Linux Cluster und Mosix

Bearbeitet von

Oliver Klima, Sebastian Roth und Norman-Timo Rehl

Im Rahmen der Vorlesung

Innovative Architekturen

An der Berufsakademie Mannheim (WS 2004).

1 Inhaltsangabe

2 Was ist ein Cluster?

Ein Computercluster, meist einfach Cluster, von engl. cluster = Traube, Bündel, Schwarm, genannt, bezeichnet eine Anzahl von vernetzten Computern, die zur parallelen Abarbeitung von zu einer Aufgabe gehörigen Teilaufgaben zur Verfügung stehen. Im Gegensatz zu Parallelrechnern findet die Lastverteilung auf der Ebene einzelner Prozesse statt, die auf einer oder verschiedenen Maschinen des Clusters gestartet werden. Man benötigt also keine parallelisierte Software oder spezielle Betriebssysteme, wohl aber einen Scheduler, der die Teilaufgaben den Einzelrechnern zuweist. Alternativ werden Cluster auch zum Steigern der Verfügbarkeit von Systemen genutzt. [1]

3 Warum Linux für einen Cluster?

Linux bietet viele Vorteile für den Einsatz in Clustern. Normalerweise ist ein Betriebssystem pro Arbeitsplatz zu lizenzieren, da Linux aber kostenfrei erhältlich ist, fallen Lizenzkosten weg. Wenn ein Cluster mehr als 10 herkömmliche PC´s enthält, können Lizenzkosten sehr schnell das finanzielle Aus bedeuten.

Weiterhin bietet Linux im Gegensatz zu anderen Betriebssystemen sehr variable Konfigurationsmöglichkeiten. So ist z.B. der Kernel von Linux frei dem Arbeitsplatz anpassbar. So bleibt der Betriebssystem Kern klein und effizient. Ein vergleichbares Betriebssystem von Microsoft® lässt in dieser Hinsicht keine Variabilität zu und ein Standard Kernel wird installiert. Standard Kernel sind grundsätzlich auf ein Mittelmaß abgestimmt, damit im Durchschnitt alle Anwendungen adäquat durchführbar sind. Der Linux Kernel lässt sich jedoch, wie oben erwähnt, auf die bestimmten Bedürfnisse anpassen, und diese Anwendung läuft im Vergleich dann schneller.

Gerade mit dem Betreiben von Clustern treten erhöhte Sicherheitsproblematiken auf. So darf ein immigrierter Prozess auf einem Arbeitsplatz nicht modifizierbar für den Benutzer sein. Deshalb ist es vorteilhaft auch in diesem Fall Linux zu verwenden. Linux bietet Sicherheitskontrollen für den Datei- und Prozesszugriff, die schon im

[1] Quelle: http://de.wikipedia.org/wiki/Computercluster

Kernel integriert sind. Daher ist es für einen lokalen Benutzer (fast) unmöglich auf immigrierte Prozesse zuzugreifen.

4 Cluster Methoden

4.1 Active – Active Cluster

Bei Active – Actice Clustern arbeitet jeder Knoten aktiv an einer Problemlösung mit. Es gibt keine Redundanzen, die die Korrektheit von Teillösungen verifizieren. Somit ist zwar eine bestmöglichste Performance aus dem Cluster zu holen, doch kann es auch unter Umständen sinnvoller sein Redundanzen einzuführen.

Bei Active – Active Clustern können trotzdem einzelne Knoten ausfallen ohne einen Systemabsturz hervorzurufen. Der unterbrochene Prozess wird einfach neu auf einen anderen Knoten ausgelagert, und die Teillösung dauert dann etwas länger.

Es können mit dieser Methode aber auch zur Laufzeit neue Knoten hinzugefügt werden. Die gesamte Rechenleistung steigt dann, weil bei einer neuen Verteilung von Prozessen auch die neuen Knoten mit berücksichtigt werden.

