TECHNISCHE UNIVERSITÄT ILMENAU

Fakultät für Informatik und Automatisierung
Institut für Praktische Informatik und Medieninformatik
Fachgebiet Telematik
Prof. Dr. Dietrich Reschke

Hauptseminar Telematik
Sommersemester 2003

Thema:

Einordnung und Vergleich unterschiedlicher Grid- und Clustersysteme

Betreuer: Dipl.-Inf. Thorsten Strufe

vorgelegt von:
Christine Otto
Christine.Otto@stud.tu-ilmenau.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

DRM	Distributed Resource Management
GIIS	Grid Index Information Service
GRAM	Globus Resource Allocation Manager
GRIS	Grid Resource Information Service
GSI	Grid Security Infrastructure
GSS-API	Generic Security Service API
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MDS	Monitoring and Discovery Service
MNP	Mosix Network Protocol
MPI	Message Passing Interface
MPP	Massively Parallel Processors
NCSA	National Center for Supercomputing Applications
NOW	Network of Workstations
PVM	Parallel Virtual Machine
RMI	Remote Message Invocation
RPC	Remote Procedure Call
RSL	Resource Specification Language
SAN	Storage Area Network
SMP	Symmetric Multiprocessor
SPPS	Serial Program Parallel System
SSI	Single System Image
SSL	Secure Socket Layer
WAN	Wide Area Network

Kapitel 1

Motivation

Die Zielstellung dieses Hauptseminars ist es, einen Überblick zu Grid- und Clustersystemen aufzuzeigen und damit ihre Fähigkeiten gegenüberzustellen. Aufgrund des großen Umfangs an existierenden Systemen in den Bereichen Grid und Cluster kann hier nur eine Auswahl bekannter Systeme behandelt werden. Im ersten Kapitel wird das Grid allgemein vorgestellt und anhand einer Definition charakterisiert. Unterschiedliche Ansätze für die Realisierung eines Gridsystems werden mit den Grid-Systemen wie Globus-Projekt, Legion, Condor und der SunGridEngine vorgestellt. Im folgenden Kapitel wird das Cluster mit einer Definition eingeführt und eine Klassifikation in High-Performance-Cluster und High-Availability-Cluster vorgenommen. Abschließend erfolgt eine Bewertung mittels eines Vergleichs zwischen Grid und Cluster.

Kapitel 2

Grids

2.1 Definition und Einordnung

Ein Gridsystem besitzt viele Parallelitäten zu einem Energieversorgungsnetz. Es wird eine allgegenwärtige Rechnerinfrastruktur mit vielen Rechenressourcen, Speichereinheiten, Datenbanken etc. bereitgestellt, die für gemeinschaftliches Rechnen, Rechnen mit hohem Durchsatz und andere Anwendungen genutzt werden kann. Diese Form wird oft für wissenschaftliche Zwecke aufgebaut, z.B. das NASA Information Power Grid.
Nach Kesselman wird ein Grid folgendermaßen charakterisiert: "A grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities."[1] Der Benutzer kann zu jeder Zeit und ortsunabhängig auf die Funktionalität des Grids zugreifen, ähnlich dem Nutzen der elektrischen Energie des Energieversorgungsnetzes. Zum einen soll das Grid dem Benutzer die notwendige Zuverlässigkeit und auch zum anderen eine Garantie für Funktionalität und Quality of Service (QoS) anbieten. Das Grid sollte als weitere wichtige Eigenschaft Konsistenz aufweisen, das bedeutet dem Benutzer ist es möglich die gleichen Dienste zu nutzen, unabhängig von der Zeit, dem Ort und der Hardware, über die sich der Benutzer mit dem System verbindet.
Die Funktionselemente des Grids reichen von einzelnen PCs und Workstations bis hin zu den typischeren Bausteinen des Grids: Clusternund Supercomputer. Sobald sich Supercomputer mehrfach zu einer einheitlichen Rechenressource verbinden, wird dies als Metacomputing bezeichnet. Als besondere Ressourcen können im Grid auch SIMD-Maschinen (bekannt als eine Klasse von Computern: single instruction, multiple data) enthalten, die dann ökonomischer als in einer einzelnen Organisation genutzt werden können. Ein wichtiger Forschungsbereich ist neben der Ausnutzung von speziellen Ressourcen, Security und Kalkulation das Design der Grid-Software, also das Design der Middleware. Die Middleware bietet nahtlose Integration der Ressourcen an und kann als "Wide-Area-Operating-System" angesehen werden. Diese Software verknüpft unabhängig entwi-

[1] I. Foster and C. Kesselman. Computational Grids.

ckelte Softwarekomponenten, einschließlich Client- und Serverprogramme und positioniert sich in der Mitte zwischen Hardware/Betriebssystem und Anwendungsprogramm. Fortgeschrittene Middleware überträgt nicht nur Nachrichten sondern beinhaltet zusätzlich Verzeichnis- und Authentisierungsdienste.

