsszenarien verwendet worden war.

3.3.1 Coupling Vector Parallel and Massively Parallel Computers: A Case Study from Polymer Physics (M. Pütz, MPIP)

3.3.1.1 Abstract

Initially the potential of coupling two or more super-computers to solve compute intensive scientific problems is discussed from a rather general point of view. Many aspects which favor such an endeavor are elucidated in conjunction with the problems posed by the real world implementations of todays systems. To demonstrate how such an undertaking can be successful a special application from polymer science and its deployment onto a coupled system consisting of a Fujitsu-Siemens VPP700 vector parallel and CRAY T3E massively parallel computer is presented in detail. The application computes inter-molecular pair distribution functions of polymer mixtures by single-polymer Monte-Carlo (MC) methods combined with PRISM (Polymer Reference Interaction Site Model) which is a distribution

function theory specifically designed for polymers. This application is a ideal test case for a coupled VPP-MPP system, since on the one hand the MC part parallelizes very well, but is virtually not vectorizable, and on the other hand the PRISM calculation vectorizes easily, but does not scale too well with the number of processors. The time scale of switching between MC and PRISM calculation is well above the one second range, hence the stress on the coupling network between VPP and MPP machines is particularly low.

3.3.1.2 Introduction

In many areas of scientific computing there is a virtually insatiable demand for computational power. The more computer power becomes available the more details one can take into account in modeling our real world. In the past decades the computational power of the fastest super-computers in the world has increased tenfold about every five years. Each new generation of super-computers has opened the doors to new areas in all natural sciences and engineering. Between factors of 50 to 10000 times more powerful than the fastest single processor workstations super-computers let us do research now where ordinary (serial) computing would allow us access only in 5 to 10 years from now.

3.3.1.2.1 Vector- and massively parallel computers

In recent years we have mostly seen two distinctive super-computer architectures: Vector parallel processing (VPP) and massively parallel processing (MPP) ones. Computers of the first type are usually built from a few (up to about 100) special purpose vector processing CPUs which heavily use pipeline processing technology to increase throughput of large data streams applying repetitive instructions to this data.The drawback of this technology is that not all problems are well-suited for pipeline processing (vectorization), which limits the area of application of such systems. Even in those cases which are suitable for vectorization special programming techniques, regarding the layout of data structures and the choice of algorithms, must be applied by the programmer. Modern workstation CPUs also use pipelining to reduce latencies between memory and the various execution units of a CPU, but these pipelines are much shorter than in their vector processing counter parts. Well vectorized codes can gain a speed-up of 5 to 15 (sometimes even higher especially on very long data streams with a high degree of data reuse) on a single vector CPU over a scalar workstation CPU.
Massively parallel systems are super-computers consisting of many (up to about 10000) commodity workstation processors with distributed chunks of memory, each chunk usually shared between 1 to 4 closely coupled CPUs. A unit consisting of local memory and the processors sharing it is usually referred to as a node. These nodes are linked by high-throughput, low-latency networks. Execution speed-up over single CPU execution is gained by distributing a computational problem over as many CPUs a possible. Ideally each of N CPUs computes only a 1/N share of the whole problem. However most problems do not parallelize in such an ideal manner and part of the calculations have to be done redundantly among the CPUs and intermediate results have to be exchanged among the CPUs. Hence the need for the fast inter-processor networks. In general a speed-up of several hundred times (some higher than 1000 times) the speed of a normal workstation can be achieved depending on the degree parallelism for a given problem. In order to reach similar acceleration with VPP

architectures one needs to parallelize over several vector processors as well on top of vectorizing the program. Due to the different strengths and weaknesses of the distinct architecturesit depends very much on the problem at hand which of the two offers the greatest potential.

3.3.1.2.2 Methods of Parallelization

Since parallel execution is inevitable for highest performance I shall briefly discuss the various strategies currently available for paralleling codes pointing out their advantages and disadvantages.

Automatic Parallelization. By far the simplest way, from a programmers point of view, is to utilize the support of many commercial compilers for generating multi-threaded code for the parallel execution of loops on multi-processor architectures with shared memory. If the loops are simple enough for the dependency analyzer of the compiler to detect inherent parallelism this requires little else but a recompilation. However most codes need some extra effort, like rearranging of some loops, introducing a few temporary variables or placing some helpful hints to the compiler to remove data dependencies which inhibit automatic parallelization. Almost any code is suitable for parallelization of this type to some extent and the obtainable speed-ups typically range between 1.5 and 10 on shared memory machines.

Parallel Programming Languages. To achieve higher speed-ups and at the same standardize loop-based parallelization huge effort has been spent on devising language extensions to standard procedural programming languages like FORTRAN and C which simplify the programmers task to distribute workload among processors. The most prominent and mature language extensions of this type are High Performance FORTRAN (HPF) and OpenMP. Both extend the base languages (FORTRAN and/or C) by additional constructs and keywords which allow the programmer to explicitly express parallelism in both algorithms and the distribution of data. The difference between HPF and OpenMP is mostly semantic: HPF extends the FORTRAN 90 standard directly and can only be compiled by a true HPF compiler, while OpenMP is a mostly language independent extension whose instructions are encapsulated in comments of the base language and are usually parsed in a pre-processing stage. Hence OpenMP codes can be compiled (as a serial version) with compilers which lack OpenMP support and therefore are more portable. Many codes can be parallelized very efficiently with this approach. On shared memory computers this is definitely the method of choice since data can be shared among processors without additional overhead. Typical speed-ups of OpenMP/HPF codes range from 4 to 100 over serial codes.

Explicit Multi-threading. If no parallelizing compilers are available one can usually parallelize a code by hand using the POSIX thread library which is supported by almost any UNIX-like operating system.Basically this is the same technique used by parallel programming languages without any specific language support for data layout and parallel loops. However it is much harder and more error-prone to program without them. Moreover inter-node communication between shared memory nodes cannot be handled in this way (which some HPF and OpenMP compilers are capable of). Thus if one can be avoid it one should generally not use explicit thread programming for scientific codes, but rather use OpenMP instead.

Message Passing. Short of programming communication hardware directly standard message passing libraries like MPI (Message Passing Interface), PVM (Parallel Virtual Machine) or CRAY SHMEM are the most explicit methods available for inter-node parallelization. Data layouts and communication schemes for all bot the most elementary constructs are the responsibility of the programmer which makes the parallelization of existing serial codes usually a very time consuming process. For newly developed codes which take message passing into account during the planning and specification phases the additional amount of development time is usually worth the effort, since the message passing method usually yields the highest possible speed-ups on distributed memory computers (like most MPP computers).

Hybrid Methods: OpenMP plus Message Passing. The recent trend in designing MPP systems is to cluster strong SMP nodes with 4 or more processors to reduce the load on the interconnect networks between the nodes.
However message passing codes usually do not exploit the fact of the tight coupling of the CPUs with a SMP node (except for a usually slightly more efficient message passing implementation for intra-node pathways using shared memory). Rather each CPU on a SMP node will have to communicate with other CPUs on different nodes. A hybrid parallelization using e.g. OpenMP parallelization inside a shared memory node and message passing between nodes offers ideal utilization of both shared memory and network. The drawback is of course increased programming complexity since a code essentially has to parallelized twice using different strategies. For many projects which only continue for a limited amount of time the effort is most likely prohibitive. Only in large scale projects with a high degree of code reuse over many years such effort would eventually pay off by the increased throughput of such a code.

3.3.1.3 Meta-computing

Many problems can be mapped more or less ideally onto a specific target architecture. In such cases this particular hardware should be used if it is available since implementation time, execution time and frequently also the total cost can be minimized at the same time. For example, molecular dynamics with short range forces can be parallelized quite efficiently using the domain decomposition approach and can also be vectorized at the same time. However vector speed-up is usually fairly low on most VPP architectures compared to MPP architectures with scalar CPUs, hence using a MPP platform provides the most efficient solution. On the other hand codes which involve large arbitrary matrix operations or molecular dynamics with long range interactions often parallelize poorly using message passing approaches since they involve costly all-to-all communications but at the same often vectorize extremely well. For such problems VPP architectures with shared memory are the best choice. Monte Carlo codes which use Markov chains to generate statistical data parallelize trivially by replicating the system on many processors (unless the simulation time on a single CPU for a full relaxation time of the Markov chain is too long) and involve a bare minimum of communication to add up the results. Such problems are ideally suited for clusters of workstations which offer the best price/performance ratio.

An equal amount of problems are not as easily mapped onto a single super-computer architecture. Often parts of a program run well while another part is running very slowly on a particular system. Molecular dynamics with both short and long range forces is an example

for this type of problems: The short range part of force calculations runs nicely on MPP hardware while the long range part favors VPP machines. Many algorithms have been invented during the past two decades to cast the long range force calculations into a form more suitable for MPP hardware with some but limited success. Most often it is the long range force part which limits the scalability of such calculations on an increasing number of CPUs. Ideally each part should run on hardware which executes it most efficiently with the minimum amount of programming effort. If such parts of a problem are separable, either in a parallel or serial execution, one could actually run each part on the most ideal architecture on a cluster of super-computers. This is what the idea of meta-computing is all about: clustering of different computers each with strengths for particular problem classes to overcome the weaknesses of their peers in solving other parts.
Clearly such an undertaking involves problems of its own kind. Most message passing libraries or parallel compilers only work on a single hardware and not for distributing code between different ones. The message passing which do support heterogeneous clustering on the other hand have no support for the special communication hardware on most super-computers. Hence one has to use different approaches for inter- and intra-machine parallelization. Another problem is the synchronization of jobs on two computers. Most systems come with proprietary batch queuing system which frequently are incompatible with each other (if they come from different vendors) and cannot easily be synchronized. However the latter could probably be overcome by using vendor independent solutions like the Portable Batch System (PBS). Moreover batch systems are usually not ideal for problems where calculations on different machines have to be serialized, i.e. part A executes on machine No. 1 while part B on machine No. 2 has to wait for part A to finish and vice versa. In dedicated execution environments a lot of resources would be wasted using standard batch systems. A possible way out would be to use over-committed execution environments where another job can be executed in the background if part of a high priority job has to wait for data from another machine. Clearly, the software environments of current super computer systems were simply not designed for clustering with other super computers and a lot of software development has yet to be done if meta-computing should become a viable option for large scale projects. Therefore before such effort is undertaken one needs to evaluate the true potential of this method, which is motivation for this work.

3.3.1.4 An Application from Polymer Science: Self-consistent PRISM Calculations

Theoretical polymer science involves the calculation of structural and dynamical properties for model polymer systems which is a computationally demanding task because of the large molecular size of these typically chain-like molecules and their tendency to form entangled and hence slowly relaxing structures in concentrated solutions and melts. While standard simulation techniques like molecular dynamics are applied in a wide range of studies the high computational demands of such calculations are often prohibitive for the study of very long chains. Considerable effort has been put into the development of computationally less demanding methods to predict structural information at least on a qualitative level. One such theory is self-consistent PRISM (polymer reaction site model) theory which combines the detail of atomistic Monte-Carlo simulations of single polymer molecules with approximate solutions of molecular liquid distribution function theory.

