

Spezielle Techniken der Rechnerkommunikation

Clusternetzwerke

Robert Hilbrich
Daniel Reinhold

17. Juli 2006



Inhaltsverzeichnis

1 Einführung	3
2 Was ist überhaupt ein Cluster?	4
3 Cluster Hardware und Architektur	6
3.1 High Performance, High Throughput and High Availability	6
3.2 Beispiele für die verschiedenen Clustertypen	6
3.3 Bestandteile von Computerclustern	7
4 Grundlagen der Netzwerke in Clustern	9
4.1 Management- und Controlnetzwerke	9
4.2 Datennetzwerke	9
4.3 HSI - Highspeed Interconnect	10
5 Betriebssysteme für Cluster	13
6 Cluster Middleware	15
7 OpenSource Cluster	17
7.1 OSCAR	17
7.2 Beowulf	18
8 Cluster Demonstration	20
8.1 High Availability mit Heartbeat	20
8.2 CLAN Cluster am CMS	23
9 Ausblick - vom Cluster zum Grid	27

1 Einführung

Die vorliegende Semesterarbeit entstand im Rahmen der Vorlesung “Spezielle Techniken der Rechnerkommunikation” von Dr. Sommer an der Humboldt Universität in Berlin. Mit einem besonderen Fokus auf die Kommunikation zwischen Rechnern werden im folgenden *Clusternetzwerke* betrachtet und in ihrer Schichtenarchitektur beleuchtet.

In Anbetracht dieser Schwerpunktsetzung werden softwarespezifische Gesichtspunkte nicht speziell betrachtet, sondern eher in den Hintergrund rücken.

Die Präsentation der Ergebnisse dieser Arbeit beinhaltet auch einen Vortrag mit einer Demonstration aus dem Bereich *High Availability* bzw. *Webserver Failover Management*.

2 Was ist überhaupt ein Cluster?

In Anlehnung an den englischen Begriff *Cluster* ist es einem relativ schnell möglich eine intuitive Ahnung über den Hintergrund eines *Clusternetzwerks* oder kurz *Clusters* zu haben. So ist man sicher nicht überrascht, dass sich der Begriff auf eine Gruppe von vernetzten Computern bezieht, die irgendwie untereinander vernetzt sind und logisch gesehen, in einer irgendwie gearteten Relation zueinander stehen.

Rein physikalisch gesehen, können Cluster jedoch völlig unterschiedliche Ausmaße besitzen und in ganz verschiedenen Größenordnungen existieren.

Klassisch stellen die riesigen *Rechnerfarmen* des Europäischen Zentrums für Nuklearforschung (*Conseil Européen pour la Recherche Nucléaire*, kurz *CERN*) oder der *National Security Agency*, kurz *NSA* einen Cluster dar, um die gigantischen Datenmengen zu möglichst effizient verarbeiten zu können.

Im Zuge dramatisch sinkender Hardwarekosten sind Clustertechnologien aber mittlerweile nicht mehr nur dem finanzstarken Anwenderkreis zugänglich, so dass diese Technologie in Zukunft sicher auch im Bereich der Klein- und Mittelständischen Unternehmen (*KMU*) für den Ersatz der bis dahin zumeist eingesetzten *Uniprozessor*- bzw. *Symmetric Multiprozessor Systeme* interessant wird.

Betrachtet man die technische Entwicklung zur Befriedigung eines stetig steigendem Bedars nach mehr Rechenleistung, die sich vom Uniprozessorsystem (*UP*) zum Symmetrischen Multiprozessor System (*SMP*) vollzogen hat, so erscheint die Clustertechnologie schlicht als logische Fortsetzung dieses Trends.

Eine bei dieser Entwicklung maßgebliche Eigenschaft von Clustern ist die Möglichkeit, weitgehend in die Architektur eingreifen zu können und den Clusteraufbau an persönliche Bedürfnisse anzupassen. In der Vergangenheit war dies bei *UP* und *SMP* System kaum oder nur unter großen Zusatzkosten möglich.

Genau diese Offenheit und leichte Modifizierbarkeit der Clusterarchitektur führte zu zwei großen Entwicklungstrends von Clusternutzergruppen. Während die Wissenschaft und Forschung ein verstärktes Augenmerk auf das sogenannte *High Performance Computing (HPC)* legte, so fokussierte sich der traditionelle Informatik-Dienstleistungssektor eher auf den Bereich der *High Availability*.

Interessant ist, dass obwohl beide Nutzergruppen unterschiedliche technische Anforderungen an einen Cluster stellen, die grundlegenden Technologiemerkmale und Hardwarecharakteristika dennoch die gleichen sind.

Wie definiert sich nun der offenbar breite und sehr vielschichtige Begriff eines Clusternetzwerks? Nach [Luc05] definiert sich ein Cluster wie folgt:

[A cluster is] A closely coupled collection of computer systems that shares a common infrastructure and provides a parallel set of resources to services or applications.

(Ein Cluster ist eine eng verbundene Sammlung von Computer Systemen, die eine gemeinsame Infrastruktur besitzen und einen Parallelzugriff auf Ressourcen für Dienste oder Anwendungen bereitstellen.)

Etwas länger und umfangreicher, aber dennoch als eine schöne Zusammenfassung dieses Kapitels zu sehen ist die Begriffsklärung von [Wik06]:

Ein Computercluster, meist einfach Cluster (von engl. cluster – Schwarm, Gruppe, Haufen), bezeichnet eine Anzahl von vernetzten Computern, die von außen in vielen Fällen als ein Computer gesehen werden können. In der Regel sind die einzelnen Elemente eines Clusters untereinander über ein schnelles Netzwerk verbunden. Ziel des *Clustering* besteht meistens in der Erhöhung der Rechengeschwindigkeit oder der Verfügbarkeit gegenüber einem einzelnen Computer. Die in einem Cluster befindlichen Computer (auch Knoten oder Server) werden auch oft als Serverfarm bezeichnet.

3 Cluster Hardware und Architektur

3.1 High Performance, High Throughput and High Availability

Beliebige Clusterkonfigurationen können ganz grob in drei Kategorien unterteilt, die sich jeweils durch das Nutzungsverhalten oder die Art der Dienste, die sie bereitstellen, unterscheiden.

Ein *High Performance*-Cluster bietet so zumeist eine oder mehrere Parallelanwendungen zur Nutzung an und stellt dafür Prozessor- und Arbeitsspeicherressourcen zur Ausführung zur Verfügung, die sich technisch in UP/SMP Systemen nur schwer realisieren lassen würden.

Der *CLAN* Linux Cluster [Cla06] des CMS der Humboldt Universität in Berlin stellt den Nutzern z.B. die Rechenkapazität von 32 Dual Intel Xeon Prozessoren mit einer Taktfrequenz von 2,6-3,0 GHz zur Verfügung. Jeder der 32 Clusterknoten verfügt zusätzlich über 4 GB an Arbeitsspeicher, so dass eine Parallelanwendung über maximal 128 GB Arbeitsspeicher verfügen könnte. In einer einzelnen Workstation sind diese Ressourcen nur sehr schwer unterzubringen.