2.2 Ausgewählte Gridsysteme

2.2.1 Globus-Projekt

Das Globus-Projekt ist ein Forschungs- und Entwicklungsprojekt (initiiert im Jahr 1996) der University of Chicago, University of Southern California, den Hauptkooperierenden IBM und Microsoft und den Hauptmitwirkenden National Computational Science Alliance, NASA Information Power Grid Project etc. Das Ziel von Globus ist, die Anwendung von Gridkonzepten für das Rechnen in der Wissenschaft durch öffentlich zugängliche Standards zu ermöglichen. Im Bereich Forschung des Globus-Projektes werden die technischen Herausforderungen betrachtet. Typische Forschungsgebiete befassen sich mit dem Ressourcenmanagement, Management und Zugriff auf die Daten, den Transfer großer Datenmengen (wichtig für wissenschaftliche Projekte, um Simulationen zu realisieren), Anwendungsentwicklungsumgebungen, Informationsdiensten und Security. Im Bereich Software-Entwicklung des Globus-Projektes wird das Globus Toolkit angeboten. Es bietet als Middleware-System Dienste und Softwarebibliotheken an, die den Aufbau von Computational Grids und gridbasierten Anwendungen erleichtern sollen. Das Globus-Toolkit unterscheidet sich als Koordinationsmodell von anderen Grid-Middleware-Systemen, da es an die interne Struktur und an das Programmiermodell keine bestimmten Prämissen macht. So wie mit dem Protokoll HTTP eine gewisse Interoperabilität im heutigen Internet möglich ist, kann durch die standardisierten Gridprotokolle eine Form von Interoperabilität zwischen verschiedenen Implementierungen von Grids erreicht werden. Dieses Toolkit beinhaltet Software für Security, Informations-Infrastrukturen, Ressourcenmanagement, Datenmanagement, Kommunikation, Fehlererkennung und Portabilität. Es setzt auf heterogenen Low-Level-Protokollen und Systemen auf und kann von Higher-Level-Systemen wie Message Passing Interface (MPI) und CORBA genutzt werden.
Der Aufbau des Globus Toolkit lässt sich symbolisch mit drei Säulen darstellen (siehe Abbildung 2.1). Jede Säule repräsentiert eine primäre Komponente des Globus Toolkit und benutzt die Grundlagen der Security. Globus Resource Allocation Manager (GRAM) implementiert ein Resource Management Protocol, Monitoring and Discovery Service (MDS) implementiert ein Information Services Protocol und GridFTP implementiert ein Data Transfer Protocol. Alle drei benutzen das Grid Security Infrastructure (GSI) Security Protocol im Connection-Layer. Die Komponenten des Toolkits können unabhängig voneinander oder auch zusammen genutzt werden. Diese Technologien sind modular aufgebaut, aber ergänzen sich gegenseitig (komplementär). Die einzelnen Komponenten können separat oder auch zusammen heruntergeladen werden. Zusätzlich kann die Client-Software unabhängig von der Server-Software besorgt werden. Die Software wird unter der Benutzung des Grid Packaging Toolkits verpackt, entwickelt in Zusammenarbeit mit dem National Center for Supercomputing Applications (NCSA).

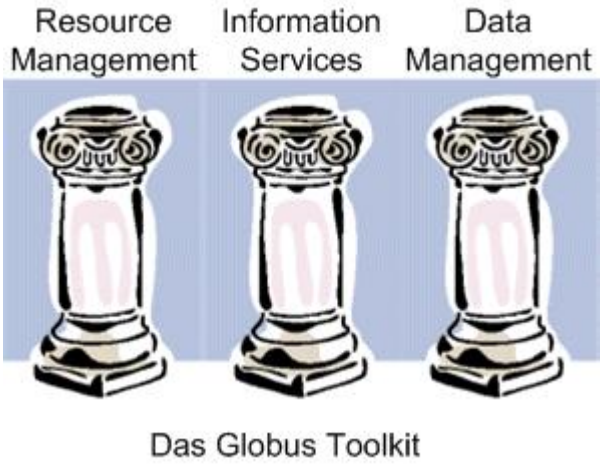

Abbildung 2.1: Modularer Aufbau des Globus Toolkits

GRAM - Resource Management

Die Globus Resource Management Architecture ist ein aus Schichten bestehendes System, in welchem "High-Level Global Resource Management Services" oben auf den lokalen "Resource Allocation Services" gelegt wird. In Abbildung 2.2 wird ein Überblick über die beliebigen Komponenten in der Globus Resource Management Architecture dargestellt.

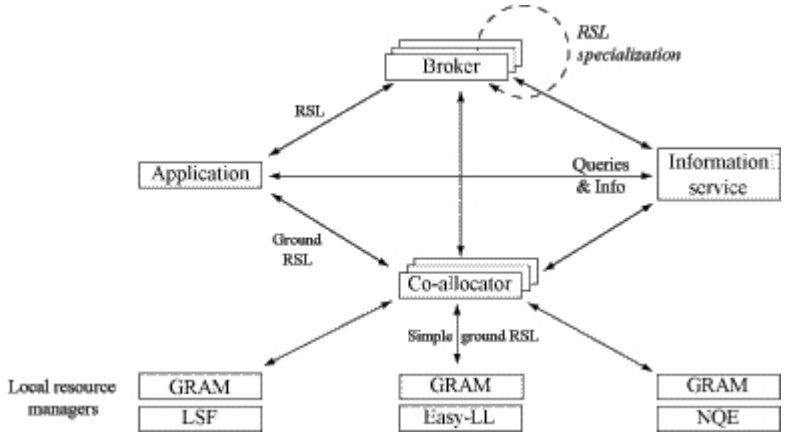

Abbildung 2.2: Globus Resource Management System

Es werden die drei Hauptkomponenten des Globus Resource Management System gezeigt:

- **Erweiterbare Resource Specification Language (RSL)**
  Die RSL bietet eine Methode für den Austausch von Informationen über Ressourcenanforderungen zwischen allen Komponenten in der Globus Resource Management Architecture.

- **Schnittstelle zu den lokalen Resource Management Tools**
  Das GRAM liefert eine standardisierte Schnittstelle zu allen beliebigen lokalen Resource Management Tools, welche gegenwärtig einen Standort besitzt, inklusive LSF, NQE, LoadLeveler und Condor. Es stellt als Funktionalitäten Ressourcenzuordnung, Prozesserzeugung, Monitoring und Managementdienste zur Verfügung. Die Implementierung des GRAM bildet Anfragen, beschrieben in RSL, mittels Befehlen auf lokale Scheduler und Rechner ab.

- **Co-Allocator**
  DUROC stellt einen Co-Allocator-Dienst zur Verfügung. Er koordiniert eine einzelne Anforderung, die mehrere GRAMs umfassen kann.