PRISM theory is an approximate extension of the standard RISM theory for small rigid molecules which itself is a rather straightforward extension of the Ornstein-Zernicke theory of simple atomic liquids. The P(RISM) equation couples the intra-molecular pair distribution function $\omega(r)$ of a single molecule with the inter-molecular pair distribution function g(r):

$$\hat{g}(k) - 1 = \hat{\omega}(k)\hat{C}(k)\left[\hat{\omega}^t(k) + \rho(\hat{g}(k) - 1)\right]. \tag{1}$$

The equation is most often written down in it's Fourier transformed representation since it is local in k-space. $\omega(r)$ describes the probability (up to a pre-factor) of finding another atom of the same molecule in the distance r of a given atom while g(r) describes the probability of finding another atom of a different chain in the distance r of a given atom in a given molecule. The knowledge of these two distribution functions allows one to calculate many physical properties of a model system in a very simple manner. ρ denotes the system density and C(r) is the so-called *direct correlation function* which - for short range interaction potentials V(r) - is well described by the Percus-Yevick approximation:

$$g(r) = (\exp(-V/k_BT) - 1)(g(r) + C(r)), \tag{2}$$

where T is the temperature and k_B is the Boltzmann constant. $\hat{g}(k)$, $\hat{\omega}(k)$ and $\hat{C}(k)$ are the Fourier transformed quantities of g(r), $\omega(r)$ and C(r) respectively.

Both equations (1) and (2) combined form a closed set of non-linear coupled equations which can be solved for g(r) if $\gamma(r)$ is known (all other quantities are control variables). If the polymer model contains n independent atoms types, g(r), $\omega(r)$ and C(k) become *n* x *n*-matrix valued distributions instead of just one single valued distribution functions and the total number of independent equations is multiplied by n(n+1)/2 (not n^2 due to some symmetries) and ρ is replaced by the partial atom density for each species. A solution for these equations may be obtained using Newton-Raphson methods or Picard iteration schemes. The latter are simpler to program and also appear to be somewhat more robust for the type of equations considered here. The Picard iteration simply starts with an initial guess (which can actually be fairly crude) for $g_{old}(r)$. From that the PY equation is solved for C(r) which is then used in the PRISM equation to re-compute $g_{new}(r)$. This process is iterated until g(r) and C(r)become stationary. (For technical reasons one works with g(r)=g(r)-C(r)-1 instead of g(r) to increase robustness and one uses $\alpha\gamma_{new}(r)+(1-\alpha)\gamma_{old}(r)$ with $0 < \alpha < 1$ as the new guess for $\gamma(r)$ instead of $\gamma_{new}(r)$).
Depending on the quality of the initial guess, the choice of α and the number of equations (atom types) a solution with acceptable accuracy is typically reached in 10 to 100000 iterations.
One problem of PRISM theory is that it is incomplete with respect to $\omega(r)$ which has to be supplied by different means. One way to calculate $\omega(r)$ approximately is to perform a Monte-Carlo simulation of just a single molecule. However this neglects many-body effects mediated by the surrounding medium of other molecules. If one assumes that these mediated interactions can be modeled by a pair interaction potential W(r) between the atoms of a single molecule one can derive a density functional approximation on how W(r) should look like:

$$W(k) = -kT \cdot C(k)\,[\omega(k) + \rho\,(g(k) - 1)]\,C(k). \qquad (3)$$

Now the Monte-Carlo simulations provide us with a guess for ω(r) starting with an initial guess W(r)=0. The PRISM-PY equations (1) and (2) can then be solved for g(r) which may finally be inserted into eq. (3) to compute a refined guess for W(r) . This loop may then be iterated until a stationary solution for W(r) is found. Thus we have a complete theory for ω(r) and g(r) as a function of ρ, T and V(r) . This loop is described graphically in Fig. 1. The results of this theory have been shown to give good qualitative to semi-quantitative agreement with much more expensive molecular dynamics simulations at about one percent of the their cost.

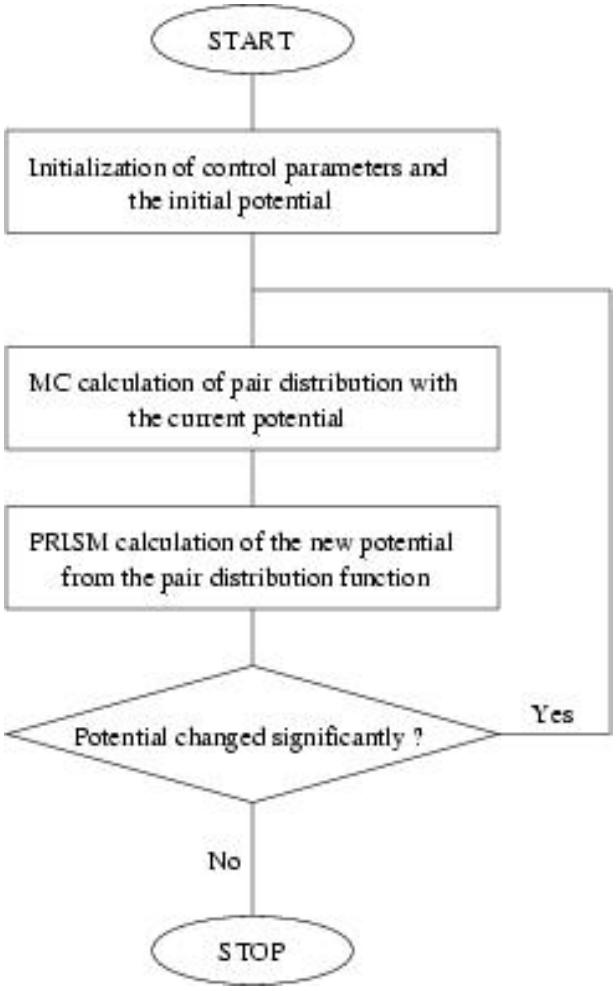

Figure 1.: *Self-consistent PRISM algorithm explained in a flowchart.*

3.3.1.4.1 Determining the best Computer Architecture

For fairly simple systems with just a few atom types and/or small molecular weights self-consistent PRISM calculations may easily performed on single CPU workstations with run-times of a couple of hours for a full self-consistent run. For the more interesting case of longer chains the effort of the Monte-Carlo simulations grows strongly with the chains lengths, whereas the solution of the PRISM equations is only weakly dependent on the chain length. However the effort of the latter depends strongly with the number of atom types in the model. Thus for more detailed models with long molecular weight chains the computational workload may require the use of super-computers to achieve results in a reasonable amount of time.

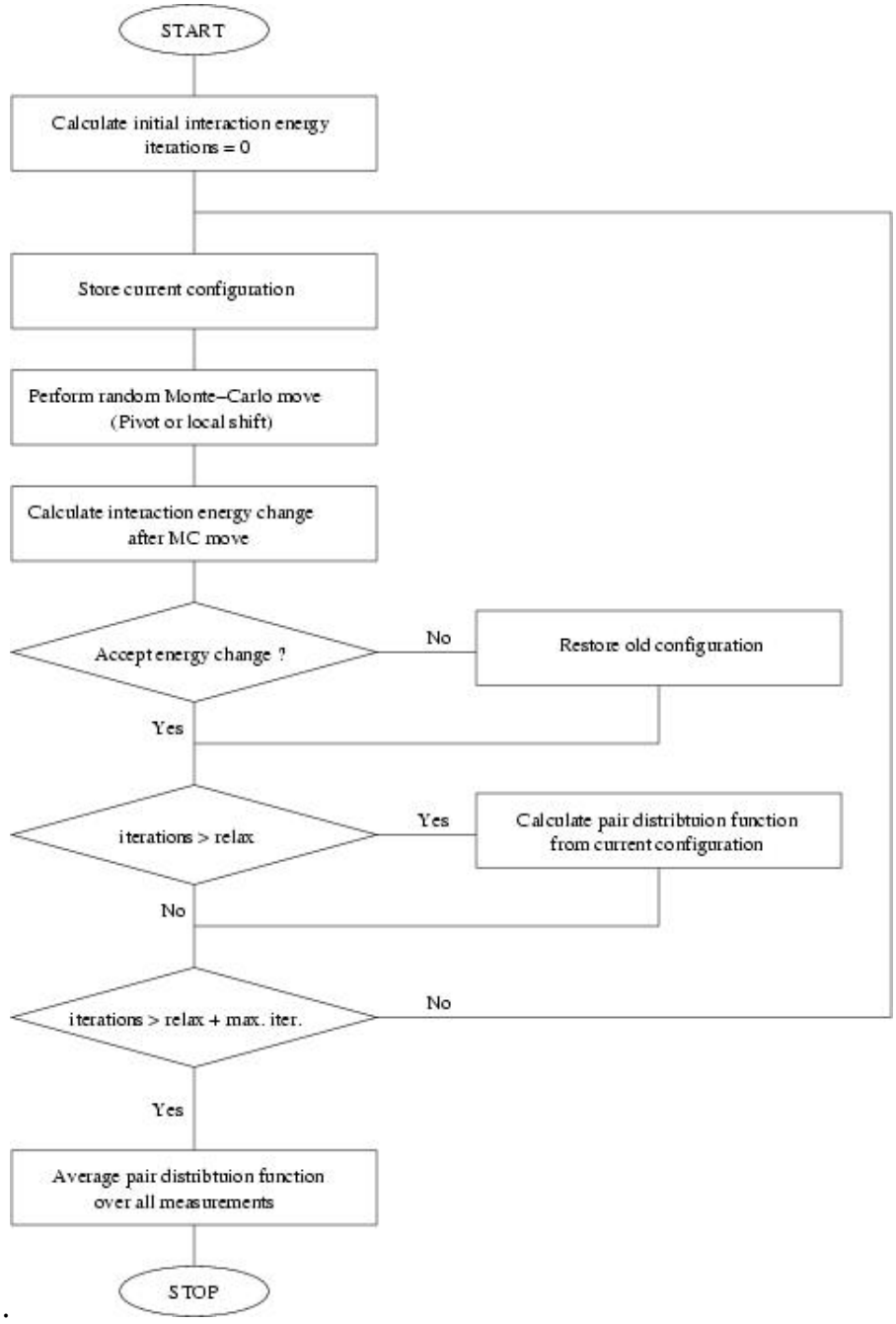

Figure 2.: *Flowchart of the single chain Monte-Carlo algorithm.*

The nature of the Monte-Carlo algorithm (depicted in Fig. 2) is such that it cannot be vectorized efficiently since inner loops typically have very short lengths (typically less than 10) and also vary strongly in length. Loop reversal to increase vector lengths is only successful in parts which are not time critical anyway. However m independent Markov chains of length *maxiter/m* may be started in parallel and the results of all chains can be averaged over to obtain an estimate for intra-molecular pair distribution function ω(r)with the same statistical quality as a single chain of length maxiter .
The only drawback is that each chain also needs its own initial equilibration phase of length warmup. Considering the fact that typically *maxiter* 1000 x *relax* the parallel efficiency is

$$\epsilon = \frac{t/m}{t/m + t/1000} = \frac{1}{1 + 1000/m} \qquad (4)$$

which for m = 100 is still about 0.9, i.e. corresponds to a speed-up of 90 if 100 chains are started on different CPUs. The code modifications for parallelizing the Monte-Carlo parts are rather simple and are shown in Fig. 3.