High Throughput-Cluster sind im Gegensatz dazu stark für die Ausführungen einer sehr großen Anzahl von unabhängigen Jobs optimiert. Hier benötigen die einzelnen Jobs nicht so viele Ressourcen, es geht im Wesentlichen darum, möglichst vielen Nutzern zur gleichen Zeit bestimmte Dienste oder Ressourcen zur Verfügung zu stellen.

Mit dem Ziel einen *garantierten* Zugriff auf gemeinsam genutzte Ressourcen zu erlauben, legen *High Availability*-Cluster ihren Fokus auf Verfügbarkeit. *Redundanz* und *Fail-Over-Techniken* auf der Hardware- und Softwareseite kommen hier verstärkt zum Einsatz.

3.2 Beispiele für die verschiedenen Clustertypen

Als Vertreter für die **High Throughput Cluster** sind *Carpet Cluster* bzw. *Compute Farms* zu nennen. Bei ersterem handelt es sich um einen Kleinstcluster, der ganz einfach auf dem Teppich unter dem Schreibtisch eines Anwenders stehen kann. Latenzzeiten bei der Kommunikation der beteiligten Rechner sind hier nur sekundär. Auch die von den Jobs benötigten Daten sind meist interdependenzfrei.

Bei Compute Farms, handelt es sich um eine Umsetzung des klassischen *Client-Server Modells*. Desktop Client Computer arbeiten hier direkt auf einem Terminal Server. Dort findet die komplette Abarbeitung aller Jobs statt. Als Ergebnis wird einfach eine

graphische Benutzeroberfläche an die Clients zurückgegeben. Auf diese Weise wird die wirkliche Rechenlast im Rechenzentrum konzentriert. Die Netzwerke zu den Clients übertragen nur die wesentlich schmalbandigere Oberfläche (*Pixel-pushing*), über ein X11 oder VNC Protokoll.

Vertreter für **hochverfügbare Cluster** sind sogenannte *virtuelle Webserver* oder *Parallele Datenbank Server*. Bei den virtuellen Webservern sorgen Direktorknoten für die Verteilung an der ankommenden Webanfragen an die mehrfach vorhandenen Webserver. Durch die Vorschaltung der Direktorknoten kann ein defekter Webserverknoten vollautomatisch deaktiviert werden und so aus der Liste der aktiven Webserverknoten entfernt werden. Für den Nutzer dieses Webservercluster geschieht dies vollständig transparent. So lange mindestens ein Webserverknoten aktiv ist, erscheint der gesamte Cluster nach außen hin als funktionsfähig. Natürlich sinkt mit jedem ausgefallenem Knoten auch die Gesamtleistung.

Bei Parallelen Datenbank Servern läuft auf jedem Knoten eine Instanz der Datenbank. Die globale Konsistenz der Daten wird über globale Locks mit Hilfe von Hochgeschwindigkeitsverbindungen zwischen den Knoten realisiert. Fällt ein Knoten aus, so sind alle anderen Knoten trotzdem weiterhin in der Lage auf die Datenbank zuzugreifen.

Beispiele für **High-Performance Cluster** existieren viele. Eines der spannenderen sind jedoch Visualisierungscluster.

Typischerweise sorgt die Grafikkarte in der Workstation für die Darstellung und Rendern von Grafiken auf dem Bildschirm. Gute Grafikkarten realisieren dabei Auflösungen von ungefähr 3840 mal 2400 Pixel. Bei sehr detailgenauen Darstellungen, z.B. Landkarten oder Naturaufnahmen, kann dies nicht ausreichend sein, um die Detailtreue zu erhalten.

Der Sinn von Visualisierungsclustern liegt nun darin, diese obere Schranke zu überwinden und das Rendern von Darstellungen jenseits dieser Grenze auf mehrere Rendering nodes zu verteilen. Unter [Pan06] kann man sich Beispiele ansehen, wie das Bild erst auf die verschiedenen Knoten verteilt, separat gerendert und dann mittels mehrerer Displays dargestellt werden kann.

3.3 Bestandteile von Computerclustern

Vergleicht man die unterschiedlichen Clusterarchitekturen für die verschiedenen Einsatzzwecke, so lassen sich grundsätzlich die folgenden Module unterscheiden.

- Rechen Knoten (computer slices)
- Master Knoten (master nodes)

- Administrative Knoten (administrative nodes)
- Hochgeschwindigkeitsverbindung (highspeed interconnect, HSI)
- Management-, Control- und Datennetzwerke
- File servers
- Storage
- Testknoten, Spareknoten

Ziel der *master nodes* ist es, einen *Single-Point of Access (SOA)* für den gesamten Cluster zu erstellen. So können die Nutzer ihre Jobs in eine Warteschlange auf den Master Knoten einreihen. Eventuell können hier noch zusätzliche Attribute, wie Prioritäten, ergänzt werden, um die Warteschlange optimal auszunutzen. Die Jobs werden nun mittels einem Scheduler auf die verschiedenen *compute slices* übertragen. Dabei können ggf. auch mehrere slices an der Erfüllung eines einzelnen Jobs beteiligt sein.

Zwischen den einzelnen Knoten müssen im Laufe des Betriebes Nachrichten zur Synchronisation ausgetauscht werden. Dazu dient ein spezielles Netzwerk, das *Highspeed Interconnect (HSI)* Netzwerk. Falls auf gemeinsam genutzte Daten zugegriffen werden soll, so existiert meist noch ein zusätzliches Datennetzwerk als Schnittstelle zu einem *File server* oder einer anderen Art von *Storage*.

Control networks, wie z.B. serielle Konsolenzugänge, erlauben es dem Systemadministrator, einfache Wartungsvorgänge, wie einen Kaltstart oder das Prüfen eines Systemstatus, zentral über einen gemeinsamen Zugangspunkt zu erledigen.

Test- und spare nodes sind nicht am aktuellen Betrieb beteiligt sondern entweder für Testzwecke oder für Redundanz von anderen compute slices vorgesehen.

4 Grundlagen der Netzwerke in Clustern

4.1 Management- und Controlnetzwerke

Performance ist in einem Managementnetzwerk nicht so wichtig wie in Datennetzwerken oder im HSI Umfeld. Der primäre Zweck eines solchen Netzwerks ist vielmehr die leichte Fehlerbehebung und Konfiguration von Geräten, ausgehend von einem zentralen Zugangspunkt im Netzwerk. Ziel ist der reine Zugriff auf alle Managementports mit einer vernünftigen Geschwindigkeit. Ein weiterer wichtiger Grund für die Separierung der Netzwerke ist der Sicherheitsaspekt. Trennt man Nutzung und Management nicht sauber voneinander, so ist es böswilligen Clusternutzern leichter, den gesamten Cluster in seiner Funktionsfähigkeit zu beeinflussen und hemmen.

Obwohl eine Trennung der Netze natürlich immer gewünscht wird, so kann es dennoch aus Kostengründen sinnvoll sein, ein einzelnes physikalische Netz für die Nutzung von Daten- und Managementnetz zu teilen.

Wichtig ist außerdem, dass nicht nur die beteiligten Cluster nodes mit dem Managementnetzwerk verbunden sind. Auch Netzwerkswitches werden hierüber administriert und gewartet.