MDS - Information Services

Der MDS, ein erweiterbarer Grid-Informationsdienst, bietet notwendige Tools an, um eine LDAP (Lightweight Directory Access Protocol)-basierte Informationsinfrastruktur für Computational Grids aufzubauen. MDS benutzt das LDAP-Protokoll als uniformes Mittel für das Abfragen der Systeminformationen von einer reichhaltigen Vielfalt der Systemkomponenten, aber auch für die mögliche Konstruktion eines einheitlichen Namensraumes für Ressourceinformationen über das System, welches viele Organisationen umfasst. Der Grid Resource Information Service (GRIS) bietet ein uniformes Mittel für die Anfrage der Ressourcen im Computational Grid für ihre derzeitigen Konfigurationen, Fähigkeiten und Status. Solche Ressourcen sind z.B.:

- Rechenknoten
- Datenspeichersysteme
- Wissenschaftliche Apparate
- Netzwerkverbindungen
- Datenbanken

Der GRIS kann einfach erweitert werden, um zusätzliche Informationen anzubieten.
Der Grid Index Information Service (GIIS) stellt ein Mittel zum Zusammenknüpfen von beliebigen GRIS-Diensten zur Verfügung, um ein kohärentes Systemimage anzubieten, welches durch Grid-Anwendungen erkundet und gesucht werden kann. Somit liefern GIISes einen Mechanismus für das Identifizieren von “interessanten” Ressourcen, wobei “interessant” beliebig definierbar ist. Zum Beispiel kann GIIS alle verfügbaren Rechenressourcen innerhalb eines Verbundes von Laboratorien oder alle verteilten Datenspeichersysteme, die zu einer Agentur gehören, aufführen. Ein GIIS kann Informationen über alle Grid-Ressourcen (Rechenressourcen, Daten, Netzwerke, Instrumente) in einem bestimmten Forschungskonsortium konzentrieren, und somit ein kohärentes Systemimage des Computational Grids des Konsortiums anbieten.

GridFTP - Data Management

GridFTP ist ein High-Performance, sicheres und zuverlässiges Datentransferprotokoll, optimiert für die hohe Bandbreite von Wide-Area-Networks. Das GridFTP-Protokoll basiert auf FTP, das bekannte Internet-Dateientransferprotokoll. Es wurden eine Menge an Protokollfeatures ausgewählt, Erweiterungen im Internet Engineering Task Force (IETF) RFCs definiert und einige zusätzliche Fähigkeiten hinzugefügt, um die Anforderungen des DataGrid-Projektes zu erfüllen. Das DataGrid-Projekt, gegründet von der Europäischen Union, hat sich das Ziel gesetzt, den Zugriff auf geografisch verteilte Rechenressourcen und Speichermöglichkeiten, zu verschiedenen Organisationen gehörend, zu ermöglichen. Die bereitgestellten Ressourcen werden für riesige Datenmengen gebraucht, die durch wissenschaftliche Experimente in den Bereichen High Energy Physics, Biologie und Earth Observation entstehen.

Protokollfunktionalitäten:

- GSI-Security auf Daten- und Steuerkanal
- mehrfache Datenkanäle für parallelen Transfer
  Zwei TCP-Streams, die für dieselbe Datei verwendet werden, sind in der Regel schneller als ein einzelner, auch wenn beide vom gleichen Server laden.
- Striped-Datentransfer

  Potenzielle Bandbreitenprobleme auf der Serverseite können umgangen werden. Denn GridFTP ermöglicht es, verschiedene Teile der gleichen Datei von mehreren Servern gleichzeitig zu laden.
- Partieller Dateitransfer
  GridFTP verwendet neue Befehle, mit denen ein kontrolliertes Terminieren eines teilweisen Transfers möglich ist. Zum Beispiel beim Herunterladen von Dateien im AVI-Format werden nur Anfang und Ende der Datei benötigt, wenn nur der Beginn bzw. das Ende des Films angesehen werden soll. Unter Verwendung von Standard-FTP ist nur das Herunterladen der gesamten Datei oder bis zu einem bestimmten Offset realisierbar.
- Third-Party (direkte Server-to-Server)-Transfer
  Wenn Bandbreitenverlust oder Umwege bei großen Daten-Transfers in "Virtuellen Organisationen"[2] vermieden werden sollen, kann die Kontrolle über den Transfer an einen Dritten übergeben werden.
- Authentisierte Datenkanäle

[2]Eine Menge an Institutionen, die gemeinsam Regeln vereinbart haben, wird als eine Virtuelle Organisation bezeichnet, z.B. "Application Service Provider" oder "Storage Service Provider".

- Wiederbenutzbare Datenkanäle
- Command-Pipelining

GSI - Security

Das Globus Toolkit benutzt die GSI, um sichere Authentisierung und Kommunikation über ein offenes Netzwerk zu ermöglichen. GSI bietet eine Anzahl von nützlichen Diensten für Grids, inklusive der wechselseitigen Authentisierung und dem "single-sign-on".
Die primären Motivationen hinter GSI sind:

- Das Bedürfnis einer sicheren Kommunikation (authentisiert und vielleicht vertraulich) zwischen den Elementen des Computational Grids.
- Das Bedürfnis Security über Organisationsverbände zu unterstützen, daher wird ein zentralgemanagtes Securitysystem unterbunden.
- Das Bedürfnis "single-sign-on" für Benutzer des Grids zu unterstützen, inklusive der Delegation von Credentials für Berechnungen, die mehrere Ressourcen und/oder Standorte bedingen.

GSI basiert auf der Public-Key-Verschlüsselung, X.509 Zertifikaten und dem Secure Socket Layer (SSL)- Kommunikationsprotokoll. Erweiterungen für diese Standards wurden für "single-sign-on" und Delegation hinzugefügt. Die Implementierung des Globus Toolkits von GSI ist verbunden mit dem Generic Security Service API (GSS-API), welche eine Standard-API für Securitysysteme ist, gefördert von der IETF.