On the other hand the solution of the PRISM equations (see Fig. 4) is mostly dominated by the fast Fourier transforms to switch back and forth between k- and r-space representations of the fields and the PRISM solver in k-space. Since the Fourier transforms are one-dimensional they hard to parallelize in an efficient way as they essentially involve all-to-all communications and most vendor supplied FFT-libraries only support parallelized multi-dimensional FFTs but not one-dimensional ones. There is limited amount of parallelism in the Fourier transformation of the *n(n+1)/2* independent fields which is easy to exploit due to the fact the matrix element of the matrix valued distribution functions are independent from each other and hence may be transformed separately. This allows one to obtain speed-ups of order *n(n+1)* fairly easily. FFTs execute very quickly on vector processors with speed-ups ranging from 5 to 20 over scalar processors depending on the hardware which one compares. Since most VPP vendors offer well optimized FFT routines for the platforms solving the PRISM equations on a VPP architecture seems the most natural choice for this part of the problem.

The solution of the PRISM equation itself is a highly parallel problem again since it is local in the k-index. On VPP architectures the k-index has to be chosen as the vectorization direction so that parallelization and vectorization inhibit each other slightly. Even so it is easy to parallelize this part easily one has to redistribute the data again for FFT for conversion into r-space again. It is this redistribution step which severely reduces the scalability of the PRISM solver.

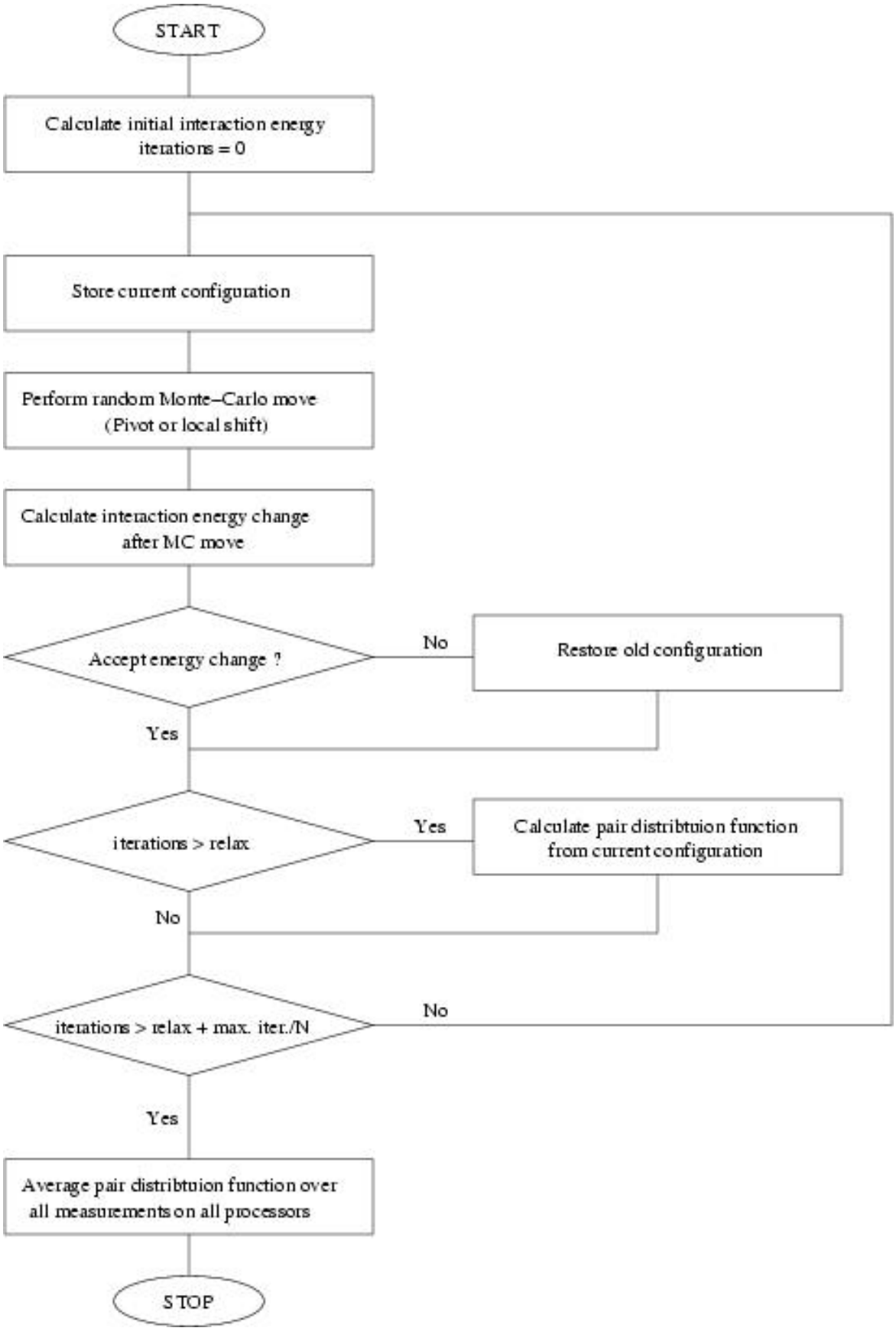

Figure 3.: *Flowchart of the parallelized Monte-Carlo algorithm.*

3.3.1.5 Coupling VPP and MPP architectures

In order to distribute the Monte-Carlo simulation and the solution of the PRISM equations over two different computer platforms a message passing approach offers a suitable pathway. Since the steps to parallelize the Monte-Carlo and PRISM parts independently of each other are easy enough an overall message passing approach is most compelling The inter-platform communication was implemented using TCP sockets directly. For that a simple TCP socket application programming interface (API) was developed consisting of just four basic functions:

1. *connect* : establishes a two-way TCP channel between to programs

2. *disconnect* : destroys a channel created by connect

3. *send* : synchronously transmits a message of given byte length over a channel

4. *recv* : synchronously receives a message of given byte length over a channel.

Fig. 5 demonstrates how these API functions were used to split PRISM and Monte-Carlo parts into two functional units which could then be deployed on different platforms. Schematically the PRISM solver takes over the role as a server for the Monte-Carlo client part. Initially the client starts the Markov process and calculates an estimate for the intra-molecular pair-distribution function ω (r). This result is passed on to the PRISM server which has been idle so far. The PRISM server solves the PRISM equations and passes the resulting inter-molecular pair distribution function back to the Monte-Carlo which has been idle in the meantime. Finally the Monte-Carlo client re-computes the new solution potential and checks for convergence. The result of the convergence check is passed back to the PRISM client, and if negative both start the next iteration and exit otherwise.