Schon ein kleines Beispiel zeigt, wie sinnvoll der Einsatz eines solchen Management Netzwerks bei täglichen Aufgaben sein kann. Das Ausschalten eines Knotens und der anschließende Neustart möge etwa 30s pro Maschine in Anspruch nehmen. Bei einem Cluster mit 1024 Knoten, was durchaus keine Seltenheit ist, würde diese Aufgabe schon fast 8,5h verschlingen. Natürlich vorausgesetzt, dass es nur einen Administrator gibt und dieser alle Knoten nacheinander neu startet. Dies ist ein gutes Beispiel für die Tatsache, wie kleine Aufgaben, die außerhalb des Clusterumfeldes nur wenig Zeit kosten, in einem Cluster plötzlich immense Ausmaße annehmen können.

Ein gelungenes Beispiel für die Management Möglichkeit ist das Light-Out-Management der Server von SUN. Mittels eines Konsolenswitches kann sich der Administrator ohne viel Aufwand mit der Management Console eines SUN Servers via *telnet* verbinden. Dort kann mit einfachen Befehlen der Status abgefragt, ein Neustart ausgelöst oder eine grundlegende Konfiguration vorgenommen werden. Mit besseren Terminalprogrammen kann so relativ einfach der Neustart allen Clusterknoten initiiert werden.

4.2 Datennetzwerke

Das Datennetzwerk verbindet sowohl die einzelnen Cluster nodes untereinander, aber auch mit der eingesetzten Storage Technologie. Unabhängig davon, welche Art von

Dateisystem (z.B. IBM GPFS) oder Storage (z.B. IBM Total Storage) eingesetzt wird, muß das Datennetzwerk natürlich für den Transport von großen Datenpaketen (*bulk packets*) optimiert sein. Eine *geswitchede* Gigabit Ethernet Verbindung stellt hier eine preiswerte Alternative für diese Verbindungsanforderungen dar.

Eine andere Art der Optimierung für solche Ethernetverbindungen, insbesondere für die Nutzung durch das *Network File System (NFS)*, stellen sogenannte *Jumboframes* dar [Jum06]. Bei klassischen Ethernet Netzwerken ist die *Maximum Transfer Unit (MTU)* auf 1500 Bytes pro *MAC Frame* festgelegt. Bei einem Jumbo Frame wird diese MTU auf bis zu 9 kB erhöht. Natürlich müssen die Switches und die Ethernet-Netzwerkkarten diese Technologie auch unterstützen.

Im Zusammenhang mit dem Einsatz von Jumbo Frames ist ebenfalls der Einsatz von *VLANs*. VLANs können in einem physikalischen Netz verschiedene Kollisionsdomänen erstellen, also in einem physikalischen Netz verschiedene logische Netze unterbringen. Es bietet sich nun an, Jumbo Frames nur in bestimmten VLANs zuzulassen. So wird nicht Jumbo Frames - fähige Hardware sauber vom restlichen Netzwerk abgetrennt.

Die Designentscheidungen über die Art des Datennetzwerks hängt natürlich auch sehr stark von der Clusteranwendung ab, die voraussichtlich Zugriff auf die Daten haben wird. In Abhängigkeit vom Nutzungsgrad des Clusters brauchen vielleicht nicht alle Knoten einen gleichzeitigen Zugriff auf die gemeinsamen Daten. Letztendlich ermöglicht Datenpartitionierung oder Netzwerkpartitionierung eine Lastenteilung auf der File Server- bzw. Stageseite.

4.3 HSI - Highspeed Interconnect

Hochgeschwindigkeitsverbindungen mit möglichst kleiner Latenzzeit sind die Grundlage der Kommunikation zwischen Parallelanwendungen oder im Bereich der Clusterdatenbankanwendungen. Die Wahl der Kommunikationsverbindung zwischen den Knoten, mit besonderem Augenmerk auf den avisierten Performancebedarf und den gesetzten Budgetgrenzen, bedarf einer guten Vorplanung.

Bei Parallelanwendungen kommt dem HSI die Funktion von gemeinsam genutztem Speicher zu, d.h. je näher die HSI Geschwindigkeit an der Speicherbusgeschwindigkeit der einzelnen Clusterknoten liegt, umso performanter ist die Ausführung der geplanten Clusteranwendung. Bei Datenbankanwendung wird das HSI meist zur Realisierung des globalen Locking Mechanismus oder zur Synchronisation der einzelnen Caches benutzt. Bei Parallelanwendungen ist die Latenzzeit der entscheidende Parameter, bei Datenbankanwendungen spielt die verfügbare Bandbreite eine wesentlich höhere Rolle. Welchem der beiden Parameter eine höhere Wichtigkeit zukommt, entscheidet letztendlich auch über die Art des eingesetzten Netzwerks.

Weitere wichtige Eigenschaften von HSI sind die Blockierungsfreiheit, d.h. die Kommunikation zwischen zwei Knoten darf die Kommunikation anderer Knoten nicht blockieren, und Ausfalltoleranz, d.h. für einzelne Knoten müssen unterschiedliche Wege zur Verfügung stehen. Zudem ist es natürlich wichtig, dass bei einer nachträglichen Skalierung des Netzwerks diese Eigenschaften erhalten bleiben.

Die intuitiv einfachste Möglichkeit, einfach alle System miteinander zu vernetzen ist zwar in bezug auf die Performance eine optimale Wahl, aber die technische Realisierung mit Crossbar Switches ab einer Größenordnung von 64 Knoten treibt die Anschaffungskosten in astronomische Höhen und verbietet sich deshalb für die meisten Cluster.

In der Vergangenheit wurden daher verschiedene Lösungen entwickelt, um die Spezifika der verschiedenen Anwendungsgebiete möglichst optimal auszunutzen. Im Allgemeinen unterscheidet man heute zwei Klassen von Verbindungsnetzen, die *Statischen* und die *Dynamischen Verbindungsnetze*. Bei den statischen Verbindungsnetzen handelt es sich um direkte Verbindungen zwischen den Knoten, d.h. die Weiterleitung von Netzwerkdaten erfolgt in den Knoten selber. Klassische Netzwerktopologien, wie der Ring, ein Gitter oder ein Hyperwürfel kommen hier zum Einsatz.

Bei den Dynamischen Verbindungsnetzen sind die Knoten indirekt über einen oder mehrere Switches miteinander verbunden. Das klassische Beispiel hier ist das Clos-Netz [Clo52] oder eine Crossbar Verteiler. Bei einem Clos Netz handelt es sich um ein erweiterbares Netzwerk auf der Basis von Crossbar Switches. Als zusätzliche Eigenschaft ist hier ebenfalls zu nennen, dass die besondere Form der Topologie eine maximal mögliche *Bisektionsbandbreite* garantiert. Bei der Bisektionsbandbreite handelt es sich um die verfügbare Bandbreite in ein Worst-Case Szenario, also den Fall, wenn die eine Hälfte der Knoten an die andere Hälfte der Knoten senden möchte. Aufgrund der letztgenannten Eigenschaften eignet sich diese Topologieform für das HSI und wird auch sehr oft eingesetzt.