2.2.2 Legion

Legion, im Jahre 1993 an der University of Virginia entworfen, ist ein objektbasiertes System für Metacomputing-Environments. Dabei werden verteilte Ressourcen nahtlos integriert und über ein Hochgeschwindigkeitsnetz verknüpft. Es erscheint dem Benutzer als ein Rechner, zu dem er Zugriff auf verschiedene Arten von Daten und physikalische Ressourcen wie digitale Bibliotheken, physikalische Simulationen, Kameras, Linearbeschleuniger und Videostreams haben. Um eine Zusammenarbeit in Forschung und Informationsaustausch zu ermöglichen, können Benutzergruppen virtuelle Arbeitsbereiche aufbauen, unterstützt durch Legions transparenten Scheduling, Datenmanagement, Fehlertoleranz, Site Autonomy (Standortunabhängigkeit) und verschiedene Securityoptionen. Jede Systemkomponente in Legion (sowohl Software als auch Hardware) wird als Objekt modelliert. Die Schnittstellen der Objekte werden mit Interface Definition Language (IDL) beschrieben. Die Kommunikation zwischen den Objekten erfolgt über eine asynchrone Variante des Remote Message Invocation (RMI, auch Kommunikationsmechanismus zwischen Client und Server, es ist eine Adaption des Remote Procedure Call [RPC, Kommunikation zwischen Server und Client, der aufgerufene Prozess nimmt die Serverrolle in Anspruch und der Aufrufende des Prozesses spielt den Client] bezogen auf objektorientiertes Rechnen). Dabei wird jeder Prozess einem Objekt zugeordnet, welches dann Objektmethoden ausführt. Durch einen Klassenmanager, der gruppenspezifisch und programmierbar ist, können Gruppen von Legion-Objekten verwaltet werden. Dieser Manager übernimmt die Objekterzeugung, das Scheduling und die Aktivierung/Deaktivierung der Objekte. Legion legt seinen Schwerpunkt auf Ressourcenmanagement und High-Performance-Rechnen, damit unterscheidet es sich doch von CORBA. Es besteht die Möglichkeit, Parallelprogrammiersprachen und Bibliotheken (HPC++, MPI) in Legion zu implementieren. Deshalb besitzt Legion eine gewisse Bedeutung im Bereich des parallelen Programmierens, u.a. auch weil die parallelen Programme in Legion-Objekten gekapselt werden können.

2.2.3 Condor

Condor ist ein Produkt des Condor Research Project an der University of Wisconsin-Madison (UW-Madison). Das Projekt startete im Jahr 1988, aufbauend auf Ergebnissen des Remote Unix (RU)-Projektes, und arbeitet auf dem Gebiet des Distributed Resource Management (DRM). Es wurde zuerst als Produktionssystem im UW-Madison Department of Computer Science vor fast 15 Jahren installiert. Diese Condor-Installation diente damals als eine Hauptquelle von Rechenleistung für die UW-Madison Fakultät und für die Studenten. Heute managt Condor in der Condorentwicklungsabteilung mehr als 1000 Workstations. Entsprechend der Benutzungsstatistiken liefert Condor mehr als 650 CPU Tage an UW-Forscher. Zusätzliche Condor-Installationen wurden über die Jahre hinweg auf dem Campus der University of Wisconsin-Madison (UW-Madison) und in der Welt eingesetzt. Hunderte von Organisationen in Industrie, Regierung und Hochschulen haben Condor benutzt, um Recheninstallationen angefangen von einer Hand voll bis hin zu über tausend Workstations einzurichten.
Condor ist ein spezialisiertes Lastenmanagementsystem für rechenintensive Jobs. Wie andere voll gestaltete Stapelverarbeitungssysteme, bietet Condor Job-Warteschlangenmechanismus, Schedulingstrategie, Prioritätenschema, Ressourcenüberwachung und Ressourcenmanagement. Benutzer reichen ihre seriellen oder parallelen Jobs ein, Condor ordnet sie in die Warteschlange, wählt basierend auf einer Strategie wann und wo diese Jobs laufen werden, überwacht sorgsam ihre Entwicklung und informiert letztendlich den Benutzer über deren Beendigung.
Während die angebotene Funktionalität der von traditionelleren Stapelverarbeitungssystemen ähnlich ist, erlaubt Condors neue Architektur, dass es auch in Gebieten, wo traditionelle Schedulingsysteme versagen, eingesetzt werden kann. Condor kann benutzt werden, um ein Cluster von dedizierten Computerknoten (z.B. Beowulf Cluster) zu managen. Zusätzlich ermöglicht ein einzigartiger Mechanismus, nicht genutzte CPU-Leistung von Desktopmaschinen im Leerlaufmodus nutzbar zu machen. Zum Beispiel kann Condor so konfiguriert werden, Desktopmaschinen nur dann zu benutzen, wenn Tastatur und Maus längere Zeit unberührt bleiben. Sollte Condor bemerken, dass eine Maschine nicht länger verfügbar ist (z.B. sobald entdeckt wird, dass die Tastatur benutzt wurde), ist Condor unter den meisten Bedingungen fähig transparent einen Checkpoint (Speicherung des Auftragszustandes, damit an dieser Stelle weitergemacht werden kann) zu produzieren und den Job auf eine andere Maschine, die sich im Idle-Modus befindet, zu verlagern. Condor braucht kein verteiltes Dateisystem wie z.B. NFS über den Maschinen - denn falls kein verteiltes Dateisystem verfügbar ist, kann es im Namen des Benutzers die Dateien mit den Daten zugehörig zum Job übertragen, oder ihm ist es möglich, alle I/O-Anforderungen des Jobs transparent an die Maschine, welche den Job eingereicht hat, umzuleiten. Als Ergebnis kann Condor benutzt werden, um nahtlos die gesamte Rechenleistung von einer Organisation in eine Ressource zu kombinieren.
Der ClassAd-Mechanismus in Condor bietet ein extrem flexibles und ausdrucksstarkes System