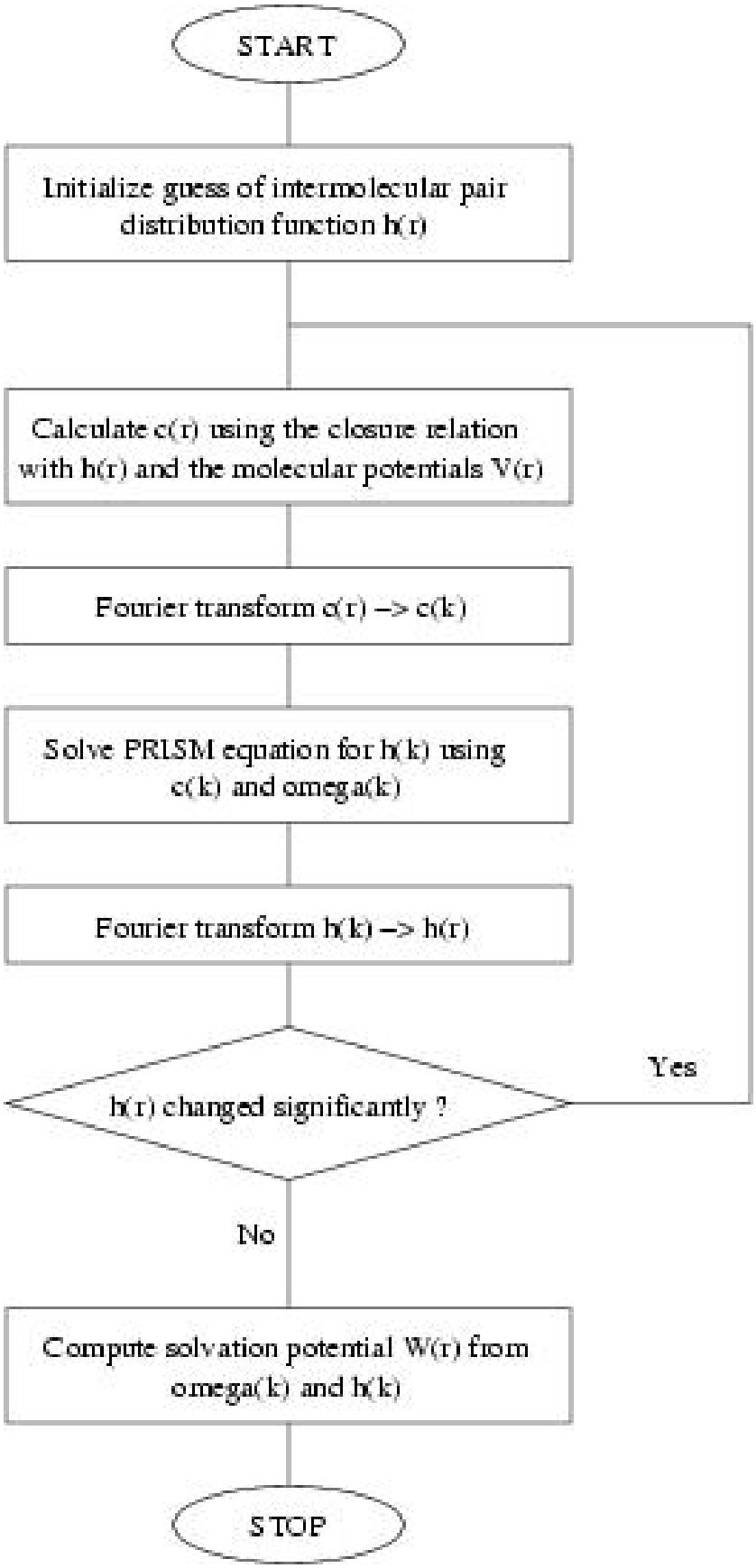

Figure 4.: *Flowchart of the PRISM equation solver.*

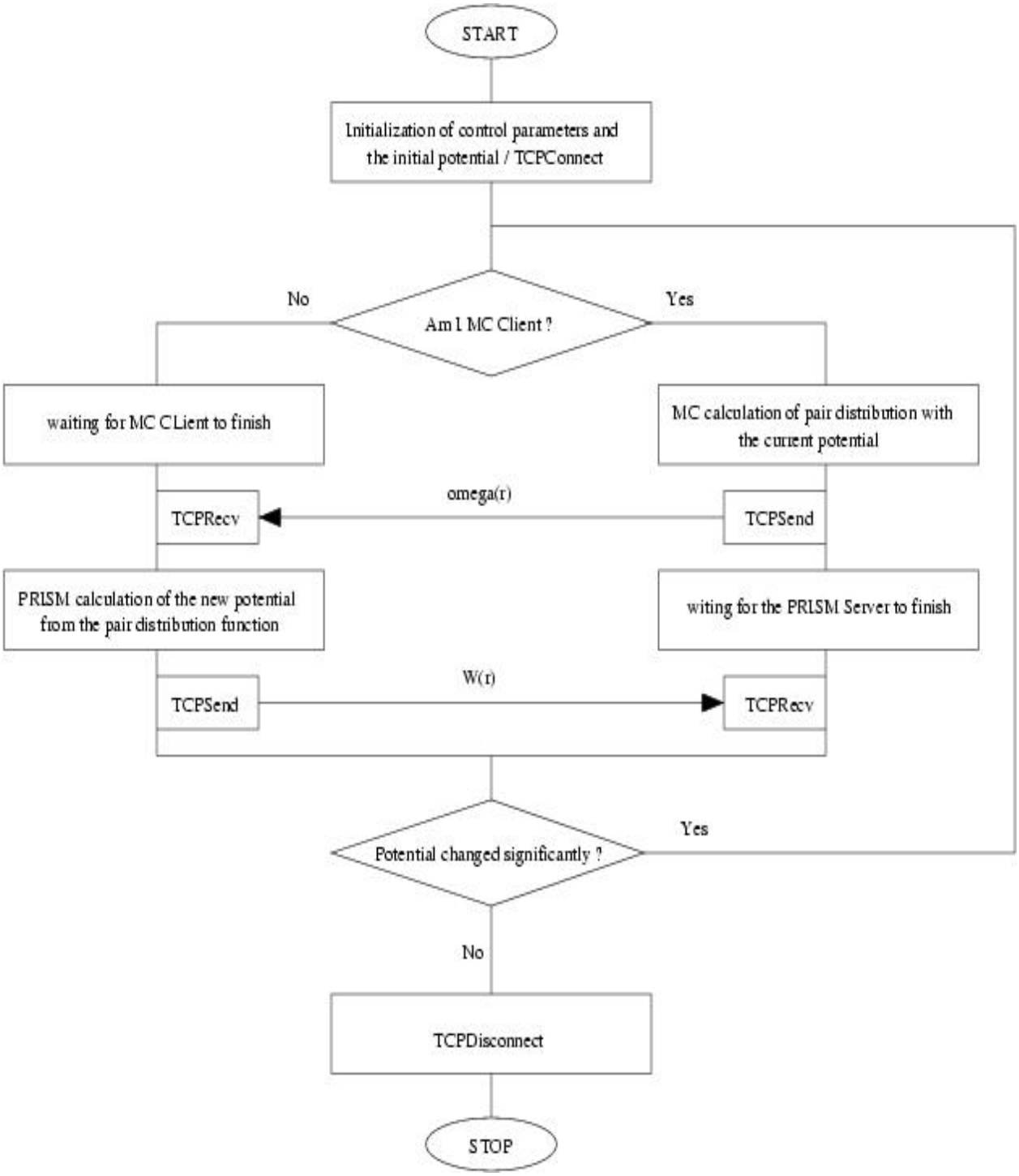

Figure 5.: *Distribution of the PRISM solver and MC parts as a flowchart.*

3.3.1.6 Benchmarking and Evaluation

For benchmarking purposes I have chosen a typical real world application. The pair-distribution function matrix of a homogeneous mixture consisting of two different polypropylen species, isotactic and syndiotactic, had to be calculated for a volume ratio of *1/2*. The chain length, N, was chosen to be 216 carbon atoms, the number of independent united atom types, n, was 6 (the backbone sites *CH*, CH_2 and the CH_3 side groups for each species) and the grid size for the distribution functions, G, was 8192 with a spacing of 0.1 Angstroem. Only the first iteration of the fully self-consistent loop was carried out. The total effort for the MC part depends on the chain length N proportionally to $N^{2.2}$. The effort for a single MC move scale like N , but the number of pivot Monte-Carlo steps (MCS) necessary to get to a new state in the Markov chain (relaxation time) which is statistically independent from the previous one scales like $N^{1.2}$. The total effort for the PRISM part scales like n^2G, where G itself scales like $N^{1/2}$. Thus it depends strongly the relative size of the parameters *N* and *n* which part dominates the calculation.

The relaxation time of the MC algorithm for a 216 unit chain is about 400 MCS. About 10000 independent chain conformations are sufficient to obtain an accurate enough estimate for the intra-molecular distribution functions, hence a total number of 4 x 10^6 MCS had to be generated plus 20 x 400 MCS steps for initial relaxation (about 20 times the relaxation time at the minimum). The 4 x 10^6 MCS may be distributed among the processors, but the initial relaxation steps have to be performed by each CPU. I.e. for 500 CPUs each has 8000 relaxation and 8000 measurement MCS to perform, which corresponds to a parallel efficiency of *1/2* (overall speed-up of about 250) and can therefore be considered as the maximum reasonable number of CPUs for this problem. Note, that ratio of initial relaxation and measurement has a value of about 500 , independent of the chain size, hence the maximum speed-up for the pathway chosen to parallelize the problem will never be better than 500.

For the PRISM part we cannot expect a better speed-up than $n(n+1)/2 = 21$ when a non-parallelized one-dimensional FFT is used. Even if a parallelized one was used the communication for switching k- and r-space representations of the distribution functions, which is of the all-to-all type, would inhibit further scaling, simply because there is not enough work to do between communication steps as will be seen in the next section.
The computations for the Monte-Carlo part were carried out on the 800 node strong CRAY T3E of the Computing Center (RZG) of the MPG in Garching while the PRISM calculations were performed on the Siemens-Fujitsu VPP700 of the Leibniz Computing Center (LRZ) of the Technical University in Munich with 52 vector processors. Each CPU of the T3E has a peak scalar performance of 600 MFlops and 128 MB of distributed memory with 1.2 GB/s bandwidth. The vector processors of the VPP700 have a peak vector performance of 2.2 GFlops and 2 GB distributed (crossbar connected) memory with a bandwidth of 18 GB/s. The scalar peak performance of these CPUS is only 275 MFlops. Both machines are quite typical representatives of the MPP and VPP architectures respectively. Moreover both systems were released in 1996, hence a relative comparison of their computing power is fair.

3.3.1.6.1 Performance

Fig. 6 shows the execution times of the MC calculation for an increasing number of CPUs on both T3E and VPP700 systems. Clearly the VPP700 is an unsuitable platform for this part of the problem being a factor of 2.4 slower than the T3E on a per processor basis.

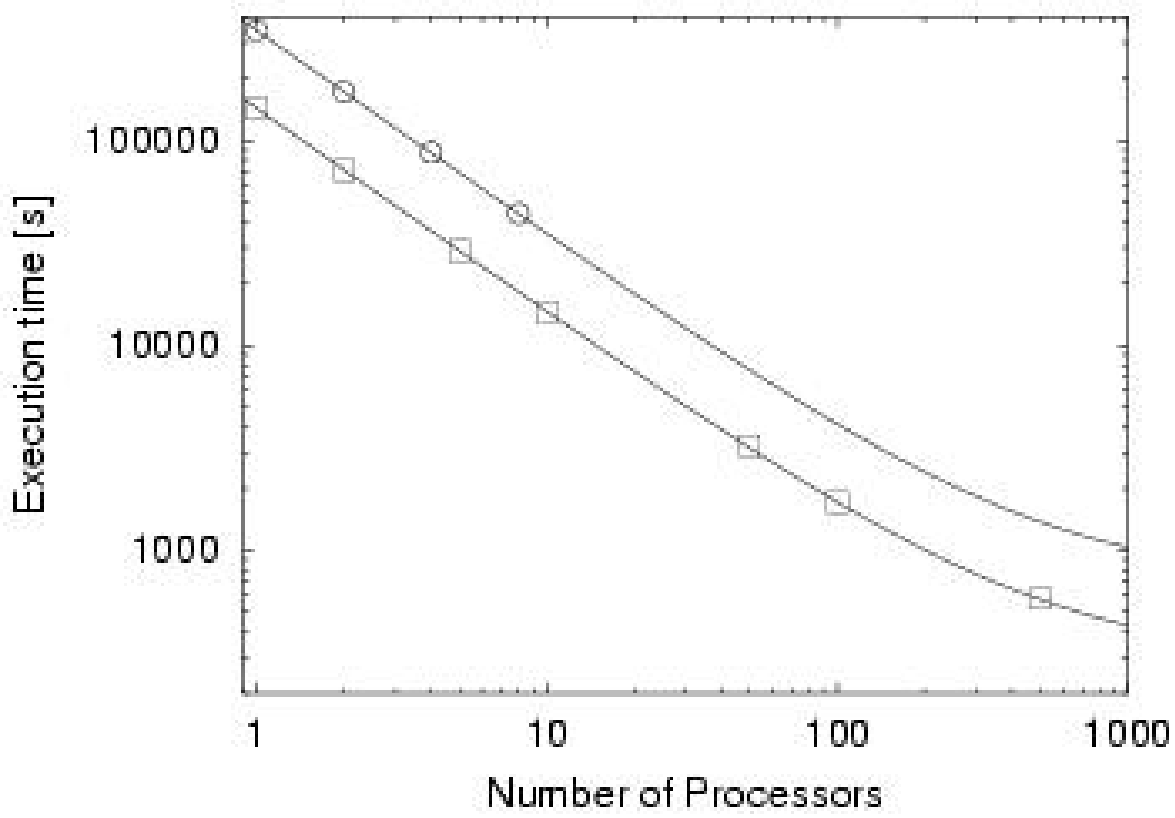

Figure 6.: *Stand-alone Monte-Carlo benchmark execution time scaling with the number of processors on the T3E (boxes) and VPP700 (circles).*

Fig. 7 on the other hand show the same for the PRISM calculations. Here we see a different picture. While at 21 CPUs both VPP700 and T3E stop scaling the overall execution is about a factor of 5 faster on the VPP700 than on the T3E. This clearly demonstrates that the expectations at the beginning of this work have been met.

To elucidate the benefit of a distributed execution of the problem Fig. 8 compares the execution times for the full problem on about 1/8 of the CPUS of each machine (100 on the T3E, 7 on the VPP700) on each machine separately and for the case when the MC part runs on 100 CPUs of the T3E and PRISM on 7 CPUs of the VPP700. Clearly the speed-up of the combined effort is much better than just the inverse arithmetic mean of the two machines separately. Note, that due to the meager Monte-Carlo performance of the VPP700 and the bad scalability of the PRISM solver little would have been gained to utilize the respective idle CPUs to support the other machine.