Marktführer auf dem Gebiet der Highspeed Interconnect Verbindungen ist die Firma *Myrinet Inc.* mit dem Produkt *Myrinet* [Myr06]. Auszug aus der Website von Myrinet:

There are today many thousands of Myrinet clusters in operation in more than 50 countries. These clusters range in size to more than 2,000 hosts. You will find Myrinet clusters used for high-performance computing (HPC) applications throughout the TOP500 supercomputer list [Top06]: 101 (20.2%) of the November-2005 Top500 systems use Myrinet technology, far more than any other low-latency technology. Myrinet is also widely used in commercial and embedded-computing applications.

Myrinet gestattet eine Bandbreite von 495 MBytes/s (3,96 Gb/s) bei einer mittleren Latenzzeit 5,7 μ s. Dabei werden pro Link bei der Verkabelung zwei Leitungen einge-

setzt. Hier können sowohl Glasfaser, als auch Kupferleitungen eingesetzt werden. So sind pro Leitung zwischen 640 Mb/s und 2,56 Gb/s als Bitrate möglich. Die maximale Paketgröße ist 4 MByte. Die Flußkontrolle findet byteweise in der Bitübertragungsschicht statt. Da die komplette Topologie jedem Switch bekannt ist, legt der Sender die Route jedes Paketes selber fest, es kommt also das Prinzip des Source Routings zum Einsatz. Dabei entfernt jeder Switch das erste Element aus der im Paket angegebenen Route und leitet es dann ggf. weiter.

Basistechnologie bei Myrinet sind Crossbar Switches mit derzeit maximal 16 Ports. Die Topologie ist ein wenig von der Anzahl der Knoten abhängig:

- max. 128 Knoten bei zweistufigem Netz mit 24 *Crossbars*
- max. 1024 Knoten in einem dreistufigen Netz mit 320 *Crossbars*
- max. 8192 Knoten in einem vierstufigen Netz mit etwa 3500 *Crossbars*

Das größte System weltweit umfasst derzeit 1782 Knoten mit je zwei PowerPC-Prozessoren und steht damit auf Platz 4 der Top500 Liste [Uni06].

Neben Myrinet sind noch weitere Produkte im Umfeld des HSI zu nennen. Zum einen wird zunehmend häufiger auch das aus dem Konsortium von IBM, Microsoft, Intel, Hewlett-Packard, Compaq, Dell und Sun hervorgegangene *Infiniband* für die Vernetzungen eingesetzt. Aus verschiedenen Gründen hat Infiniband aber die anfänglichen Erwartungen nicht erfüllen können und sowohl Microsoft als auch Intel scheinen von der Entwicklung von Infiniband zurückgezogen zu haben.

Im Zuge der Weiterentwicklung des traditionellen Ethernet Standards zum 10 Gigabit Ethernet, wird auch diese Technologie zunehmend für preiswertere Cluster interessant. Diesen Trend erkennend, wird auch Myrinet Inc. in Zukunft in der *Myri-10G* Produktserie auch Geräte auf der Basis von Gigabit Ethernet für Low-Cost-Cluster anbieten [Myr06].

5 Betriebssysteme für Cluster

Einen Cluster aus verschiedener Hardware aufzubauen und dann geschickt zu vernetzen ist nur der Anfang. Um diese Hardware in eine arbeitsfähigen Cluster umzuwandeln, benötigt man natürlich noch ein clusterfähiges Betriebssystem und die entsprechenden Parallelanwendungen.

Folgende Attribute sind für ein Betriebssystem im Clusterumfeld relevant und können als Entscheidungsgrundlage dienen:

- Hardware Unterstützung
- Stabilität
- Kosten
- Verwaltbarkeit
- Skalierbarkeit
- Softwareverfügbarkeit
- Support und sonstige Produktunterstützung

Die Hardwareunterstützung für die im Cluster verwendete Hardware ist als essentiell zu betrachten. Nur durch eine vollständige Unterstützung aller vorhanden Geräte, kann der Cluster sein volles Potenzial ausspielen und die geplante Performance erreichen.

Gerade weil der Cluster nach außen hin als eine logische Maschine von den Endanwendern wahrgenommen wird, ist die Gesamtstabilität nicht zu letzt auf der untersten Ebene auch von der Stabilität des Betriebssystems abhängig. Stabilität auf Betriebssystemebene bezieht sich hier auf die Fähigkeit der Software, lange Zeit ohne Neustart oder Absturz zu überleben.

Kosten sind ein sehr wichtiger Aspekt bei der Entscheidung für das passende Betriebssystem. Gerade weil jeder Knoten mit einer Kopie dieses Systems läuft, steigen die Softwarekosten schnell an. Gerade dies sollte jedoch mit der Anschaffung eines Clusters im Vergleich zur gleichen Anzahl unabhängiger Systeme vermieden werden.

Schließlich muß das Betriebssystem auch Hilfsmittel für Fernwartung und sonstige Administration bereitstellen. Gerade kleinere Aufgaben, die an einem einzelnen Knoten nur kurze Zeit in Anspruch nehmen, können im Cluster Umfeld schnell beträchtliche Ausmaße annehmen. In diesem Zusammenhang ist es auch sehr sinnvoll, eine geschickte Strategie zum Einspielen von Updates auf allen beteiligten Clusterknoten zu erstellen.

Manche Betriebssysteme bieten hier Mechanismen an, um diesen Vorgang zu automatisieren und damit Last von den Schultern der Administratoren zu nehmen.

Mit dem Punkt Skalierbarkeit sind zwei Arten von Skalierung gemeint. Zum einen sollte das Betriebssystem den Einsatz von mehreren Prozessoren in einem Knoten gestatten (SMP-fähig), denn auch so läßt sich die gesamte Leistung eines Clusters mit relativ geringem Aufwand nachträglich steigern. Zum anderen geht es hier um die Skalierung der Kommunikationsinterfaces zu anderen Clusterknoten. Mit anderen Worten, einer nachträglichen Ergänzung von weiteren Clusterknoten und der hierbei benötigten zusätzlichen Netzwerkverbindungen darf das Betriebssystem nicht im Weg stehen.

Wenn auch nicht technisch direkt mit dem Wahl eines Betriebssystems verknüpft, so doch immens wichtig sind noch die Faktoren Softwareverfügbarkeit, sowie Support und sonstige Produktunterstützung. Was nützt ein Cluster, für den es keine Anwendungen gibt? Oder was nützt ein Cluster, wenn ein entscheidender Bug im Betriebssystem sinnvolles Arbeiten verhindert?

Neben den verschiedensten Unix Derivaten (AIX, HPUX) oder Linux Distributionen (SLES, RHEL), hat der Anwender natürlich auch die Wahl Produkte von Microsoft einzusetzen. Die letztendliche Entscheidung ist stark von den unterschiedlichen Gegebenheiten abhängig und kann im Rahmen dieser Arbeit nicht allgemeingültig getroffen werden.

6 Cluster Middleware

Im folgenden wird nun die Umgebung betrachtet, die die Ausführung von Cluster Software und Parallelanwendungen ermöglicht. In diesem Zusammenhang werden auch Management- und Monitortools näher beleuchtet. Da der Fokus dieser Arbeit jedoch verstärkt auf der Netzwerkkommunikation liegt, kann hier nicht so detailliert auf alle Einzelheiten eingegangen werden.