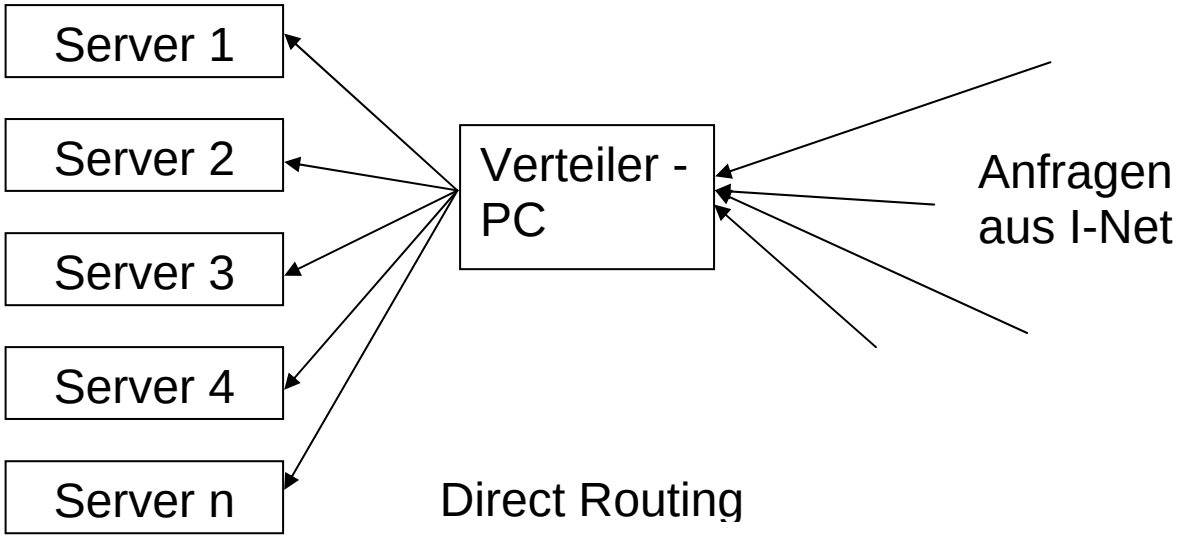
4.2 Active – Standby Cluster (Hot Standby)

Als Unterschied zu Active – Active Clustern können auch Knoten redundant ausgelegt sein. So ist in diesem Fall ein Knoten dabei die Teillösung zu errechnen, während mindestens ein weiterer zyklisch die Aktivität des aktiven Knotens überprüft. Sollte der aktive Teil ausfallen, so springt sofort ein redundanter Knoten ein. Dies erspart Management Arbeit, denn ein abgebrochener Prozess muss nicht neu vergeben werden, da sofort ein Knoten einspringt, der die abgebrochene Arbeit fortsetzt.

Im Detail können sich die Hot Standby Knoten in dem Punkt unterscheiden, dass redundante Knoten „mitrechen“ und anschließend Ergebnisse vergleichen, oder dass redundante Knoten nur die Aktivität des aktiven Parts überprüfen. Welche Strategie angewendet werden sollte, hängt von der Aufgabenstellung des Clusters ab. So sollten Bankkontotransaktionen eher redundant mit Ergebniskontrolle durchgeführt werden, während eine Datenbankapplikation keine Ergebniskontrolle benötigt. Eine Auswertung bei einem KFZ-Crashtest sollte nach diesem Prinzip Active – Active gestaltet werden, damit ein möglichst rasches Ergebnis produziert werden kann.