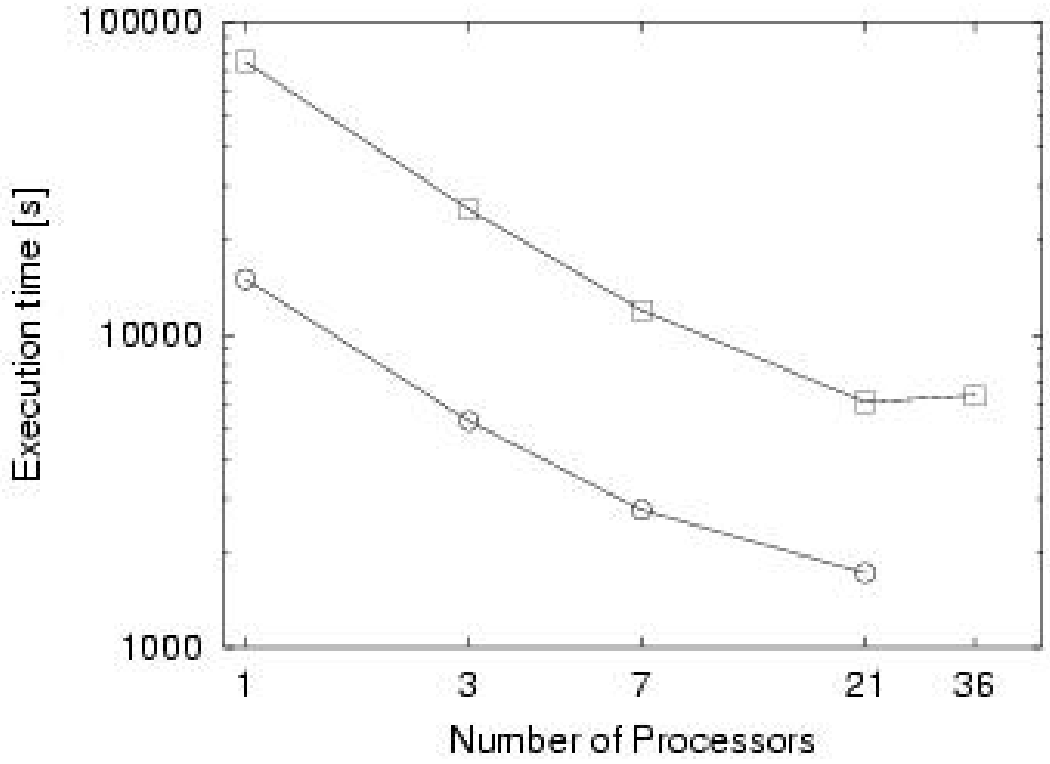

Figure 7.: *Stand-alone PRISM benchmark execution time versus the number of executing processors on the T3E (boxes) and VPP700 (circles).*

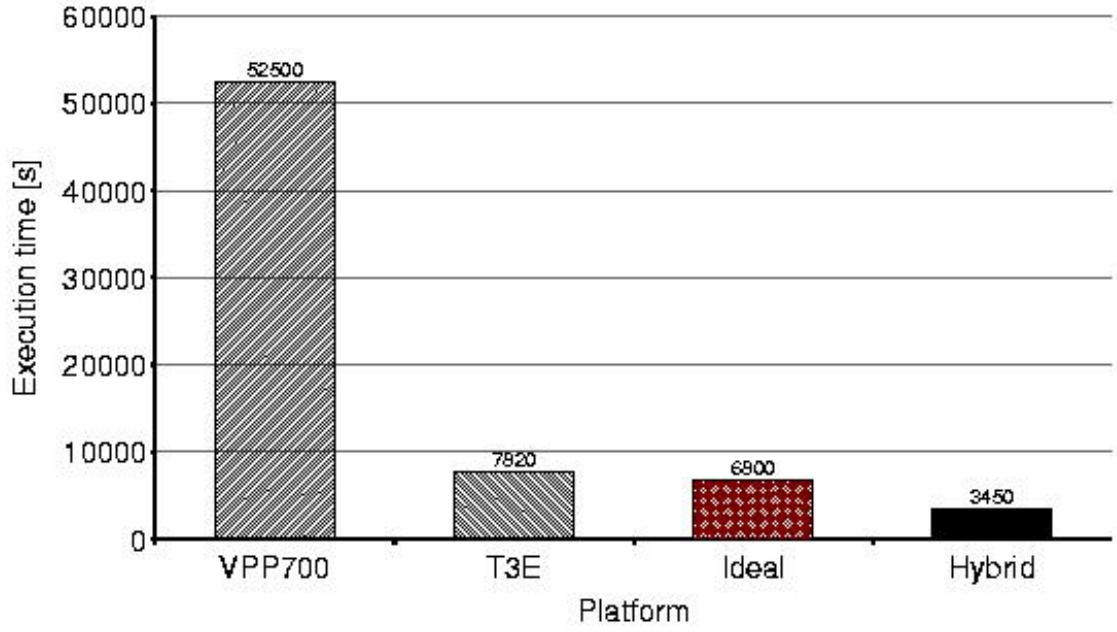

Figure 8.: *Combined Monte-Carlo/PRISM iteration execution times on a VPP700 (using 7 CPUs), T3E (using 100 CPUs), on a clustered VPP700-T3E (100/7 CPUs) system for symmetrical distribution (idealized) and on the clustered VPP700-T3E (100/7 CPUs) using the described asymmetric distribution of workload.*

3.3.1.6.2 Programming Effort

How long did it take to make the necessary modifications for vectorizing and parallelizing the code ? The answer to this question provides another measure, on top of the pure performance of the code, which is important for evaluating the potential of meta-computing.
The serial code itself took about 7 months of dedicated developing time including design, programming and debugging phases and about 2 weeks of tuning. Vectorization of the PRISM calculations took about one week and parallelization of the entire code using MPI took about another week (again including all phases). The coupling of the two computers over TCP sockets cost an additional 2 weeks. However 90% of this time went into the development of a small message passing library to encapsulate the necessary communication functionality. This time should not be taken into consideration for the following efficiency analysis since it was a one time effort and the library may be reused in other projects.

The ratio of tuning/vectorization/parallelization and functional programming was about 4/14=0.29. For comparison, if the whole code had to be tuned for each platform separately this ratio would have been about 5/14=0.36 on the T3E (about two additional weeks for improved parallelization of the PRISM minus one week of vectorization for an additional speed-up of at most 2 in the PRISM calculation) and about 6/14=0.43 on the VPP700 (about 2 additional weeks for partial vectorization of the MC part for an additional speed-up of at most 2.5 in the MC calculation). Hence even without the aspect of improved performance meta-computing wins in terms of pure implementation time.

3.3.1.7 Conclusions

The analysis clearly demonstrates that meta-computing definitely has a lot of potential which manifests itself in three main advantages:

- Maximum performance when each part of a program runs on a platform which executes it in the fastest way.

- Minimum programming and tuning effort for each part.

- Both of the above automatically lead to the most cost effective solution since hardware is exploited in its most efficient way and valuable implementation time is saved.

At the same time it must be stated, that currently meta-computing is not well supported by supercomputing facilities. Lack of interoperability of queuing systems and the high latencies of current long distance and high bandwidth interconnects severely cripple the opportunity for large scale applications.

Clearly, clustering different computer platforms with supplementing strengths and weaknesses may well prove to be a much more efficient solution for tomorrows super computing applications than more expensive architectures which try to be equally potent for every task. It is mainly a question of availability of the necessary software middle-ware whether super computer clusters can succeed in the future.

References

I Detailed up-to-date information on current super-compting facilities and parallel programming methods can be found on the world web under the following portals:

- http://www.top500.org
- http://www.openmp.org
- http://www-unix.mcs.anl.gov/mpi

II Introduction into simulation techniques relevant to polymer science and introductions into vectorization and parallelization are covered by the following books:

- R.W Hockney and J.W. Eastwood, "Computer Simulation using Particles", McGraw-Hill, New York (1981)
- M.P. Allen and D.J.Tildesley, "Computer Simulation of Liquids", Clarendon Press, Oxford (1987)
- M.C. Rapaport, "The Art of Molecular Dynamics Simulation", Cambridge University Press, New York (1995)

III Monte-Carlo Algorithms for polymers are covered in detail by the book

- K. Binder (Editor), "Monte Carlo and Molecular Dynamics Simulations in Polymer Science", Oxford University Press, Oxford (1995)

IV PRISM Theory is reviewed in

- K.S. Schweizer and J.G. Curro, Adv. Pol. Sci., 116, S. 321, 1994
- K.S. Schweizer and J.G. Curro, Adv. Chem. Phys. 98, S. 1, 1997

V Recent results obtained with the program and a complete description of the simulation methods are published in:

- M. Pütz, J.G. Curro and G.S. Grest, J. Chem. Phys. 114 (2001), in press
- T.C. Clancy, M. Pütz, J.D. Weinhold, J.G. Curro and W.L. Mattice, Macromolecules 33,S. 9452 (2000)

3.3.2 Untersuchungen mit PACX-MPI und Molekulardynamik-Code CPMD

Für eine Untersuchung der Verteilung einer homogen strukturierten Anwendung auf zwei verschiedene Rechnerarchitekturen wurden die Kopplungssoftware PACX-MPI und der ab-initio Molekulardynamik Code CPMD eingesetzt.
Hierfür wurden zunächst die nötigen Implementierungsarbeiten auf dem Fujitsu VPP-System am LRZ München vorgenommen, um die Voraussetzungen für eine Kopplung mit dem Cray T3E-System am RZG zu schaffen.

3.3.2.1 Anpassungsarbeiten bei PACX-MPI und Test der Verbindung

Die ursprünglich verfügbare Version 3.7.0 der Interkommunikationsbibliothek PACX-MPI (HLRS, Stuttgart) war nicht darauf ausgelegt, Fujitsu VPP-Systeme handzuhaben. Dort werden parallele Jobs auf Prozessoren mit vorab nicht bekannten Internet-Adressen gestartet. Erst die Version 3.7.4 von PACX-MPI, die ab 21. Januar 2000 zur Verfügung stand, löste dieses Problem. Da bei den VPP-Systemen jeder Knoten eine eigene IP-Adresse besitzt, und die Zuordnung auf die Knoten erst zur Laufzeit bekannt ist, mußte der Startmechanismus für die Metacomputing-Anwendungen neu erstellt werden. Die folgende Abbildung veranschaulicht den gewählten Ansatz:

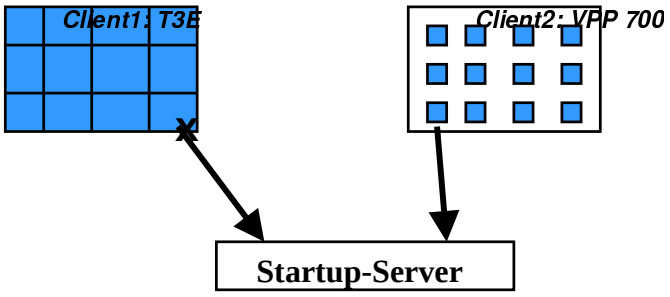

Abbildung PACX 2:
Startup-Server Konzept für PACX-MPI
für die Handhabung vorab nicht bekannter IP-Adressen

Der Startup Server dient dabei als Fixpunkt, der als einziger eine feste IP-Adresse besitzen muss. Im GTS diente dazu eine SUN, die am LRZ zur Verfügung stand. Da der Startup-Server in C++ implementiert wurde, und z.T. keine Compiler zur Verfügung standen, die dem ISO/IEC 14882 Sprachstandard entsprachen, wurden vom RUS Binaries für verschiedene Plattformen zur Verfügung gestellt (True64, Solaris, Unicos, Linux).