Die Cluster Middleware beschäftigt sich vorwiegend mit den folgenden Aufgabenbereichen:

- Allokierung von Ressourcen, die notwendig sind, um Clusteranwendungen zu starten
- Ausführen eines *Schedulers*, um konkurrierende Jobs vernünftig verwalten zu können
- Erstellen von Kommunikationspfaden über das HSI zwischen den beteiligten Clusterknoten
- Überwachen der Performance und sonstiger Systemparameter der Knoten

Eine spezielle Opensource Version einer *Message Passing Interface (MPI)* Bibliothek, um die Kommunikation einzelner Komponenten einer Parallelanwendung zu vereinfachen ist *MPICH* [Mpc06]. Auch von Myricom wird eine besonders auf Myrinet angepasste Version dieser MPI Bibliothek unter dem Namen *Myrinet GM interconnect* zum Download zur Verfügung gestellt.

MPICH implementiert den MPI Version 1.0 Standard und MPICH2, dass den MPI 2.0 Standard implementieren soll, ist zur Zeit im Beta Status. Genauere Informationen zum MPI Standard findet man hier [Mps06].

Ein Opensource Resource Management System, das häufig Anwendung findet ist zum Beispiel *SLURM - Simple Linux Utility for Resource Management* [Slu06]. Slurm ist dafür konzipiert drei Aufgaben zu erfüllen:

- Garantie von exklusiven und nicht exklusiven Zugriff auf Ressourcen (Knoten) für einen gewissen Zeitraum, um Wartungsarbeiten durchführen zu können
- Bereitstellung einer Umgebung, um (Parallel-) Anwendungen auf einer Menge von Knoten starten, stoppen oder überwachen zu können
- Serialisieren von konkurrierenden Ressourcenzugriffen durch einen Job-Scheduler

Slurm wurde erfolgreich bei der Entwicklung des BlueGene/L Clusters mit 65,536 Knoten eingesetzt und ist daher als sehr stabil und zuverlässig anzusehen.

Das letzte Opensource Produkt, das in die Kategorie der Middleware Produkte fällt und reif für den Produktiveinsatz ist, ist die *Maui Scheduler Software* [Mau06]. Maui ist zugleich ein Scheduler und Manager für MPI Anwendungen und Myrinet oder Ethernet als HSI. Die Funktionsvielfalt von Maui ist sehr komplex und sehr vielseitig. So existieren Schnittstellen zu weiteren Paketen, wie zum Beispiel Load-Balancern.

Die Auswahl an Software für diesen Bereich ist gigantisch. Häufige Anwendung findet auch *Nagios* oder *Ganglia*, aber eine zufriedenstellende Beschreibung der Möglichkeiten dieser Software ist im Rahmen dieser Arbeit leider nicht möglich. So vielfältig die Anforderungen im Middlewarebereich sind, so vielfältig ist glücklicherweise auch das Softwareangebot.

7 OpenSource Cluster

7.1 OSCAR

Nachdem die vorigen Kapitel viele verschiedene Konzepte für die unterschiedlichen Subsysteme eines Clusters eingeführt haben, ist es nun leicht die Vorteile eines quasi-automatischen Cluster-Building Toolkits, wie *OSCAR*, zu schätzen.

Obwohl es relativ viele Tools gibt, die einen Teil des Cluster-Baus automatisieren, so gibt es doch nur wenige die den kompletten Prozess automatisieren. Die beiden Hauptvertreter sind *OSCAR* [Osc06] und *ROCKS* [Roc06]. Beide erlauben es einem, einen vollständig funktionsfähigen Cluster, quasi out-of-the-box zu erstellen. Für welchen von beiden man sich dann entscheidet, hängt ein wenig von der Clustergröße und den persönlichen Präferenzen des Systemadministrators ab. Für Studienzwecke wird häufig OSCAR empfohlen.

Durch die Nutzung von Technologien, wie TFTP in Verbindung mit PXE fähigen Netzwerkkarten, ist OSCAR in der Lage, den Installationsprozess der Clients enorm zu vereinfachen und zu standardisieren. Nachdem der Cluster installiert ist, mischt sich OSCAR kaum noch in die täglichen Prozesse ein und läßt den Anwender das komplette Management übernehmen. Im Prinzip kann OSCAR als eine Richtlinie gelten, denn viele der oben angesprochenen Subsystem werden hier automatisch vorkonfiguriert, so dass eine nachträgliche Modifikation der Parameter in einfach Clusternetzen selten notwendig ist. Da OSCAR ebenfalls OpenSource ist, kann dieses Produkt auch als Ausgangsbasis für einen komplett modifizierten und auf die eigenen Bedürfnisse angepassten Cluster verwendet werden.

Obwohl OSCAR ein guter Startpunkt zum Lernen des Umgangs mit einem Clusters ist, so gibt es dennoch ein paar Nachteile, die hier nicht unerwähnt bleiben können.

- Beschränkung auf ein einziges Ethernet Netzwerk im inneren des Clusters
- Ein Großteil der Infrastrukturdienste sind in einem einzigen Master Knoten untergebracht
- Highavailability zwischen mehreren Masterknoten nur mit Zusatzpaketen realisierbar

Trotz dieser Nachteile können wir OSCAR nur empfehlen, um Cluster Architektur zu verstehen oder zu lernen.

7.2 Beowulf

Beowulf ist ein Cluster, der unter dem freien Betriebssystem Linux oder BSD läuft. Ein Beowulf Cluster ist aber vielmehr als eine Art Definition eines abstrakten Clusters mit bestimmten Eigenschaften zu sehen. Es gibt also nicht *die* eine Beowulf-Cluster Implementation. Vielmehr ist ein beliebiger Cluster, der bestimmten Eigenschaften genügt, auch ein Beowulf Cluster.

Per Definition besteht ein Beowulf-Cluster aus Hardware einer bestimmten Kategorie und natürlich aus Software, die zum Parallel-Computing notwendig ist. Die Hardware-Basis eines Beowulf-Cluster besteht aus:

1. "Standard" - Rechnern, das können Intel-basierte PCs (oder Kompatible) oder solche mit Alpha-Architektur, aber auch Apple-Rechner (Zur Zeit noch PowerPC Architektur) sein; jedenfalls sollte man diese *von der Stange* kaufen können
2. und einem Netzwerk

Die Art der Netzwerk-Kopplung ist nicht festgelegt; üblicherweise werden Netzwerk-Komponenten eingesetzt, die für die zu lösenden Probleme das beste Preis- / Leistungsverhältnis darstellen.

Es wurde bereits gesagt, dass in einem Beowulf-Cluster immer ein freies Betriebssystem (Linux, FreeBSD o.ä.) eingesetzt wird. Die verwendete Anwendungssoftware sollte aber auch nicht kommerziell sein, sondern den Open Source Software-Kriterien genügen. Damit kann das Cluster als Ganzes als freies System betrachtet werden. Man kann ein Cluster auch als einen virtuellen Rechner ansehen. So etwas wird z.B. mit der Software MOSIX der Hebräischen Universität von Jerusalem versucht.