für die Abbildung der Ressourcenanforderungen (Jobs) auf die angebotenen Ressourcen (Maschinen). Jobs können einfach ihre Anforderungen und Vorzüge angeben. Gleichfalls können Maschinen Anforderungen und Vorzüge bezüglich der Jobs, die sie laufen lassen werden, spezifizieren. Mit Hilfe des Vergleichs der Anforderungen und Vorzüge von beiden Seiten, kann im Falle der Übereinstimmung der Job auf der Maschine ausgeführt werden. Diese Anforderungen und Vorzüge können in starken Ausdrücken beschrieben werden, so dass damit die Adaption von Condor zu fast jeder verlangten Strategie resultiert. Condor kann benutzt werden, um Rechenumgebungen im Gridstil aufzubauen, die die administrativen Grenzen überqueren. Condor selbst kann nur innerhalb einer Organisation eingesetzt werden und kann nicht mit anderen Ressourcen, auf denen es nicht installiert ist, zusammenarbeiten. Die sogenannte “flocking” Technologie von Condor erlaubt es nun, dass mehrere Condor-Recheninstallationen zusammenarbeiten. Das Lastenmanagementsystem baut viele von den aufkommenden gridbasierten Rechenmethoden und Protokolle ein. Zum Beispiel ist Condor-G vollständig kompatibel mit Ressourcen, die durch das Globus-Projekt gemanagt werden, so dass eine Integration von Condor in Globus-Grids möglich ist.

2.2.4 Sun Grid Engine

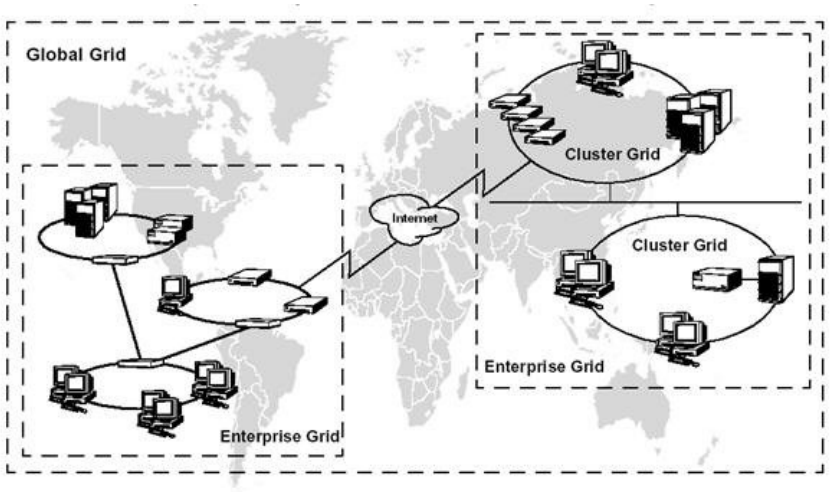

Abbildung 2.3: Ebenen des Grid Computing nach SUN

Sun (siehe Abbildung 2.3), unterteilt Grid Computing in drei logische Entwicklungsebenen:

- **Cluster Grids**

  Die einfachste Form des Grids, ein Cluster Grid besteht aus mehreren Systemen, die über ein Netzwerk verbunden sind. Sun bietet hier die "Sun ONE Grid Engine Software" an. Cluster Grids können verteilte Workstations und Server beinhalten, so wie zentralisierte Ressourcen in einer Datenzentrum-Umgebung. Typischerweise wird es von einer einzelnen Abteilung oder für ein Projekt genutzt. Cluster Grids unterstützen Jobs mit hohem Durchsatz und hoher Performance. Ein allgemein gültiges Beispiel von der "Cluster Grid Architektur" besteht aus Computerfarmen, Gruppen von Multiprozessor-HPC-Systemen, Beowulf Cluster und Network of Workstations (NOWs).

- **Enterprise/Campus Grids**

  Durch das Wachstum der Nachfrage nach nötigen Kapazitäten können mehrere Cluster Grids zu einem Enterprise Grid kombiniert werden. Für diese Art von Entwicklung wird die "Sun ONE Grid Engine Enterprise Edition Software" verwendet. Enterprise Grids ermöglichen mehreren Projekten oder Abteilungen Computerressourcen in kooperativer Art und Weise zu teilen. Sie enthalten typischerweise Ressourcen von mehreren administrativen Domains, aber diese befinden sich im selben geografischen Gebiet.

- **Global Grids**

  Global Grids sind eine Sammlung von Enterprise Grids. Dabei vereinbaren alle Enterprise Grids globale Benutzungsstrategien und Protokolle, aber nicht notwendigerweise der selben Implementierung. Hier wird zusätzlich die Software des Globus Toolkits benötigt.

Rechenressourcen können geografisch verteilt sein und die verbundenen Standorte befinden sich weltweit. Global Grids bieten die Leistung von verteilten Ressourcen für die Benutzer irgendwo in der Welt, deshalb sind sie so entworfen, dass sie die Bedürfnisse von mehreren Standorten und Organisationen Ressourcen zu teilen, unterstützen und behandeln.

Sun bietet eine komplette Cluster-Grid-Solution an. Diese Cluster Grid Architektur kann als Softwarestack angesehen werden, jede Ebene repräsentiert eine andere Funktionalität. Die Sun Grid Engine übernimmt dabei als Stapelverarbeitungssystem die Aufgabe des DRM.
Die Abbildung 2.4 zeigt die Software, welche auf allen Knoten in einem Cluster Grid läuft.