Damit die Anwendungsprogramme heterogen zwischen verschiedenen Rechnern ablaufen können, gibt es verschiedene Probleme zu lösen.

1. Unterschiedliche Zahldarstellungen auf den verschiedenen Rechnern: Hier geht es zuerst einmal um die Frage ob es sich um ein *big endian* oder *little endian* System handelt. Ein zweiter Punkt ist die Länge der Datentypen, z.B. ist ein INTEGER auf der T3E 64 Bit und auf der VPP 32 Bit lang. Als Konsequenz muss die PACX-MPI Bibliothek die versendeten Daten konvertieren. Dazu werden die Daten in die XDR-Darstellung überführt, die unabhängig von einer Rechnerarchitektur ist.

2. Die Applikation muss der MPI-Bibliothek mitteilen, welche Datentypen verschickt werden, damit diese die entsprechenden Konvertierungen durchführen kann. Der MPI Standard sieht dazu bei allen Kommunikationen vor, dass zusätzlich zur Angabe über die

Datenmenge die Art der Daten beschrieben wird. Diese auf homogenen Plattformen eigentlich überflüssige Angabe wird vom Standard vorgeschrieben, um einen problemlosen Einsatz auf heterogenen Plattformen zu ermöglichen. Alternativ dazu kann der Anwender bei der Angabe von MPI_BYTE als Datentyp eine Kommunikation ohne Konversion durchführen. In diesem Fall ist es in der Verantwortung des Anwendungsprogrammierers, dass im heterogenen Fall alles korrekt abläuft.

Da die notwendigen Konversionsoperationen einen zusätzlichen Overhead bewirken, kann bei PACX-MPI der Anwender zur Übersetzungszeit der Bibliothek entscheiden, ob er diese Konversion anschalten möchte (Option --enable-conversion beim Konfigurieren). In diesem Fall findet die Konversion unabhängig vom Zielrechner immer statt, da zum Zeitpunkt der Konversion der Zielrechner z.T. nicht bekannt ist (z.B. bei MPI_Pack, MPI_Unpack). Ein korrektes Ablaufen der Applikation im homogenen Fall bei aktivierter Konversion zeigt daher auch das prinzipiell korrekte Arbeiten der PACX-MPI Bibliothek mit dieser Applikation.
Das korrekte Arbeiten der Konversion wurde mit eigenen Testprogrammen und Programmen aus der MPICH Testsuite überprüft. Um die Konversion zwischen Big und Little Endian zu verifizieren, wurde auch heterogenes Metacomputing zwischen der Fujitsu VPP und Rechnern mit Intel Prozessoren durchgeführt.

3.3.2.1.1 Anpassung der Netzwerkkonfiguration

Da bei der Verwendung von PACX-MPI zwei Knoten der parallelen Applikation dazu verwendet werden, die externe Kommunikation via TCP/IP durchzuführen, musste die Netzwerkkonfiguration abgeändert werden. Ursprünglich hatte nur der Knoten vpp0 mit dem HIPPI Interface eine Route über die schnelle GTS Verbindung. Für die Verwendung von PACX-MPI wurde das Routing von den Einzelknoten der VPP zur GTS Adresse der T3E über den Knoten vpp0 konfiguriert. Der Verkehr fließt dabei zuerst über das schnelle interne Crossbar-Netzwerk des Fujitsu-Systems.
In der Gegenrichtung war das Routing von Anfang an so eingestellt, daß alle Knoten der VPP über die GTS Verbindung angesprochen wurden, daher war keine Anpassung erforderlich.

Die alternative Lösung, beide Kommunikationsprozesse auf dem Knoten vpp0 ablaufen zu lassen, scheitert an der Einschränkung der MPI Implementierung des Fujitsu-Systems. Es ist nicht möglich, zwei oder mehr Prozesse desselben MPI-Programms auf einen Knoten zu legen.

3.3.2.1.2 Performance Messungen

Die exakte Vergleichbarkeit der Messwerte von netperf und den korrespondierenden Testprogrammen von PACX-MPI ist leider nicht gegeben, da bei PACX-MPI nicht dieselbe Paketzahl erreichbar war.
Eine Erhöhung der Paketzahl bei den Ping-Ping Benchmarks war leider nicht möglich, da die MPI Bibliothek auf der Fujitsu VPP 700 mit „Receive queue/mailbox overflow“ abbrach. Dabei handelt es sich um einen Mangel in der Implementierung der MPI Bibliothek des Herstellers auf der Fujitsu VPP.

Ein weiteres Problem entsteht durch die Tatsache, daß es auf der VPP nicht möglich ist, die Zuordnung der einzelnen Prozesse eines parallelen Jobs auf die Knoten festzulegen. Damit ist es nicht möglich, die Prozesse, die Kommunikation via TCP/IP durchführen, auf den Knoten mit dem HIPPI Interface zu legen (vpp0). Wie oben beschrieben wird daher diese Kommunikation erst über das interne Netzwerk geroutet, bevor es dann auf das HIPPI Interface umgesetzt wird. Unsere Tests mit TCP/IP Benchmarks zeigten aber, daß die Performance dadurch nicht negativ beeinflusst wurde.

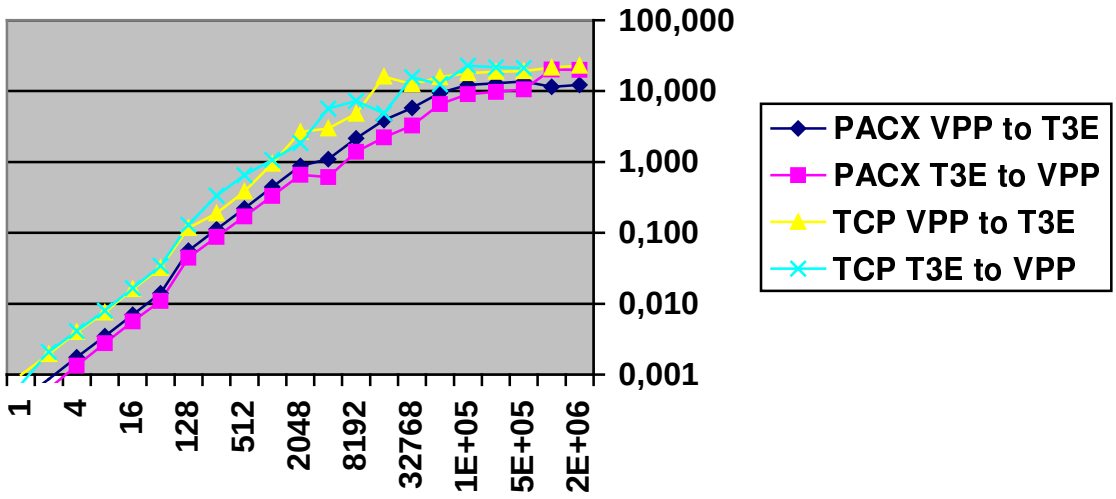

Abbildung PACX 3
Vergleich der Bandbreiten über TCP/IP und PACX-MPI.
Bei PACX-MPI war dabei die Datenkonversion aktiviert.

Insgesamt konnte bei der Kommunikation über PACX-MPI eine Bandbreite von bis zu 13.7 MB/s (110 Mbit/s) erzielt werden. Der Abstand zur theoretisch möglichen Bandbreite ist hier größer als im Fall der T3E. Die Gründe sind zuerst einmal die notwendige Konversion der Daten, die einen zusätzlichen Aufwand bedeutet, und die Tatsache, dass die einzelnen Knoten der VPP nicht dediziert zur Verfügung standen. Insbesondere hatte anscheinend die Last auf Knoten vpp0 einen großen Einfluß auf die Performance. Diese These wird gestützt durch die Beobachtung, dass bei Durchführung der TCP/IP Tests auf dem Knoten vpp0 die Performance abnahm. Sie lag dann in dem Bereich, wie er mit PACX-MPI beobachtet wird.

3.3.2.2 Einsatz von CPMD (MFK)

Im ersten Schritte wurde CPMD wurde auf das Fujitsu VPP-System portiert und für dortige Parallelverarbeitung angepasst. Der Code war nach der Anpassungsgarbeit korrekt und effizient parallel lauffähig.
Im nächsten Schritt wurde die Kopplungssoftware PACX-MPI mit einbezogen. Hierzu wurde das in der Abb. C 2 dargestellte Konfigurationsschema gewählt.

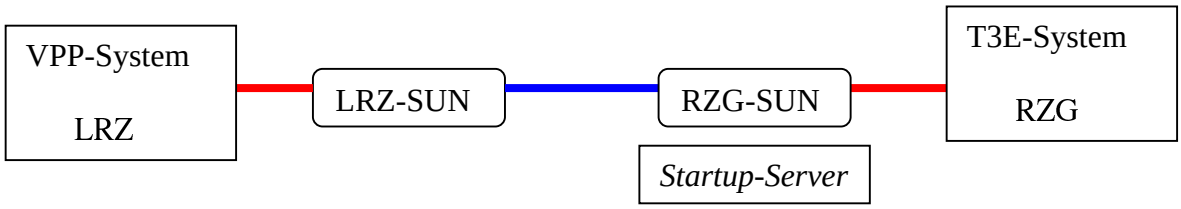

Abbildung C 2:
Konfiguration zum Test des heterogenen Metacomputings mit CPMD
HIPPI-Verbindung
GTB-Verbindung mit ATM
Implementierung und Start eines Startup-Servers auf RZG-SUN

Mit dem Startup-Server auf der RZG-SUN, auf der die HIPPI-zu-ATM Umsetzung für das T3E-System erfolgt, konnte eine PACX-MPI Verbindung sowohl zum T3E- wie auch zum VPP-System erfolgreich aufgenommen werden.

Im nächsten Schritt wurde die direkte Verteilung der CPMD-Anwendung auf das T3E- und VPP-System versucht. Als Voraussetzung wurden alle Systeme im Verbund als PACX-MPI *Hostnames* konfiguriert, da das VPP-System im Gegensatz zur T3E aus vielen verschiedenen Individualrechnern mit Individualadressen besteht.

Der Startup-Server auf RZG-SUN startete die vorbereiteten, lokal abgelegten CPMD-*Executables* auf dem T3E- und dem VPP-System, und die Programme begannen zu laufen und zu rechnen. Bei der ersten Kommunikation von Datenstrukturen mit PACX-MPI über das GTB blieben die Programme jedoch hängen. An dieser Stelle begannen nun aufwendige Analysen zum Verständnis des Problems.

Ergebnis:

In CPMD werden aus Effizienzgründen beim Datenaustausch verschiedene Datentypen (Integergrößen, Realgrößen) als eine einzige Datenstruktur verpackt als BYTE-Variable mit MPI kommuniziert (MPI_BYTE). Auf den unterschiedlichen Rechnerarchitekturen VPP und T3E werden Integergrößen jedoch unterschiedlich behandelt. Das T3E-System realisiert sie als 8 Byte Größen, das VPP-System als 4 Byte Größen. Dies führt zu einer Inkompatibilität und zum *Deadlock* der Programme. Als Lösungsmöglichkeiten wurden diskutiert:

- automatische Veränderungen von Integerpräzisionen per Compiler-Option auf den beiden Rechnern: Erhöhung der Präzision von 4 auf 8 Bytes auf VPP, oder Erniedrigung von 8 auf 4 Bytes auf dem T3E-System mit Hilfe der Fortran90 KIND Deklaration.