Bei der Programmierung von Anwendungen, die auf mehrere Rechner verteilt laufen sollen, muss natürlich auch der Datenaustausch zwischen den Teilprogrammen vorgesehen und programmiert werden. Dafür können standardisierte Bibliotheken genutzt werden, die eine abstrakte Kommunikationsschnittstelle zur Verfügung stellen. Im Idealfall ist diese sogar plattformunabhängig, d.h. die Kommunikation kann zwischen Prozessen auf verschiedenen Rechnern mit unterschiedlichen Plattformen ablaufen. Die bekanntesten Vertreter solcher Bibliotheken sind das schon behandelte Message Passing Interface (MPI) oder die Parallel Virtual Machine (PVM).

Nun zurück zum Beowulf. Ein Cluster besteht aus einer gewissen Anzahl von Rechen-Knoten (compute nodes), einem oder mehreren Server-Knoten (server node) und in der Regel aus einem (oder mehreren) Zugangs-Knoten (front end), auf dem bzw. denen sich die Nutzer einloggen können. Von dort aus können sie sich die benötigte Menge von Rechen-Knoten für ihre Arbeit (Auftrag) reservieren und diese benutzen. Die

Rechen-Knoten sind nur durch das Netz mit der “Außenwelt” verbunden und benötigen dementsprechend keinerlei Ein- oder Ausgabe-Peripherie wie Tastatur, Maus oder Bildschirm. Soll der Zugangsknoten auch zur Softwareentwicklung benutzt werden dürfen, muss eine entsprechende Entwicklungsumgebung (Compiler, Debugger usw.) dort bereitgestellt werden.

Mehr Informationen zu Beowulf Clustern findet man unter [Beo06].

8 Cluster Demonstration

Im Zusammenhang mit der Präsentation dieses Papers, wird es auch eine Präsentation eines Demo Clusters geben, um dem Zuschauer einen kleinen praktischen Einblick in das Arbeiten mit Clustern zu ermöglichen.

8.1 High Availability mit Heartbeat

Wenn man über hochverfügbare Systeme spricht, so wird schnell die Forderung nach einer 100% Verfügbarkeit laut. Betrachtet man allerdings die Kosten, die mit einer solchen Garantie einhergehen, so reduziert sich der Anspruch recht schnell. Für die meisten Anwendungen ist eine garantierte Verfügbarkeit von 99% durchaus ausreichend, wie die folgende Tabelle zeigt.

Uptime	Downtime	Downtime pro Jahr	Downtime pro Woche
98%	2%	7,3 Tage	3h 22min
99%	1%	3,65 Tage	1h 41min
99,8%	0,2%	17h 30min	20min 10s
99,9%	0,1%	8h 45min	10min 5s
99,99%	0,01%	52min	1min
99,999%	0,001%	5,25min	6s
99,9999%	0,0001%	31,4s	0,6s

Damit muß natürlich auch genau festgelegt werden, was genau unter einer *Downtime* zu verstehen ist. Eine einfache und sehr praktikable Definition ist die folgende aus [Ips00]:

If a user cannot get his job done on time, the system is down.

Diese Definition mag auf den ersten Blick sehr streng erscheinen. Betrachtet man jedoch die Tatsache, dass ein System einen Dienst oder eine Anwendung für die Nutzer bereitstellen soll, so ist die Nichtverfügbarkeit dieser Anwendung, unabhängig vom Grund der Störung, ein Fehlerfall und damit in die Kategorie *Downtime* zu verbuchen.

Nach einer Studie des *IEEE*[Ips00] aus dem Jahre 1995 können die folgenden Ursachen als Gründe für Downtimes angesehen werden:

- Software Fehler (40%)
- Geplante Downtime, etwa für Wartungsarbeiten (30%)

- Administratoren und Anwender (15%)
- Hardware (10%)
- Umgebung (5%)

Obwohl es viele Strategien gibt, um mit den obengenannten Gründen umzugehen und die Häufigkeit zu minimieren, so lassen sich dennoch nicht alle Eventualitäten ausschließen und die IT-Welt mußte Lösungen erarbeiten, um mit den damit verbundenen zufälligen Ausfällen umgehen zu können. Da diese Arbeit den Fokus auf Netzwerke im Clusterumfeld legt, werden im Folgenden *Failover-Strategien* im Netzwerk vorgestellt, die der Gefahr eines Ausfalls durch Redundanzen zu begegnen versuchen. Natürlich finden sich ähnliche Mechanismen auch in anderen Clustersubsystemen, z.B. der unterbrechungsfreien Stromversorgung, aber diese Themen würden den Rahmen dieser Arbeit sprengen.

Im einfachsten Fall betrachten wir hier die folgende Situation. Eine Firma betreibt einen Webserver auf der Adresse *1.2.3.4* und möchte diesen möglichst hochverfügbar gestalten. Der einfachste Fall zur Realisierung dieses Vorhabens ist eine *Active/Passive* Konfiguration. Dies bedeutet, dass es zu einem aktiven Server noch einen zweiten passiven Server gibt, der bei einem Ausfall des ersten Servers, dessen Arbeit übernimmt.

Dazu sind zwei Probleme zu lösen. Erstens muß der passive Server irgendwie vom Ausfall des aktiven Servers erfahren. Zweitens muß der passive Server die Kontrolle der öffentlichen IP Adresse *1.2.3.4* übernehmen.

Die Lösung zum zweiten Problem liegt in der Tatsache begründet, dass es möglich ist, mehrere IP Adressen an einen Netzwerkadapter zu binden. Also wird man die Konfiguration wie folgt gestalten. Zunächst erhält der aktive Server der IP Adresse *1.2.3.101* und der passive die Adresse *1.2.3.102*. Zusätzlich bindet der aktive Server nun noch zusätzlich die öffentliche Zugangsadresse *1.2.3.4* an sein Netzwerkkinterace und ist damit für zwei IP Adressen *verantwortlich*. Erfährt nun der passive Server vom Ausfall des aktiven Servers, so bindet dieser einfach die öffentliche IP Adresse *1.2.3.4* an sein Netzwerkkinterace und übernimmt nun die Arbeit vom ehemals aktiven Server.

Die Lösung des ersten Problems liegt darin, dass zwischen den beiden Servern eine zusätzliche Netzwerkverbindung, das sogenannte *Heartbeat* Netzwerk existiert. Benannt nach dem Pulsschlag, werden auch hier kurze *Keep-Alive-Pakete* ausgetauscht, die beide Server über den Zustand Ihres Partners informieren sollen. Bleibt eine solche Nachricht für eine gewisse Zeit aus, so kann der passive Server davon ausgehen, dass der aktive Server ausgefallen ist. Natürlich wäre es auch denkbar, dass der Switch ausgefallen ist und deswegen keine Pakete im Heartbeat Netzwerk ausgetauscht werden können. Um diese Möglichkeit zu eliminieren, versucht man meist, das Heartbeat Netz technisch möglichst einfach zu gestalten. Eine einfache Verbindung mittels seriellen

Kabel erfordert keinen zusätzlichen Netzwerkschwitch und gestattet so eine wesentliche präzisere Analyse des Serverausfalls.