5 Cluster Typen:

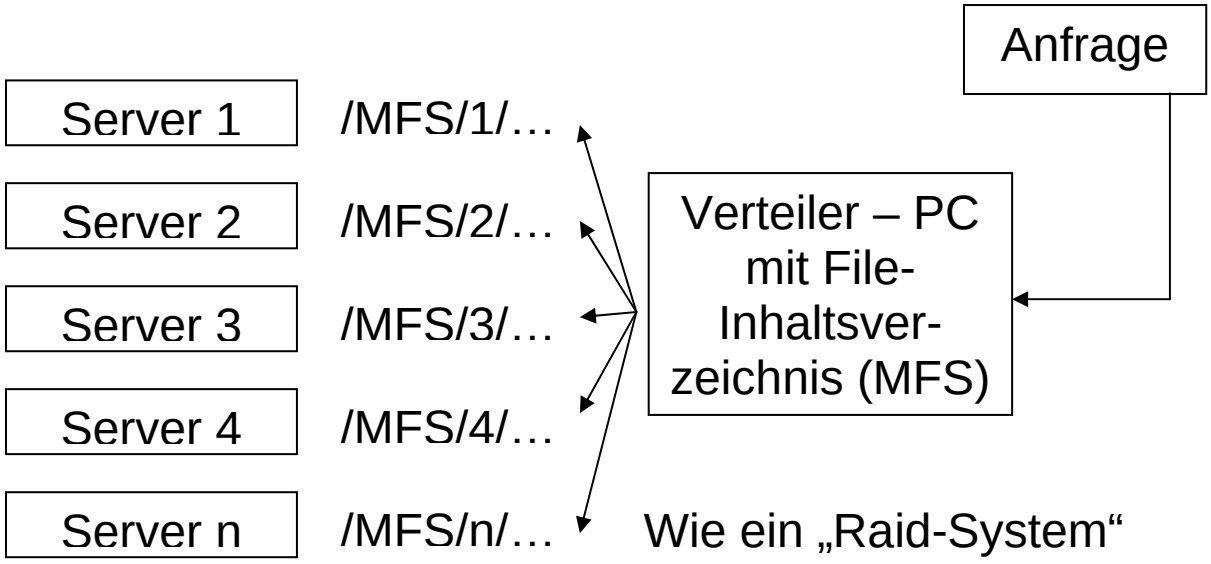
5.1 Web-Server-Cluster



Bei einem Web-Server-Cluster ist es wichtig, dass alle Anfragen aus dem Internet möglichst schnell behandelt werden. So kann man die Server redundant mit dem Inhalt der Webseiten ausstatten. Meist sind diese Informationen nicht zu groß, um diese Informationen lokal auf die Serverfestplatte zu speichern.

Die Server kommunizieren ausschließlich mit dem Verteiler – PC, der sich auch um die IP – Adressvergabe mittels „NetworkAdressTranslation“ kümmert. Dieser vorgang wird auch „Direct Routing“ bezeichnet. Es gibt mehrere Methoden die Auswahl auf die Server zu treffen. Einmal lässt sich die Kapazität der einzelnen Server als Kriterium wählen. D.h., dass bei Erreichen der möglichen Kapazität des ersten Servers automatisch alle zusätzlichen Internetanfragen auf den zweiten Server geschaltet werden. Aber auch ein „Round – Robin – Verfahren“ ist durchaus denkbar.

5.2 File-Server-Cluster



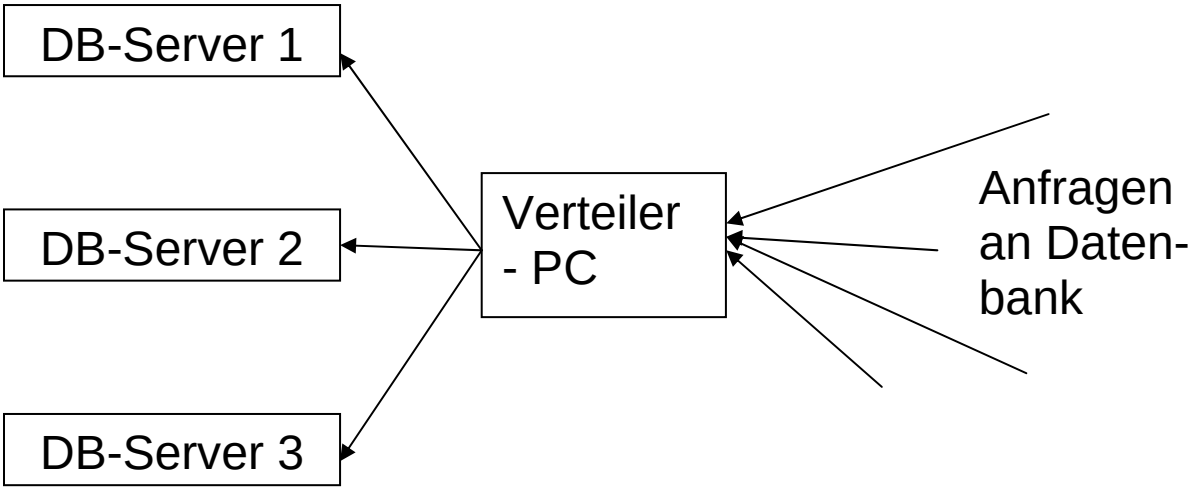
Bei einem File-Server-Cluster gibt es einen PC, der ein Inhaltsverzeichnis aller Dateien enthält. Bei einem MOSIX System bezeichnet man dieses File – System auch Mosix – File – System.

Der Verteiler – PC gibt nach außen immer die gesamte Dateiliste aus. Bei einer Anfrage auf eine bestimmte Datei hängt der Verteiler – PC vor den Pfad aus der Anfrage einfach ein „/MFS/n/“ (n steht für eine laufende Nummer der Server, wobei der Verteiler – PC aus seiner Dateiliste erkennt, wo welche Datei gespeichert ist).

Das ganze System funktioniert so wie ein bekanntes Raid – System. Der Anwender sieht in dieser Verfahrensweise nicht, auf welchem Server sich nun wirklich diese Datei befindet, denn es ist für Ihn auch uninteressant.