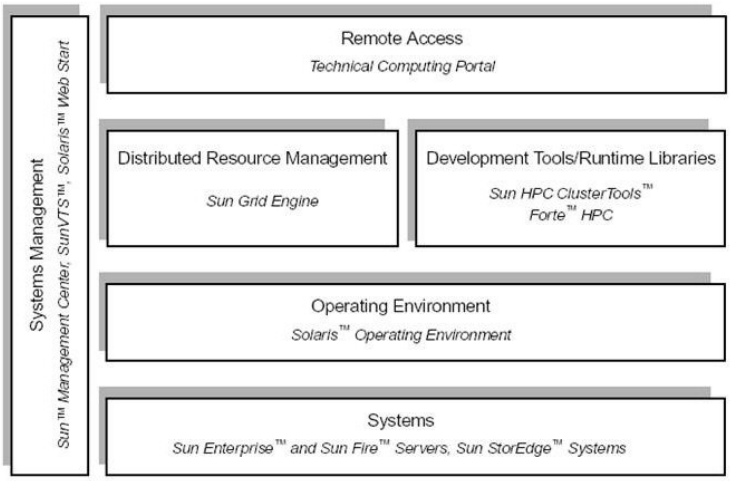

Abbildung 2.4: Software-Stack des Cluster Grid

Es können folgende Komponenten in solch einem Cluster Grid vorkommen:

- Sun Grid Engine Software
- Entwicklungstools und Runtime-Bibliotheken (Sun HPC Cluster Tools, Forte for HPC)
- Technische Computing Portal-Software (Sun ONE Portal Server)
- Systemmanagement-Tools (Sun Management Center, Sun VTS und Solaris JumpStart und Web Start Flash)
- Unterliegende Plattformen (Solaris Operating Environment, Sun Servers und Sun StorEdge Speicherprodukte)

Die Sun Grid Engine ist eine verteilte Ressourcemanagement-Software und gilt als essenzielle Komponente von irgendeinem Cluster Grid. Sie optimiert den Gebrauch von Software- und

Hardwareressourcen in heterogenen vernetzten Umgebungen. Sun Grid Engine fasst die Computerleistung, welche in den Cluster Grids verfügbar ist, zusammen und präsentiert sie als eine einzige Ressource. Dem Benutzer, welcher die CPU-Leistung nutzen möchte, wird ein einfacher Zugriffspunkt auf diesem Grid präsentiert. Sun Grid Engine Software bietet einen zuverlässigen, konsistenten und überall vorhandenen Zugriff auf Leistung mit hohem Durchsatz sowie hoher Rechenparallelität. Sie besitzt die traditionellen DRM-Funktionen wie Warteschlangen, Lastenausgleich, Zugriffsstatistiken der Jobs, benutzerspezifische Ressourcen, Sperren und Wiederaufnehmen von Jobs und clusterweite Ressourcen. Die Software der Sun Grid Engine beinhaltet Verbesserungen wie z.B. die Batch-Aware-Shell (Qtsch), diese erlaubt das Benutzen von interaktiven Anwendungen mit der Sun Grid Engine Software und ressourcenintensive Anwendungen werden automatisch an einen geeigneten Server ohne Eingriff des Benutzers gesendet.

Kapitel 3

Cluster

3.1 Definition und Einordnung

Nach Pfister lässt sich ein Cluster wie folgt beschreiben: “A cluster is a type of parallel or distributed system that consists of a collection of interconnected whole computers, and is used as a single, unified computing resource.”[1] Obwohl dieser Definition nicht allgemein zugestimmt wird, gibt sie eine gute Vorstellung des Konzeptes wieder. Die Bezeichnung “whole computer”, zu sehen in der Abbildung 3.1, repräsentiert einen Knoten, der einen oder verschiedene Prozessoren (P), Cache (C), Arbeitspeicher (M), Festplatte, ein I/O-Interface (I/O) und ein eigenes Betriebssystem (OS) beinhaltet. Ein “whole computer” kann ein PC, eine Workstation, aber auch

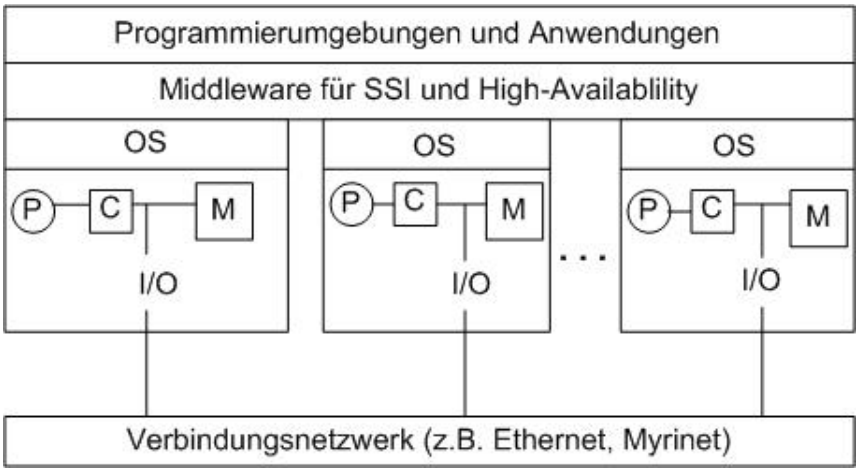

Abbildung 3.1: Typische Clusterarchitektur

ein Symmetric Multiprocessor (SMP) sein. Das Verbindungsnetzwerk (interconnection network) benutzt entweder Standardtechnologien wie Ethernet oder High-Performance-Technologien wie

[1] G.F. Pfister. In Search Of Clusters. Prentice Hall, 1995

Myrinet[2]. Die Anzahl der Knoten variiert von zwei bis zu einigen tausend. Der Begriff "single, unified computing resource" steht für ein Single System Image (SSI). Folgende SSI-Features werden unterstützt:

- ein Eintrittspunkt: Anstatt sich in den einzelnen Knoten des Clusters einzuloggen, erfolgt der Login-Prozess nur am Cluster selbst.
- ein Filesystem
- eine koordinierte Job-Warteschlange und Transparenz des Ortes für Serial Program Parallel System (SPPS)
- ein SSI für die Systemadministration

Typischerweise werden diese SSI-Features oberhalb des Betriebssystems durch die Middleware-Schicht angeboten. Alternativ kann ein Teil der Funktionalität durch das Betriebssystem oder auch durch die Hardware bereitgestellt werden.
Bezüglich der Kommunikationsbiblioteken gibt es zwei Konzepte: Parallel Virtual Machine (PVM, Open Source) und MPI (Standard der Hardwareentwickler). MPI ist weiter verbreitet als PVM. PVM bildet Cluster auf einen Parallelrechner ab und MPI verschickt Nachrichten unter den Knoten (es existieren viele Methoden um Nachrichten an einzelne Knoten, Gruppen oder an alle zu senden).