Die softwaretechnische Umsetzung findet sich im Paket *Heartbeat* [Hea06a] und die Konfiguration hier [Hea06b].

Eine Beispiel Konfiguration für Heartbeat in der Version könnte dann so aussehen (*/etc/ha.d/ha.cf*). Dabei ist das Netzwerk *eth0* das Datennetzwerk nach außen und *eth1* das Heartbeat Netzwerk. Die Keepalive Pakete werden hier als Broadcasts verschickt. Sollte der *Ping-Test* zum Router *1.2.3.254* fehlschlagen oder der Server ganz ausfallen, so werden keine Keepalive Pakete ausgetauscht und der Sekundärserver übernimmt die Ip Adresse *1.2.3.4*, die zunächst an *Paul* gebunden war.

```
logfacility daemon          # Log to syslog as facility "daemon"
node paul silas           # List our cluster members
keepalive 1               # Send one heartbeat each second
deadtime 10               # Declare nodes dead after 10 seconds
bcast eth1                # Broadcast heartbeats on eth1 interfaces
ping 1.2.3.254            # Ping our router to monitor ethernet connectivity
auto_failback no         # Don't fail back to paul automatically
respawn hacluster /usr/lib/heartbeat/ipfail # Failover on network failures
```

In der Datei */etc/ha.d/haresources* ist dann noch der präferierte Host mit seiner öffentlichen IP Adresse einzutragen:

```
paul    1.2.3.4
```

Wichtig ist, dass diese Adresse für nichts weiter benutzt werden darf. Sie darf auch nicht beim Booten des Betriebssystems schon an irgendein Netzwerkkarte gebunden werden. Heartbeat kümmert sich vollständig um die *richtige* Anbindung dieser Netzwerkadresse.

Bei der obenerwähnten Konfiguration wird insbesondere das Programm *ipfail* eingesetzt. Dies ist als eine Art Add-On zu betrachten, dass mit der Heartbeat API kommuniziert, um den Failover-Vorgang zu initiieren. Mehr zur Konfiguration gibt es unter [Ip06].

Diese *IP-Adress-Übernahme* benötigt etwas Zeit, ehe alle ARP Caches der beteiligten Switches die Übernahme realisiert haben.

Neben dieser Methode des Failovers existieren noch zwei weitere, die hier noch genannt werden sollen. Prinzipiell ist auch eine Übernahme der MAC Adresse möglich, wenn

auch etwas unsauber, da jede Netzwerkadresse eigentlich genau eine weltweit gültige Adresse besitzen sollte und diese Forderung damit ausgehebelt wird. Dafür ist sie wesentlich schneller und reagiert quasi instantan.

Die zweite Möglichkeit des Failovers ist ein dynamische DNS Rekonfiguration, bei der ein DNS Eintrag zur Laufzeit bei Übernahme durch den Passiv Server geändert wird. Erneute DNS Anfragen verweisen dann auf den neuen Server. Mittels dieser Technik lassen sich auch Load-Balancing Strategien mit relativ wenig Aufwand implementieren.

Ein weiteres Problem tritt dann auf, wenn Clients nicht mit einem plötzlichen Server-switch klarkommen. Erfahrungsgemäß neigen Microsoft Clients dazu diese Verbindungen komplett zu schließen oder gar komplett abzustürzen. Ein solches Verhalten ist natürlich nicht hinnehmbar, so dass auch Clientseite beim Clustereinsatz auf die gewünschte Funktionalität geprüft und getestet werden sollte.

Eine ganz andere Klasse von Failover Techniken sind mit zusätzlichem Hardwareaufwand und *Network Address Translation (NAT)* möglich. Dabei werden die IP Adressen in Paketen durch ein zusätzliches NAT Gateway überschrieben. Damit hat man die Möglichkeit den Server quasi zur Laufzeit *auszutauschen*, denn für alle Clients wirkt es so, als würden sie nur mit dem NAT Gateway kommunizieren. Die eigentlichen Server im Hintergrund bemerken sie nicht. Potentiell könnte aber auch ein NAT Gateway ausfallen und sollte daher redundant ausgelegt werden. Damit ist man dann auch wieder bei der ersten Klasse der erwähnte Failover Techniken.

8.2 CLAN Cluster am CMS

An der Humboldt Universität steht jedem Nutzer ein Beowulf Cluster zur Verfügung, der aus 32-bit und 64-bit Servern besteht, die mit einer unterschiedlichen Anzahl von Prozessoren ausgestattet sind. Die Mehrheit der Server ist über ein schnelles Kommunikationsnetzwerk (Myrinet 2000) verbunden. Damit ist das LINUX-Cluster für parallele Rechnungen sehr gut geeignet. Das Rechnersystem kann einerseits zur Entwicklung von parallelen Programmen mit Hilfe von MPI genutzt werden. Es ist andererseits auch bereits mit Anwendungssoftware ausgestattet.

Berechtigte Nutzer loggen sich auf dem Login-Knoten `clan.cms.hu-berlin.de` mittels SSH2 ein. Dieser ist mit einer Firewall abgeschirmt, und nur aus der HU Domäne erreichbar. Ein Einloggen auf die Beowulf-Knoten von außen ist nicht möglich. Sollte der Login-Knoten nicht erreichbar sein, so steht ein 2. Login-Knoten (`clan3.cms.hu-berlin.de`) zur Verfügung.

Auf jedem Knoten des Clusters stehen den Nutzern `/scratch` Filesysteme zur Ablage von temporären Daten zur Verfügung. Lokal auf jedem Knoten befindet sich das Filesystem `/scratch`. Jeder Nutzer erhält ein Unterverzeichnis, `/scratch/$USER` in das

er seine Daten während der Rechnung ablegen kann. Außerdem steht an den Knoten das Verzeichnis */scratch2* als NFS-Filesystem zur Verfügung, in das ebenso Daten geschrieben werden können, die dann an allen Knoten lesbar sind. Auch hier ist ein Unterverzeichnis für jeden Nutzer angelegt. Die */scratch* Verzeichnisse unterliegen einer Säuberungsroutine, d.h. bei Erreichen der Hochwassermarke von 75% werden die Nutzer per E-Mail darüber informiert, welche Dateien älter als 30 Tage sind und im Falle des Erreichens der Hochwassermarke von 90% gelöscht werden. So bleibt Zeit, die Daten an anderer Stelle sicher abzulegen. Erst bei 90% werden dann wirklich Daten gelöscht. Auch darüber erhält der Nutzer eine ausführliche E-Mail.

Daten, die permanent abgelegt werden sollen und die im Homeverzeichnis keinen Platz finden, können im erweiterten Speicher TSM (Tivoli Storage Manager) abgelegt werden.