Mit diesem System lassen sich performante File – Server erstellen, die weit über irgendwelche Begrenzungen eines handelsüblichen Raid – Systems eines einzelnen Servers hinausragen. Auch hier können Redundanzen wie in einem Web – Server – Cluster erstellt werden.

5.3 Datenbank – Cluster



Der Datenbank – Cluster ist im Gegensatz zu den anderen beiden vorgestellten Cluster – Typen erheblich komplizierter. Datenbanken benutzen aus Performancegründen einen sehr großen Cache, der bei einem Cluster auf allen Server verteilt ist. Wenn eine Applikation Änderungen an der Datenbank vornehmen möchte, so müssen die Inhalte der Caches aller DB – Server überprüft werden, denn es besteht die Möglichkeit, dass auf einem anderen Cache eine zuvor getätigte Transaktion auf das Datenelement durchgeführt wurde, aber noch nicht physikalisch auf die Festplatte geschrieben wurde.

Somit gilt das oberste Prinzip, dass alle Caches synchronisiert werden müssen. Wenn dies realisiert werden kann, obgleich vom Datenbankmanagementsystem oder vom Verteiler – PC, dann kann erst von einem konsistenten DB – Cluster gesprochen werden.

5.4 Zusammenführung aller Typen

Alle drei vorgestellten Cluster – Typen lassen sich wiederum zu einem (drei) Cluster zusammenfügen. So ist es z.B. denkbar, dass bei einem angebotenem Web – Dienst ein Web – Cluster aufgestellt wird, der zur Datenbewältigung eine Datenbank betreibt, die wiederum auf einem DB – Cluster läuft. Der DB – Cluster benötigt aber viel Speicherplatz auf der Festplatte, also schließt man diesem ein File – Cluster an. Somit lassen sich Cluster zu einem riesigen Rechenwerk zusammenschließen, bei dem andere einzelne Server oder Supercomputer das Nachsehen haben.

6 Single System Image

Bei Clustern dieses Typs steht die Eigenschaft, dass Cluster dem Benutzer als ein geschlossenes System respektive ein Rechner erscheint, im Vordergrund. So soll es dem Anwender möglich sein, alle Ressourcen des Clusters transparent zu nutzen. Somit stehen ihm bedeutend mehr Rechenleistung, Speicher, etc. zur Verfügung, als dies mit einem einzelnen Rechner in einer vergleichbaren Preisklasse möglich wäre.
Um dies zu erreichen, ist der größte Teil der Funktionalität eines solchen Clusters meist in einem modifizierten Linux-Kernel implementiert, so dass keine oder nur wenige Veränderungen an Anwendungen notwendig sind, um von den Vorteilen eines Single System Images zu profitieren. Änderungen sind hierbei allein in den Grenzen der SSI-Produkte begründet, die allerdings immer wieder zu Gunsten der Anwender verschoben werden können. Neben den Modifizierungen des Kernels, stehen jeweils noch einige Anwendungen zur Verfügung, die der Konfiguration, Überwachung und Steuerung des Clusters dienen.
Ein weiteres Merkmal von Single System Images ist es, dass alle Knoten gleichberechtigt sind und somit auch alle verfügbaren Ressourcen nutzen können. Einer der wichtigsten Bestandteile eines SSIs ist jedoch das Load Balancing zusammen mit der Prozess Migration. Durch Load Balancing versucht das System die Last der einzelnen Knoten gleichmäßig zu verteilen. So wird bei hoher Last eines Knotens versucht, einen oder mehrere Prozesse auf andere Knoten auszulagern. Hierbei spielt die CPU-Leistung eine entscheidende Rolle, andere Faktoren, wie zum Beispiel der verfügbare Arbeitsspeicher, werden jedoch auch berücksichtigt. Um dies möglichst effizient zu erreichen wurden im Laufe der Zeit verschiedene Verfahren und Algorithmen entwickelt, doch es existieren noch immer Grenzfälle in denen Prozesse nicht migrieren. Ein Beispiel hierfür sind kurzlebige Prozesse, wie sie unter anderem beim Kompilieren meist der Fall sind.
Ein weiterer Bestandteil eines Single System Images ist meist ein spezielles Cluster File System, das es den Knoten erlaubt, relativ einfach und effizient auf Daten eines anderen Knotens zuzugreifen. Dabei handelt es sich nicht um ein Dateisystem, wie zum Beispiel ext2, reiserfs oder fat, sondern es baut auf einem solchen auf und regelt den Zugriff über das Netzwerk.
Aufgrund der wenigen Einschränkungen und Vorraussetzungen eines Single System Images, können und werden sie für eine Vielzahl von Programmen verwendet wer-

den, sowohl im Bereich des High Performance Computing, als auch im Bereich der High Availability.