- Umsetzung der Integer-Deklarationen im Source Code mit Hilfe eines Präprozessors.

Beide Verfahren führen leider dazu, dass alle Integer-Datentypen verändert werden, auch an Stellen, wo dies nicht geschehen darf.

Folgende elementaren Software-Komponenten sind involviert:

CPMD-Version auf VPP, MPI-Bibliothek auf VPP, PACX-MPI auf VPP, PACX-MPI auf T3E, MPI-Bibliothek auf T3E, CPMD-Version auf T3E.

An die jeweilige Fortran MPI-Bibliothek werden vielfach Integergrößen als Parameter übergeben. Die Bibliothek erwartet dabei die jeweilige Default-Länge. Für Rückgabewerte gilt dasselbe. Weiterhin wird der Status einer MPI-Operation in Fortran als "Integer Status (MPI_STATUS_SIZE)" implementiert. Ändert man die Länge dieser Integervariablen, führt das zu Inkompatibilitäten. Deshalb dürfen auf dem VPP-System Integervariablen nur bei jenen Daten in Integer*8 Größen umgesetzt werden, die tatsächlich an das T3E-System kommuniziert werden. Werden dieselben Daten einmal nach "außen" zur T3E gesandt, und einmal als sonstiger Parameter (VPP-intern) an MPI übergeben, was das Fall ist, müssten für dieselbe Größe zwei verschiedene Variablen eingeführt und die korrekte Belegung durch entsprechende Kopiervorgänge sichergestellt werden. In CPMD dürften also nur jene Intergervariablen durch Integer*8-Größen ersetzt werden, die anschliessend von CPMD als MPI_BYTES verschickt werden. Bei expliziten Deklarationen von MPI_INTEGER hingegen muss weiterhin die normale Integer-Deklaration verwendet werden, da MPI_INTEGER*8 in PACX-MPI nicht unterstützt wird, da dieser Datentyp kein Bestandteil des MPI 1.1 Standards ist.

Diese aufwändigen Umgestaltungen des CPMD-Codes wurden nicht mehr durchgeführt, da sie den für diese Machbarkeitsstudie gesetzten Rahmen überstiegen hätten.

Aus diesen Untersuchungen mit CPMD in dem beschriebenen heterogenen Metacomputingszenario wurde die Erfahrung gemacht, dass es sehr schwierig sein kann, bei einem vorhandenen Produktionscode alle Programmteile verteilt auf zwei unterschiedlichen Rechnerarchitekturen zum Ablauf zu bringen, wenn sich die Programmierumgebungen insbesondere bzgl. Default-Einstellungen von Variablendeklarationen unterscheiden.

Als weitere Komplikation, auf die man bei erfolgreicher Verteilung der einen Anwendung auf die beiden ungleichen Rechnertypen gestoßen wäre, sei erwähnt, dass die auf Cray T3E und Fujitsu VPP eingesetzten Prozessoren in ihrer Leistung sehr stark voneinander abweichen (Faktor 10). Der Code CPMD behandelt aber derzeit alle Prozessen als gleichwertig, und allen Maschinen in der verteilten Rechnung wird dieselbe Anzahl von Prozessoren zugeordnet. Deshalb wäre weitere Umstrukturierungsarbeit nötig gewesen, um prinzipiell eine gute Lastverteilung zwischen den Maschinen und damit eine gute Effizienz zu erreichen.

4. Zusammenfassung der Arbeiten

Im Rahmen des Gigabit-Testbeds Süd und Berlin wurden Metacomputingstudien unter Einsatz des Cray T3E Hochleistungsrechner am Konrad-Zuse-Zentrum für Informationstechnik in Berlin (ZIB), des Cray T3E Hochleistungsrechners am Rechenzentrum Garching der Max-Planck-Gesellschaft (RZG), sowie des Fujitsu VPP Hochleistungsrechners am Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften (LRZ) durchgeführt.

Die erste Aufgabe bestand darin, die Hochleistungsrechner in Berlin, Garching und München an den Backbone des Gigabit-Testbeds anzuschließen. Da das GTB auf ATM-Technologie basiert, wurde zunächst an den genannten Zentren eine lokale ATM-Infrastruktur aufgebaut, die in allen Fällen auch für weitere GTB-Teilprojekte genutzt wurde. Da bei keinem der Hochleistungsrechner ATM OC12 Netzinterfaces unterstützt wurden, sondern die leistungsfähigsten Interfaces auf HIPPI-Technologie beruhten, war für eine effiziente HIPPI zu ATM Umsetzung zu sorgen. Für diesen Zweck waren keine kommerziellen, voll funktionsfähigen Switches verfügbar. Der Projektpartner RUS überprüfte in Zusammenarbeit mit der Fa SUN, die die nötige Hardware zur Verfügung stellte, die Effizienz einer SUN-Workstation mit HIPPI- und ATM OC12 Interfaces als Switch-Ersatz (siehe Anhang). Aufgrund der positiven Testergebnisse wurden drei gleichartige SUN-Workstations für ZIB, LRZ und RZG für die Etablierung der End-zu-End-Verbindungen der Hochleistungsrechner beschafft und eingesetzt.

Für diese Verbindungen wurde dann von EANTC/Berlin ein Class C Subnetz für Classical IP bereitgestellt, ein weiteres für die HIPPI-Interfaces. Aufgrund von Schwierigkeiten, ein funktionierendes CLIP-Netz mit dynamischem ARP für die Endanwender aufzusetzen, wurden durch die GTB-Betriebsverantwortlichen pflegeaufwendigere, statische PVCs geschaltet, so dass im Nov. 1999 die GTB-Verbindung der Endgeräte vorlag.

Der Phase der funktionellen Inbetriebnahme der End-zu-End-Verbindungen schloss sich die Phase der Leistungsoptimierung aller beteiligten Netzkomponenten an. Dabei konnte für die Verbindung zwischen den SUN-Workstations der einzelnen Einrichtungen über das GTB mit 400 bis 500 Mbit/s ein hoher Anteil der theoretischen Grenze von 622 Mbit/s erreicht werden. Die Hochleistungsrechner hingegen waren nicht in der Lage, Daten mit einer gleich hohen Rate über HIPPI-Interfaces zu kommunizieren. Zwischen den Cray T3E-Rechnern in Berlin und Garching wurde für große Datenpakete eine Übertragungsrate von 30 MB/s als Obergrenze erzielt, zum VPP-System am LRZ eine Übertragungsrate von bis zu 22 MB/s.

Die Implementierungs- und Optimierungsarbeiten zur Verbindung der Endrechner mittels GTB waren bis Feb. 2000 praktisch abgeschlossen. Damit war die Grundlage geschaffen für detailliertere Analysen innerhalb der Metacomputing-Machbarkeitsstudie mit Hilfe der Anwendungen, für deren Einsatz bereits vorab Implementierungs- und Umstrukturierungsarbeiten erfolgt waren.

Die Studien wurden unterteilt nach homogenem Metacomputing, bei dem eine Anwendung auf gleichartige Rechner verteilt wird, und heterogenem Metacomputing, bei dem eine Anwendung auf unterschiedliche Rechnerarchitekturen verteilt wird.

Für homogenes Metacomputing wurden drei Anwendungen eingesetzt: Der Chemie-Code GAMESS (UK), der Astrophysik-Code PROMETHEUS und der ab-initio Molekulardynamik-Code CPMD. Es wurden Ablaufverhalten und Kommunikationsmuster bei simultaner Abarbeitung auf zwei gleichartigen Cray T3E-Rechnern untersucht. Die Unterschiede in der Leistungsfähigkeit von internem Netwerk der Hochleistungsrechner und externem Netzwerk (GTB) wurden hierbei deutlich. Für diese Studien wurde für die Rechnerkopplung ausschließlich die Kommunikationssoftware PACX-MPI eingesetzt und weiterentwickelt.

In GAMESS-UK wurden für die dynamische Lastbalancierung in der Metacomputing-Umgebung globale verteilte Taskzähler implementiert. Hierfür notwendige größere Erweiterung in PACX wurde vom RUS in enger Abstimmung mit dem ZIB vorgenommen. Im Vergleich zur lokal laufenden MPI-Version zeigte die PACX-Version in der Metacomputing-Umgebung für ein mittelgroßes Beispiel (ca. 400 Basisfunktionen) einen Leistungsabfall von etwa 8%. Als Vorbereitung für den Einbau der Global Arrays und damit des parallelen Eigensolvers in die Metacomputing-Version von GAMESS-UK wurde der Prototyp eines sogenannten ARMCI-Servers implementiert. Dieser erlaubt die Durchführung einseitiger asynchroner Speicherzugriffe über das WAN unter Nutzung der PACX-Bibliothek.

Mit PROMETHEUS wurden verteilte Läufe auf den T3Es in Berlin und Garching durchgeführt und das Kommunikationsschema detailliert analysiert. Bei einem Einsatz von zweimal 64 Prozessoren für gleich große Rechengebiete wurde bei der Simulation einer thermonuklearen Brennfront in einem Gas aus zwei Komponenten bei einer Gittergröße von 50 x 50 x 100 bei der Berechnung eines Zeitschritts 90% für die Kommunikation über das GTB aufgewendet. Auf einem einzigen T3E-System werden für lokale Kommunikation nur 29% der Zeit für einen Zeitschritt aufgewendet. Die Kommunikationszeit für verteilte Rechnung über das GTB lag um den Faktor 37 über der Kommunikationszeit für lokale Rechnung. Günstigere Situationen ergeben sich durch Erhöhung der Gesamtzahl der Gitterzellen, wie es z.B. für Berechnungen von nur Typ 1A Supernovae der Fall ist, oder bei Supernova-Simulationen mit komplizierterer und damit rechenaufwändigerer lokaler Physik.

Beim Einsatz von CPMD stieg die für die verteilte Rechnung über das GTB benötigte Rechenzeit zunächst um den Faktor 11, um den Faktor 8 nach Optimierung. Durch Anwendung des Gruppenkonzeptes zur Pfadintegralmethode konnte jedoch die Kommunikation zwischen Gruppen so deutlich reduziert werden, dass sich damit der Gesamtcode mit allen Programmteilen effizient auf zwei gleichartigen Parallelrechnern abarbeiten lässt, wenn ein externes Netzwerk mit den Leistungsmerkmalen des Gigabit-Testbeds und damit des G-WiNs verfügbar ist.

Für heterogenes Metacomputing wurde das Fujitsu VPP-Systeme am LRZ mit dem Cray T3E-System am RZG gekoppelt. Zwei gänzlich unterschiedliche Ansätze wurden verfolgt.