3.2 Klassifikationsmöglichkeiten

Um eine Klassifikation der Cluster vorzunehmen gibt es verschiedene Möglichkeiten. Cluster können z.B. nach

- der Anwendung (High-Performance-Cluster [HPC], High-Availability [HA]-Cluster),
- dem Besitzer (exklusive Knoten, nicht exklusive Knoten),
- der Hardware (Cluster of Workstations [COW], Cluster of SMP [CLUMPS],...),
- dem Betriebssystem (Linux Cluster [z.B. Beowulf], Solaris Cluster,...),
- der Konfiguration (homogen, heterogen bezüglich der Hardware und dem Betriebssystem),
- der Reichweite des Kommunikationsnetzwerkes (Groups of Clusters [2-99 Knoten], Departmental Clusters [10-100 Knoten], Organizational Clusters [mehrere 100 Knoten], National Metacomputer [mehrere Departmental oder Organizational Clusters], International Metacomputer auf Internetbasis [1000 bis zu Millionen von Knoten])

[2]Ein kommerzielles Produkt, das Technologien übermittelt, welche original in der parallelen Domäne von den Gebieten des SANs (Storage Area Networks) und LANs (Local Area Networks) entwickelt wurden.

klassifiziert werden. Clustersysteme können wieder zu größeren Systemen verbunden werden, damit entstehen Metacluster und Metacomputing.
Ich habe mich hier für die Klassifikation der Cluster nach der Anwendung entschieden und werde im Folgenden High-Performance- und High-Availability-Cluster vorstellen.

3.3 High-Performance-Cluster

3.3.1 Allgemeines

Mit dem Einsatz von High-Performance-Cluster soll eine Verteilung der ankommenden Systemlast erzielt werden. Diese performancegünstigen Cluster laufen im SPPS-Modus oder führen parallele Programme aus. Hier kann eine Unterteilung dieser Clusterart vorgenommen werden.

Dedizierte Cluster

Dedizierte Cluster werden ausschließlich für parallele Maschinen benutzt. Sie treten meist kompakt und homogen auf. Die Knoten befinden sich in einem kleinen technischen Bereich und besitzen keine Peripheriegeräte wie Tastatur und Monitor. Einige zweckbestimmte Cluster basieren normalerweise auf einer schnellen Verbindungstechnologie wie Fast Ethernet, Gigabit Ethernet oder Myrinet.

Campus-Wide-Cluster

Campus-Wide-Cluster werden hauptsächlich durch das Potenzial des "Scavenging" angeregt. Sie verbinden PCs oder Workstations, welche normalerweise für einen Benutzer gedacht sind (Rechner, die sonst auf dem Schreibtisch stehen.). Um die Leerlaufzeiten (idled cycles) zu nutzen, werden diese Rechner über das Verbindungsnetzwerk, welches schon existiert, zusätzlich als Cluster genutzt. Campus-Wide-Cluster sind langsamer als dedizierte Cluster. Sie werden daher für laufende hoch-granulare Anwendungen sowie auch für die Programmentwicklung und Ausbildung gebraucht. Neben der Geschwindigkeit gibt es einen anderen typischen Unterschied zu dedizierten Clustern: das Netzwerk ist offen. Dies bedeutet, dass clusterbezogener und clusterunabhängiger Traffic auf den gleichen Verbindungen entlangläuft. Meist sind Campus-Wide-Cluster von heterogener Struktur. Zusätzlich zu diesen beiden Formen gibt es auch Formen, die Eigenschaften von beiden Clusterarten besitzen.

3.3.2 Detaillierte Beispiele

Beowulf

Dieses System ist benannt nach der englischen Heldenfigur Beowulf, die die Dänen von dem Monster Grendel befreite. Im übertragenen Sinn soll das Beowulf-Cluster die Befreiung von begrenzter Rechenleistung erbringen. Das Beowulf-Projekt begann im Jahr 1994 bei der NASA. Es wurde für das "Earth and Space Sciences project" (ESS) ein Cluster bestehend aus 16 Knoten mit 486DX4 100 MHz Prozessoren, jeweils mit 16 MB Speicher, 540 MB HD, 10 Mbit Ethernetkarten und je drei Busmaster in Betrieb genommen. Der dedizierte High-Performance-Cluster Beowulf setzt sich primär aus handelsüblichen, günstigen und nicht spezialisierten Hardwarekomponenten im Gegensatz zu Massively Parallel Processors (MPPs) à la Cray zusammen. Somit kann der Benutzer sich die Komponenten, auch als Commodity-Sets bezeichnet, selbst kaufen und einbauen. Daher sind Beowulf-Cluster bekannt für ihr gutes Preis-Performance-Verhältnis. Das Clustersystem kann zwischen einem MPP und einem NOW eingeordnet werden. Ein MPP ist ein Rechner, der ein eigenes optimiertes Betriebssystem besitzt und speziell für rechenintensive Anwendungen entwickelt wurde. Dazu zählen die bekannten Cray Supercomputer, wie 3TD oder 3TE. Als sogenanntes "Feierabendcluster" wird ein NOW bezeichnet. Diese vernetzten Computer dienen vorwiegend als Arbeitsplatzrechner und werden in der Zeit, in der sie nicht benötigt werden (meist nachts), als Parallelrechner benutzt. Oft sind sie physikalisch an einem Ort und werden über einen Masterrechner, der die Arbeitsverteilung übernimmt, bedient. Die einzelnen Knoten besitzen in der Regel keine Monitore oder Eingabegeräte. Somit besteht Beowulf aus einem Cluster von PCs oder Workstations, die für die Ausführung von High-Performance-Rechenaufgaben dediziert sind. Zur Zeit reicht die Anzahl der Rechnerknoten von zehn bis zu einigen hundert. Auf dieser Hardware laufen vorwiegend freie Betriebssysteme wie Linux oder FreeBSD, aber es ist auch möglich, ein Beowulf-Cluster unter Windows laufen zu lassen. Die Knoten des Clusters werden über verschiedene Netzwerktechnologien, (Fast-)Ethernet, Hochgeschwindigkeitsnetze wie SCI oder Myrinet verbunden. Die Parallelisierung der Software erfolgt über Message Passing mit MPI oder PVM.