6.1 openMosix

openMosix ist eine Single System Image Implementierung, die vor allem für Anwendungen des High Performance Computing gedacht ist. Es ermöglicht das Hinzufügen und auch Entfernen von Knoten während des Betriebes ohne weitere Konfiguration, weshalb der Cluster bei Engpässen ohne Probleme erweitert werden kann. Zu dem ist es fast linear skalierbar, die Netzwerklast für das Management des Clusters soll 2% nicht übersteigen.
Dabei beinhaltet openMosix lediglich einen Patch für einen Standard Linux-Kernel, sowie einige Programme zur Administration.

6.1.1 Geschichte

Die Geschichte von openMosix schließt automatisch die des Vorgängers und nun Konkurrenten Mosix (Multicomputer Operating System for UnIX.) mit ein. Dessen Geschichte reicht bis in das Jahr 1977 zurück, als mit der Entwicklung des „UNIX with Satellite Processors“ begonnen wurde. Dies geschah unter Leitung von Professor Amnon Barak an der Universität von Jerusalem und wurde 1979 abgeschlossen. Die Entwicklung basierte auf Bell Lab's Unix 6 und lief auf PDP-11/45 und PDP-11/10 Systemen.
Nachdem mit der Zeit Linux an Bedeutung gewann, erfolgte 1999 die Portierung hierfür, während die Unterstützung von BSD endete. Anfang 2002 startete Moshe Bar das openMosix-Projekt, da es zu Uneinigkeiten über die Lizenzierung von Mosix gekommen war. So wird openMosix unter der GPL veröffentlicht, Mosix hingegen unter einer eigenen, mit stärkeren Einschränkungen.

6.1.2 Gegenwart

Zur Zeit (Ende 2004) existiert ein openMosix-Patch für den Kernel der Version 2.4.24, wobei nach Erstveröffentlichung eine zweite Version mit einigen Bugfixes herausgebracht wurde.
Die dazugehörigen Anwendungen liegen in der Version 0.3.6-2 vor und beinhalten unter anderem omdiscd, ein Daemon, der für die Verwaltung des Clusters verant-

wortlich ist und mosmon, ein wenn auch recht schlichtes Programm zur Überwachung der Prozessorauslastung aller Knoten.
Zu dem gibt es jedoch auch noch weitere Programme, hauptsächlich zur Verwaltung von openMosix-Clustern, die allerdings von anderen Entwickler-Teams betreut werden. Dazu gehören zum Beispiel openMosixview, das vor allem eine graphische Benutzeroberfläche mit einschließt und gomd (general openMosix daemon).
Als Cluster File System kommt zur Zeit oMFS (openMosix File System) mit DFSA (Direct File System Access) als Erweiterung zum Einsatz. Dabei ähnelt das openMosix File System recht stark dem NFS, bietet jedoch noch einige Zusatzfeatures, wie die Konsistenzsicherung bei konkurrierenden Zugriffen. DFSA ermöglicht es darüber hinaus, dass Knoten auf anderen Knoten Systemaufrufe machen können, ohne komplett dorthin migrieren zu müssen.

6.1.3 Zukunft

Ein gesetztes Ziel des Entwickler-Teams von openMosix ist es, für die aktuelle 2.6er Kernel-Serie ebenfalls einen Patch zur Verfügung zu stellen. Eine Vorabversion davon existiert bereits, welche jedoch noch nicht einsatzbereit ist, da zuviel der Funktionalität noch nicht implementiert ist. Zu dem ist die Portierung für die 64 Bit Intel and AMD Architekturen in Arbeit respektive Planung.
Weiterhin soll das oMFS durch PVFS (Parallel Virtual File System) ersetzt werden, das von einer separatem Projekt-Gruppe entwickelt wurde.
Außerdem will man sich darum bemühen, so viel wie möglich von der Implementierung vom Kernel in Programme zu verlagern, wodurch zum Einen der sicherheitsrelevante Code-Anteil geringer wird und zum Anderen die Portierung auf andere Systeme vereinfacht wird.