Im einen Ansatz wurde mit dem CPMD-Code, der sich bei homogenem Metacomputing als hervorragend geeignet bewährt hatte, versucht, alle Programmteile auf den beiden Rechnerarchitekturen mit Hilfe der PACX-MPI Kommunikationsbibliothek verteilt ablaufen zu lassen. Dieser Ansatz stellte sich als erheblich komplizierter als der homogene Ansatz heraus und war mit CPMD letztlich nicht durchführbar aufgrund der in CPMD gewählten Kommunikationsform und der unterschiedlichen Implementierung und Repräsentation von Datentypen auf den beiden verschiedenen Rechnerarchitekturen.

Im anderen Ansatz wurde versucht, den PRISM-Code, der aus zwei funktionell verschiedenen Teilen besteht, auf die beiden verschiedenen Rechnerarchitekturen zu verteilen.
Für den gut vektorisierenden Programmteil in PRISM mit in sich abgeschlossener Funktion wurde der VPP-Vektorrechner am LRZ eingesetzt, für den hoch parallelisierbaren Anteil mit ebenfalls in sich abgeschlossener Funktion das T3E-System am RZG. Die erforderliche Kommunikation zwischen diesen Programmteilen wurde über eine eigene TCP/IP-Schnittstelle realisiert. PRISM zählt als vorteilhaftes Beispiel für heterogenes Metacomputing unter Einsatz der für die entsprechenden Programmteile jeweils optimalen Rechnerachitekturen.

5. Diskussion

Unter dem Begriff des Metacomputings werden mehrere Varianten der Rechnerkopplung zusammengefasst. Neben der allgemeinsten Variante, dem lose gekoppelten Rechnerverbund, gibt es die Variante des *Visual Supercomputing*s, bei der in der Regel eine *online* Visualisierung von Supercomputerergebnissen mit meist hohen Anforderungen an die Bandbreite des Verbindungsnetzwerkes erfolgt. Bei der in dieser Studie durchgeführten engen Kopplung von Hochleistungsrechnern für eine einzige Anwendung handelt es sich um die Metacomputing-Variante mit den höchsten Anforderungen an das Verbindungsnetzwerk, da nicht nur Bandbreiten, sondern insbesondere auch Latenzzeiten eine wesentliche Rolle spielen.

Mit den eingesetzten Anwendungen aus der Chemie, der Materialforschung, der Astrophysik und der Polymerphysik, die unterschiedliche Kommunikationsanforderungen stellen, wurden verschiedene Szenarien erprobt und Problemzonen beleuchtet.

Mit keiner der Anwendungen war in der Ursprungsform, in der sie auf einzelnen Hochleistungsrechnern normalerweise eingesetzt werden, Metacomputing unter vernünftigen Leistungsgesichtspunkten möglich. Es wurde eine besondere Einsatzform der Codes benötigt, um den Kommunikationsbedarf gegenüber der Originalform zu reduzieren.

Bei der Chemieanwendung GAMESS wurden globale Zeiger, das Global Array Toolkit sowie Doppelhaltung von Daten eingeführt. Bei PROMETHEUS ist durch geeignete, ausgewählte Problembeispiele sehr stark auf das Verhältnis von Kommunikation zu Rechnung zu achten. Bei CPMD musste die Gruppenbildung über die Pfadintegralmethode eingesetzt werden. Unter diesen Randbedingungen können die genannten Anwendungen als prinzipiell tauglich für homogenes Metacomputing mit Showcases betrachtet werden.
Als vorteilhaft muss bei homogenem Metacomputing bezeichnet werden, dass nicht nur homogene Rechnerarchitekturen vorliegen, sondern prinzipiell auch homogene oder homogen gestaltbare Softwareumgebungen.

Beim heterogenen Metacomputing hingegen ist dies nicht der Fall, was in der Regel zu höherem Programmier- und Handhabungsaufwand führt. Im Fall von CPMD, mit dem der Versuch der Abarbeitung aller Programmteile gleichermaßen auf zwei grundverschiedenen Rechnerarchitekturen (Fujitsu VPP und Cray T3E) unternommen wurde, wäre beträchtlicher Umstrukturierungsaufwand nötig geworden.
Bei dieser Art des heterogenen Metacomputings kommen weitere Nachteile neben evtl. stark unterschiedlicher Softwareumgebung zum Tragen: Das zusätzliche Erfordernis für Loadbalancing zwischen den beiden Rechnerarchitekturen, da meist unterschiedlich leistungsfähige Prozessoren mit unterschiedlicher Hauptspeicherausstattung angetroffen werden.
Die Vorzüge des heterogenen Metacomputings liegen vielmehr bei einer Einsatzart, wie sie mit PRISM demonstriert werden konnte: nicht alle Programmteile (wie bei CPMD), sondern nur jeweils einer für eine bestimmte Architektur besonders geeigneter Programmteil wird auf die entsprechende Maschine verlagert.

Wenn man die in diesem Projekt durchgeführten Metacomputingszenarien unter einem Routinegesichtspunkt betrachtet, so bestehen noch erhebliche Defizite im Softwarebereich.

Nachdem beteiligte Hochleistungsrechner in der Regel immer ausgelastet sind, war immer ein koordinierter Administrator-Eingriff nötig, um die erforderlichen Ressourcen synchron bereitzustellen. Erforderlich wären geeignete Schnittstellen zu den Batchsystemen und ein Mechanismus zur Synchronisation derselben. Neue Entwicklungen wie im UNICORE- oder GLOBUS-Projekt sehen zwar die elementar erforderlichen Schnittstellen zu den lokalen Batchsystemen vor, eine Synchronisation derselben ist jedoch nicht vorgesehen.

Eine weitere für einen Routinebetrieb bestehende Problemzone ist die erforderliche Handhabung großer Datensätze für Output bzw Input oder Restart Files für Kettenjobs.
Metacomputing in der hier vorgestellten Art scheint damit noch weit von einem routinemäßigen Einsatz entfernt zu sein.

Das Projekt hat als Testbed eine Reihe von Erfahrungen ermöglicht, die in Zukunft die Arbeit mit schnellen Netzen leichter machen werden. Grundsätzlich ist das Projekt auch deshalb für alle Beteiligten positiv einzustufen, weil im Bereich der verteilten Simulation wertvolle Erfahrungen gesammelt werden konnten.

Das Projekt hat die Notwendigkeit der Entwicklung von Middleware deutlich gemacht. Hier sollte in Zukunft auch auf die Erfahrungen von UNICORE (www.unicore.de), GLOBUS (www.globus.org, ANL), LEGION (www.cs.virginia.edu/~legion, University of Virginia) und des Information Power Grid (www.ipg.nasa.gov, NASA) zurückgegriffen werden.

Auf der Anwendungsseite zeigt sich, dass bei Kommunikation über WAN sinnvolle Einsatzmöglichkeiten vor allem im Bereich lose gekoppelter Anwendungen bestehen. Der Einsatz eng gekoppelter Anwendungen auf verteilten Rechnerressourcen hingegen scheint derzeit einzig in LAN-Umgebungen sinnvoll.

Zum Gigabit-Testbed wäre anzumerken, dass für die effiziente End-zu-End Verbindung der Hochleistungsrechner auf der Basis der ATM-Technologie ein erheblicher Aufwand nötig war, das inzwischen eingeführte G-WiN jedoch auf einer anderen Technologie basiert.
Die mit den Anwendungen über IP gemachten Erfahrungen sind hingegen auf das neue G-WiN übertragbar.

Abschließend sei bemerkt, dass das Projekt mit einer gemeinsamen Zielsetzung und auch einem enormen Abstimmungsbedarf die Zusammenarbeit zwischen allen beteiligten Zentren erfreulich gefördert hat. Alle Beteiligten waren sehr bemüht, den anspruchsvollen Projektumfang in der doch knapp bemessenen Zeit durchzuführen. Das vorliegende Projekt wurde von sieben unmittelbar partizipierenden Partnern (LRZ, MPA, MFK, MPIP, RUS, RZG, ZIB) durchgeführt, weitere externe Partner waren involviert (EANTC, TU Berlin, DFN-Verein).

Danksagung

Die Projektbeteiligten danken dem DFN-Verein und dem Bundesministerium für Bildung und Forschung für die gewährte Unterstützung für dieses Projekt.

Anhang: Vorarbeiten zur HIPPI-zu-ATM-Umsetzung

HIPPI to ATM/OC-12c IP Router based on Sun Ultra 60

Peter W. Haas, RUS. March 3, 1999.

Dual 360-MHz Ultra SPARC processors
512 Mbyte DRAM
Essential/ODS PCI Serial HIPPI, Sun HIPPI Beta 1.0 Driver
FORE Systems HE622 PCI ATM/OC-12c, solaris-HE_5.0.0_20494.tar Driver

Test Setup:

```
NSC       ODS     Sun       FORE                    FORE     Sun
SGI/DM-2 - HIPPI <->  PS32 <-> HIPPI - Ultra 60 - HE622 <-> ATM/OC-12c <-> HE622 - E-450
HIPPI           FORE IP                                 (agate2)
Crossbar          Router
(agate1)
```

```
#!/bin/sh
# agate1 FORE IP configuration
ifconfig fa0 192.168.155.5 mtu 65280 netmask 255.255.255.252 up
/opt/FOREatm/bin/atmarp -l fa0 0 100 5
/opt/FOREatm/bin/atmarp -s agate2 fa0 0 100 5
```

```
#!/bin/sh
# agate2 FORE IP configuration
ifconfig fa0 192.168.155.6 mtu 65280 netmask 255.255.255.252 up
/opt/FOREatm/bin/atmarp -l fa0 0 100 5
/opt/FOREatm/bin/atmarp -s agate1 fa0 0 100 5
```

Measurement of TCP/IP Memory-to-Memory Throughput
between SGI/DM-2 and Sun E-450

SGI_DM-2 <-> (Ultra60) <-> Sun_E-450
==========

Configuration for FORE IP, SPANS signalling

Sun Ultra 60 CPU load 10%

SGI/DM-2: TCP Segment 49152
Sun E-450: TCP Segment 65228, IP MTU 65280 (CPU load 63%)
TCP Socket Buffer 512k

comptsock: output is tsock throughput [Mbytes/s]
Filesizes: 1k 2k 4k 8k 16k 32k 64k 128k 256k 512k 1M 2M 4M
7.30309
10.91464
9.51491
87.89717
146.98764
261.31303

```
40.26519
28.93634
38.34050
38.63772
44.26959
51.85060
66.1713


SUN_E-450 <-> (Ultra60) <-> SGI_DM-2
====================================

Configuration for FORE IP, SPANS signalling

Sun Ultra 60 CPU load 10%

SGI/DM-2:  TCP Segment 49152
Sun E-450: TCP Segment 57304, IP MTU 65280 (CPU load 83%)
TCP Socket Buffer 512k

comptsock: output is tsock throughput [Mbytes/s]
Filesizes: 1k 2k 4k 8k 16k 32k 64k 128k 256k 512k 1M 2M 4M
11.52883
17.38354
26.29711
46.11792
72.57541
88.82023
13.1823
14.7775
25.0771
26.8987
48.0383
52.3565
57.7246


Measurement of UDP/IP Memory-to-Memory Throughput
between SGI/DM-2 and Sun E-450:
```

No datagram loss with 56-Kbyte UDP datagrams
up to full payload bandwidth of ATM/OC-12c.