Um gleichzeitig auf allen Knoten Kommandos ausführen zu können, z.B. Dateien/Verzeichnisse in den lokalen */scratch/\$USER* anlegen, löschen oder kopieren zu können, gibt es parallele Kommandos, die auf allen Knoten ausgeführt werden.

```
pls [path]           list contents of a directory
prm <f1..fn>        remove specified files
pcopy <f1..fn> <dir> copy files to <dir>
pexec <command>     execute <command>
pmkdir <directory> create directory
pkilluser           root: kill processes of all users
                   user: kill processes owned by user
pps                show process information
pshowinfo          show information
pping              show response times in microseconds
```

Da das Beowulf-Cluster aus Xeon- und Opteron Knoten besteht, kann mit der Gruppenoption *-g=xeon,opteron* das Kommando auf den Xeon und den Opteron Knoten ausgeführt werden. Ohne die Gruppenoption wird es nur auf den Xeon Knoten ausgeführt. Mit der Knotenoption *-n=node1,node2,...* wird das Kommando nur auf den ausgewählten Knoten ausgeführt.

Prinzipiell können auch serielle Programme gerechnet werden, aber in der Regel soll der Rechner parallelen Programmen vorbehalten sein. Zum Parallelisieren von C- oder Fortran-Programmen steht das MPI (Message Passing Interface) zur Verfügung. Die Wahl des Compilers wird durch Umgebungsvariablen gesteuert, die durch Abarbeiten von Skripten gesetzt werden.

Auf dem LINUX-Cluster können folgende Compiler genutzt werden:

- Intel Compiler
- Portland Group Compiler
- GNU Compiler

Die Kommandos *mpicc*, *mpiCC*, *mpif77*, *mpif90* erzeugen parallele Programme, die unter MPI laufen. Sie werden mit dem Kommando *mpirun* gestartet. Die Auswahl des gewünschten Compilerumgebung (*mpich-gm-intel*, *mpich-gm-gcc*,...) erfolgt über Module.

```
module avai          listet alle verfügbaren module
module load < module > lädt das gewünschte module
module list          listet die geladenen module
module rm < module > entfernt ein module
module switch < module1 > < module2 > wechselt auf ein anderes module
```

Entscheidend für die Auswahl des gewünschten Modules ist auf der einen Seite die Wahl des Compilers und auf der anderen Seite die Wahl des Kommunikationsnetzes. Programme, die sehr viel kommunizieren, sollten unbedingt über die schnellere Myrinet-Verbindung arbeiten. Das heißt, es muss das Module verwendet werden, welches die GM-Bibliotheken einschließt. Soll beispielsweise die Berechnung der Zahl Pi parallelisiert werden, so könnte das folgendermaßen aussehen:

```
cd /scratch2/\$USER
cp /perm/run_test/cpi.c .
module load mpich-gm-intel
module list
mpicc -o cpi cpi.c
```

Das erzeugte Programm *cpi* ist eine paralleles Programm, das mit dem Intel Compiler übersetzt wurde und über das Myrinet kommuniziert. Um das Programm zu rechnen, muss es in ein Batchskript verpackt werden und an LSF geschickt werden.

Der Cluster bildet den von *Load Sharing Facility (LSF)* verwalteten Cluster *Cluster of Unix Machines (CLOU)*. Alle Rechnungen, die auf einem/mehreren der Knoten laufen sollen, sind an eine der Queues

- *comp*, *comp64* (Compiler Queues)
- *par*, *par16*, *par64* (parallele Queues)

- ser, ser64 (serielle Queues)
- g03_64 (Gaussian Queue)

abzuschicken. Das sollte vom Login-Konten aus geschehen. Die Compiler Queues sind für kurze Compiler Aufrufe und für Testzwecke gedacht. Sie haben deshalb die höchste Priorität aber nur eine geringe Verweilzeit für den Job. Die seriellen Queues sind für serielle Rechnungen (auf nur einem Knoten) und haben eine Verweilzeit von maximal 24 Stunden. Die parallelen Queues sind ausschließlich für parallele Rechnungen. Alle Queues sind Fair Share Queues, d.h. alle Nutzer bzw. Gruppen erhalten gleiche Ressourcen-Anteile in den Queues.

Bleibt leider nur ein einziger Wermutstropfen. Für Studenten der Informatik ist dieser Cluster leider nicht zugänglich, so dass sich diese Arbeit auf die Beschreibung der technischen Möglichkeiten Beschränken muß.

9 Ausblick - vom Cluster zum Grid

In den Medien wird *Grid* so langsam zum *Buzz*-Wort des Jahres und die Schlagworte *grid-enabled* und *grid-capable* scheinen das zukünftige Verkaufargument von Software zu werden. Um es deutlich zu sagen, ein Grid ist zwar ein nützliches Konzept, steckt aber von der theoretischen Fundierung her noch in den Kinderschuhen. Die Frameworks zur Erstellung von Grid-Applikationen werden erst jetzt ganz langsam standardisiert, so dass der Kauf von Software, die schon jetzt *grid-enabled* ist, als verfrüht bezeichnet werden kann.

Die grundsätzliche Frage ist jedoch immer noch, wie steht ein Grid in Relation zum Cluster?

So wie die Clustertechnologie die logische Fortführung einer technischen Entwicklung war, um den Bedarf nach mehr Rechenleistung zu befriedigen, so ist die Gridtechnologie der nächste Schritt. Ein Grid ist als eine Vernetzung von verschiedenen Clustern oder auch SMP/UP Systemen untereinander zu sehen. Also ein Meta-Cluster, dessen Knoten Cluster sind.

Durch diese globalen Vernetzung über das Internet wird die Technik mit ganz neuen Herausforderungen konfrontiert, die in naher Zukunft einer effizienten und praktikablen Lösung bedürfen.

Literatur

- [Beo06] <http://www.beowulf.org>, Juli 2006.
- [Cla06] http://www.cms.hu-berlin.de/dl/systemservice/computeservice/server/clan_all.html, Juli 2006.
- [Clo52] C. Clos. *A Study of Non-Blocking Switching Networks*. 1952.
- [Hea06a] <http://www.linux-ha.org>, Juli 2006.
- [Hea06b] <http://www.linux-ha.org/gettingstarted/oneipaddress>, Juli 2006.
- [Ipf06] <http://www.linux-ha.org/configureipfail>, Juli 2006.
- [Ips00] Robert Ipson. *Blueprints for High Availability - Designing Resilient Distributed Systems*. Wiley Computer Publishing, 2000.
- [Jum06] http://en.wikipedia.org/wiki/jumbogram#ethernet_jumbograms, Juli 2006.
- [Luc05] Robert W. Lucke. *Building Clustered Linux Systems*. Prentice Hall Professional Technical Reference, 2005.
- [Mau06] <http://mauischeduler.sourceforge.net>, Juli 2006.
- [Mpc06] <http://www-unix-mcs.anl.gov/mpi/mpich>, Juli 2006.
- [Mps06] <http://www.mpi-forum.org>, Juli 2006.
- [Myr06] <http://www.myrinet.com>, Juli 2006.
- [Osc06] <http://oscar.openclustergroup.org>, Juli 2006.
- [Pan06] <http://www.panoramtech.com>, Juli 2006.
- [Roc06] <http://www.rocksclusters.org>, Juli 2006.
- [Slu06] <http://www.llnl.gov/linux/slurm/>, Juli 2006.
- [Top06] <http://www.top500.org>, Juli 2006.
- [Uni06] www.bs.informatik.uni-siegen.de/www/lehre/ws0506/rn2/v05.pdf, Juli 2006.
- [Wik06] <http://de.wikipedia.org/wiki/computercluster>, Juli 2006.