6.1.4 Anwendungen

Wenngleich fast ausnahmslos alle Anwendungen auf einem openMosix-Cluster funktionieren, migrieren nicht alle. Die größten Probleme bereiten hierbei Programme, die Shared Memory oder Threads verwenden. Für erstere existiert allerdings bereits eine Lösung in Form von migshm (Migration of shared memory), ein Patch für openMosix. Zu den Anwendungen, die davon profitieren, gehören unter anderem Apache, mySQL, SAP und Oracle. Bei Multithreading-Programmen gibt es hingegen nur eine Ausnahme. Java-Anwendungen, die in einer Green Threads JVM

laufen, können ebenfalls migrieren, da die Threads durch eigenständige Prozesse realisiert werden.
Unter den Programmen, die ohne Probleme migrieren, finden sich einige Audio- und Video-En- respektive Decoder, wie zum Beispiel MJPEG Tools, bladeenc, LAME und mpg321. Auch viele andere Anwendungen, wie Postfix, SpamAssassin, make und TightVNC, migrieren. Selbst Programme die MPI verwenden, profitieren von openMosix. So genügt es auf einem Knoten mehrere Prozesse zu starten und deren Migration dem System zu überlassen. Ein weiterer Vorteil, wie auch bei Single System Images allgemein, ist es, dass die Anwendung oder wie im Fall von MPI die Libraries nur auf einem Rechner installiert werden müssen.
Um den Einstieg zu erleichtern und die Möglichkeit zu bieten, einen openMosix-Cluster mit relativ wenig Aufwand testen zu können, existieren mehrere Live-CDs und auch Boot-Floppies, die einen entsprechenden Kernel und die notwendigen Programme enthalten. Dazu gehören ClusterKnoppix, PlumpOS, openMosixLOAF und GoMF.

6.2 OpenSSI

Eine andere Single System Image Implementierung, OpenSSI, wird oft als direkter Konkurrent zu openMosix genannt, wenngleich der Schwerpunkt hier bei Anwendungen im Bereich der High Availability liegt.
Dies begründet auch, warum OpenSSI Techniken wie das HA Cluster File System, HA Linux Virtual Server und HA interconnect verwendet. Zu dem können bei einem Ausfall eines Knotens darauf laufende Anwendungen auf einem anderen Knoten automatisch wieder gestartet werden.
Im Gegensatz zu openMosix sind alle Prozesse, Devices und IPC-Objekte für alle Knoten sichtbar und auch verfügbar. Bei openMosix kann ein Knoten nur auf die lokalen und die von ihm aus migrierten Prozesse zugreifen.
Im Bereich des Load Balancing baut die Implementierung von OpenSSI allerdings auf der von openMosix auf, weshalb auch die gleichen Einschränkungen bezüglich Programmen, die Threads verwenden, gelten.

7 Massiv parallele Systeme

Als massiv parallele Systeme bezeichnet man Rechner, die bereits von ihrem internen Hardwareaufbau einen massiven Grad an Parallelität bereitstellen. Das Hauptaugenmerk solcher Systeme ist es, eine größtmögliche Performance zu erzielen und grenzen sich somit von anderen bekannten Systemen ab. Bei diesen Anlagen handelt es sich überwiegend um so genannte Supercomputer, die aufgrund ihrer preislichen Gestaltung nur von größeren Unternehmen oder Organisationen angeschafft und betrieben werden können. Um auch eine Möglichkeit zu schaffen, durch Ausschöpfung von vorhandenen Ressourcen, eine preisgünstige Alternative zu erhalten, müssen einige Faktoren berücksichtigt werden.

7.1 Werkzeuge zur Parallelisierung

Um das gewünschte Ergebnis zu erreichen, war es notwendig, Schnittstellen bereit zu stellen, die den Datenaustausch mehrerer Rechner automatisiert, um dadurch einen oder mehrere Prozess zu parallelisieren und somit zu beschleunigen. Da auf Hardwareebene kein gemeinsamer Speicher mehr vorhanden ist, bietet es sich an, mit Nachrichtenaustausch (*message passing*) zu arbeiten.

7.1.1 Message Passing Interface

MPI ist eine Programmierbibliothek für C / C++, Fortran und Java. An der Entwicklung und Standardisierung sind über 40 Institutionen aus Staat, Hochschulen und Industrie (darunter führende Unternehmen der Computerbranche wie z.B. IBM, SUN, Cray), in erster Linie aus den USA und Europa beteiligt. Ziel ist es, eine einheitliche, portable Programmierschnittstelle zu schaffen, die es Programmen erlaubt Nachrichten auszutauschen. Dabei soll es keinen Unterschied machen, ob es sich um eine Architektur von Prozessoren mit gemeinsam genutztem Speicher handelt (Mehrprozessormaschinen) oder ob jeder Prozessor über seinen eigenen Arbeitsspeicher verfügt (vernetzte Einprozessoranlagen). Auch Mischformen sollen möglich sein.

Das Ziel von MPI war es eine leistungsfähige auf Message Passing basierte Software zu entwickeln, die ausschließlich auf Schnelligkeit ausgerichtet sein sollte. Da die Entwickler aus der Industrie verständlicherweise kein Interesse daran hatten, das ihre Cluster mit denen von z. B. anderen Hardwarefirmen zusammenarbeiteten, gibt

es in MPI keine Funktionen, die ein gemischtes Netzwerk z. B. durch die Umsetzung verschiedener Netzwerkprotokolle ermöglichen. MPI verfügt aber über eine große Auswahl an Kommunikationsroutinen, die es ermöglichen mit einem Befehl mit einer ganzen Gruppe von Prozessen zu kommunizieren.[2]

MPI besitzt ein statisches Prozessmodell. D.h. alle Prozesse, die in einer MPI-Laufzeitumgebung existieren, betreten und verlassen die Umgebung quasi gleichzeitig. Innerhalb der laufenden Umgebung existieren die Prozesse in Gruppen, die die Kommunikationsfähigkeit eines Prozesses innerhalb dieser Gruppen regeln. Die eigentliche Kommunikation in MPI erfolgt durch Zugriffe auf Kommunikatoren, die durch Gruppen definierten Kommunikationskanäle. Zudem ist der Rang eines Prozesses innerhalb der Gruppen für alle Mitglieder der Gruppe gleich (globale Sicht). MPI unterstützt Threads nicht explizit, ist jedoch threadsafe konstruiert, d.h. die Verwendung von Threads führt nicht zu abnormalen Zuständen.

Es gibt mehrere Implementationen des MPI. Die meisten davon stehen unter kostenpflichtigen Lizenzen, es gibt allerdings auch einige frei erhältliche Versionen. Dies sind unter anderem LAM (Local Area Multicomputer) und MPICH, beide Bestandteil der SuSE-Linux Distribution. Die aktuellste Version ist MPI 2.0, die über Message Passing hinausgeht und auch Shared- Memory- Operationen zulässt.

7.1.2 Parallel Virtual Machine

Bei PVM handelt es sich um ein frei verfügbares Softwarepaket, das es ermöglicht, ein heterogenes Netz von Computern als einen zusammenhängenden virtuellen Parallelrechner erscheinen zu lassen. Auf diese Weise können große und komplexe Probleme mit Hilfe von bestehenden Ressourcen berechnet werden. Die Entwicklung des PVM- Pakets begann 1989 am Oak Rich National Laboratory (ORNL) und wird heutzutage weltweit weitergeführt.
Das PVM- Paket ist für alle Unix- und Windows-Systeme erhältlich und unterstützt vom Benutzer ausgewählte Vektorcomputer, Parallelrechner, Grafikcomputer, Workstations und PCs. Diese, in PVM Hosts genannten Computer, erscheinen, durch ein Netzwerk verbunden, als ein großer virtueller Parallelrechner, der komplexe Berech-

[2] http://parallel.fh-bielefeld.de/pv/studien/pvm/71PVMvsMPI.html

nungen schneller ausführen kann als ein sequentieller Rechner. PVM unterstützt alle TCP-basierten Rechnernetze wie FDDI, ATM, Ethernet, etc. und enthält C, C++ und Fortran Bibliotheken.

Ein Task ist in PVM, in Analogie zu einem Unixprozeß, eine Berechnungseinheit im Programm. PVM bietet den Tasks die Möglichkeit der Kommunikation und Synchronisation, d.h. sie können miteinander kommunizieren und vor kritischen Programmabschnitten synchronisiert werden. So können die Tasks miteinander kooperieren und Anwendungen parallel ausführen.

Die Heterogenität wird auf der Anwendungs-, Maschinen- und Netzwerkebene unterstützt, d.h. PVM gibt den einzelnen Tasks die Möglichkeit die verschiedenen Architekturen optimal auszunutzen und kümmert sich gänzlich um eventuelle Datenkonvertierung zwischen zwei kommunizierenden Rechnern, wenn diese unterschiedliche Darstellungsformate der Daten benutzen. Zusätzlich kann die virtuelle Maschine durch eine Vielzahl verschiedener Netze verbunden sein.[3]

7.1.3 Vergleich zwischen MPI und PVM

Der Vergleich zwischen PVM und MPI liegt nahe, weil beide dem selben Zweck dienen: Programmierung portabler paralleler Programme.

Eigenschaft	PVM	MPI
Portabilität	+	+
Heterogenität	+	-
Performance	+ -	++
Fehlertoleranz	+	-
Resourcenkontrolle	+	-
Topologien	-	+
Sichere Kommunikation	-	+

Übersicht der PVM- und MPI eigenschaften

PVM und MPI unterscheiden sich bereits in der Zielsetzung. PVM wurde entwickelt, um mehrere Rechner zu einer virtuellen Maschine zusammenzuschließen und sie als einen einzigen Parallelrechner zu behandeln. Dieser Zusammenschluss kann Workstations und Parallelrechner genauso enthalten wie Personal Computer, die mit Windows oder Linux betrieben werden.

[3] http://www.syssoft.uni-trier.de/systemsoftware/Download/Seminare/Middleware/middleware.2.book.html

Die MPI-Entwickler hingegen wollten ein erweiterbares Konzept schaffen. Daher entschieden sie sich dafür, Informationen an das System übergeben. Dies schränkt zwar die Portabilität ein, weil diese Informationen implementationsabhängig sind, ermöglicht aber zusätzlich zur Erweiterbarkeit auch die Nutzung spezieller Kenntnisse des Programmierers bezüglich des Systems.

7.2 Beowulf- Cluster

Mit den Beowulf-Clustern[4] entstand eine neue Klasse von Systemen, bei denen die Workstations intern vernetzt waren. Dadurch beschleunigte sich die interne Kommunikation. Im Sommer 1994 bauten Thomas Sterling und Don Becker aus Massenkomponenten den ersten Beowulf-Cluster - COTS (Commodity off the Shelf). Sie vernetzten 16 Intel 80486 (100 MHz, 16 MByte RAM) mit einem dualen 10-MBit/s-Ethernet zu einem Rechner.
Beowulf-Cluster sind Cluster aus Workstations und bestehen aus Knoten, die nur zum Cluster gehören. Die Knoten können einen oder mehrere Prozessoren enthalten, die sich einen gemeinsamen Speicher teilen. Die Knoten sind über ein internes Netzwerk verbunden und können daher schneller kommunizieren. Als Software werden offene und Hardware-unabhängige Bausteine verwendet, Linux, GNU-Compiler, Werkzeuge und Bibliotheken wie PVM und MPI. Auf diesem Prinzip beruhen heute viele Cluster mit unterschiedlichen Prozessorarchitekturen. Inzwischen entwickelt und unterstützt das Unternehmen Scyld Computing Corporation Hochleistungslösungen, die auf offenen Systemen wie dem Beowulf-Cluster beruhen.[4]

Beowulf-Cluster haben sich zu einer interessanten Alternative zu den kommerziell verfügbaren Rechnern entwickelt. Sie besitzen ein hervorragendes Preis-Leistungs-Verhältnis, skalieren auf Hunderte von Knoten und schließlich ist für sie eine breite Softwarepalette von Open- Source- Software verfügbar. Damit eignen sie sich für Forschungseinrichtungen und Hochschulen optimal, die eigenes Personal für Anpassungen einsetzen können.

[4] www.linux-magazin.de/Artikel/ ausgabe/2002/05/sce/sce.html

7.3 Andere Cluster- Betriebssysteme

Die Verwendung von Cluster- Systemen dient im Allgemeinen jeweils für verschiedene, spezielle Anforderungen.

Aufgaben:	Betriebssystem:
Manuelle Prozeßverteilung	openPBS, PBS Pro
Distributed Shared Memory	Plurix (gemeinsamer Adressraum im gesamten Cluster
Web Servicing	LVS, Piranha
Storage	openGFS, Lustre
Database	Oracle RAC, IBM ICE
High Availability	LifeKeeper, HA Linux
Beowulf	openSCE, ROCKS

Da es nicht möglich ist, alle positiven Merkmale und Eigenschaften der verschiedenen Systeme in einem Betriebssystem zu sammeln, sind die Hersteller gezwungen, für ihre bestimmten Aufgaben, eigens Betriebssysteme zu entwickeln, die nur den gewünschten Zweck erfüllen. So gibt es eine Vielzahl an Betriebssystemen, die einen bestimmten Dienst erweisen. Die abgebildete Tabelle zeigt möglich Aufgaben und die dazugehörigen Systeme. Darüber hinaus gibt es mannigfach verschiedene Betriebssysteme, die ganz bestimmten Anforderungen nachgehen.