Anwendungen für ein Linux-Beowulf-Cluster

- Quantenchemie
- Kontinuumsmechanik, Hydrodynamik, Wettervorhersage, Crashtest
- Kombinatorische Optimierung
- Messwerterfassung und -auswertung
- Bioinformatik (DNA-Sequenzierung)

- Data-Mining
- Bildverarbeitung, Trickfilm

Konkretes Beispiel eines Beowulf-Clusters An dem Institut für Theoretische Physik der Uni Magdeburg existiert ein Eigenbau des Beowulf-Clusters.
Technische Daten des Beowulf-Cluster TINA:
Es besitzt fünf grundlegende Komponenten.

- zwei separate Ethernets
  Jedes besitzt einen ENTERASYS Matrix E5 Switch. Die Knoten haben keine lokalen HDs, so dass Booten und Speicherzugriff über NFS erfolgt. Der load wird unterteilt in ein Netzwerk für Kommunikation und eins für Systemdienste wie NFS etc.
- Computing Farm
  Diese Computing Farm besteht aus 78 disk-less PCs. Jeder Knoten hat:
  - Dualprozessor Board Gigabyte GA-6VXD7 ATX mit 2*PIII/800Mhz oder Dualprozessor Board Gigabyte Tyan AMD 760 MPX mit 2*Athlon/1200Mhz
  - 1024 MB SDRAM/133
  - 2 3COM 100Mbit Ethernet cards, eine mit BOOT-PROM
  - eine billige Grafikkarte (nur fürs Booten und Debuggen)
  - ein 3.5” Diskettenlaufwerk (Debuggen)
  - und sitzt in einem 19” Gehäuse

  Diese PCs bilden den Kern des parallelen Computers. Es wird nicht beabsichtigt, dass die Nutzer interaktiv auf diesen Knoten arbeiten. Deshalb wird ein Batchsystem die Arbeitsverteilung übernehmen.
- System Server
  Dieser Server bietet verschiedene Dienste für die Knoten an.
  - Dual-Prozessor Board Gigabyte GA-6VXD7 ATX mit 2*PIII/800Mhz
  - 1024 MB SDRAM/133
  - 3COM 100Mbit Ethernet card (uplink zum Campusnetzwerk)
  - 3COM 1000Mbit Ethernet cards (uplink zum Systemnetzwerk)
  - 3COM 100Mbit Ethernet card (redundanter Link zum Kommunikationnetzwerk)
  - SCSI hard-discs, total 90GB
  - eine billige Grafikkarte

  - ein 3.5” Diskettenlaufwerk (nur Debuggen)
  - und sitzt in einem 19” Gehäuse

- Kommunikationsserver
  Dieses System bildet das Frontend, es befasst sich mit dem Batchsystem, der Kompilierung und der Kontrolle durch das WWW.

  - Dual-Prozessor Board Gigabyte GA-6VXD7 ATX mit 2*PIII/800Mhz
  - 1024 MB SDRAM/133
  - 3COM 100Mbit Ethernet card (uplink zum Campusnetzwerk)
  - 3COM 100Mbit Ethernet card (Link zum Kommunikationnetzwerk)
  - 3COM 1000Mbit Ethernet cards (uplink zum Systemnetzwerk)
  - SCSI hard-discs, total 40GB
  - eine billige Grafikkarte
  - ein 3.5” Diskettenlaufwerk (nur Debuggen)
  - und sitzt in einem 19” Gehäuse

Der Beowulf-Cluster TINA findet Anwendung ist der kombinatorischen Optimierung, der statistischen Mechanik und dem Kristallwachstum.

Mosix

Mosix (Multicomputer Operating System for UnIX) ist ein Softwarepaket, welches ein Cluster von x86 basierten Linux-Servern und Workstations (Knoten) formieren kann, dass dann läuft wie ein SMP. Das Mosix Projekt wird an der Hebrew University of Jerusalem entwickelt und gepflegt. Mosix ergänzt Linux um skalierbare Cluster-Funktionen. Dabei ist es möglich alle Dienste und Operationen über das Cluster hinweg zu verteilen. Mosix zählt auch zu der Art von Clustern, welche nicht auf reine Hochverfügbarkeit (High Availability), sondern in erster Linie auf die Verteilung der anfallenden Last über die am Cluster beteiligten Systeme Wert legen. Der Hauptvorteil ist die Einfachheit der Benutzung und eine fast optimale Performance. Zum Beispiel bei der Erzeugung von Prozessen wird Mosix die Prozesse automatisch und transparent dem bestmöglichen Knoten zuordnen (auch wenn nötig neu zuteilen), um die Performance zu maximieren. Der Kern von Mosix sind anpassungsfähige Managementalgorithmen, die auf die Ressourcenanforderungen von allen Prozessen reagieren und diese in Abhängigkeit der verfügbaren und verbreiteten Clusterressourcen überwachen. Die Algorithmen von Mosix benutzen präventive Prozessmigration für:

- automatische Jobverteilung
  Dies wird benutzt für die parallele Verarbeitung oder die Prozessmigration von langsameren zu schnelleren Knoten.

- Lastenausgleich (sogar für die Jobverteilung)
  Mosix benutzt einen präventiven und verteilten Lastenausgleichs-Algorithmus, welcher auf der Unix “load avg metric” basiert. Das bedeutet, dass jeder Knoten versucht einen weniger ausgelasteten Knoten zu finden, um seinen Prozess auf diesen zu migrieren. Folgende Variablen werden zwischen den Mosix-Knoten ausgetauscht, um den verteilten Lastenausgleichs-Algorithmus mit den Informationen zu unterstützen: load (basierend auf einem PII 400MHz), Anzahl der Prozessoren, Geschwindigkeit des Prozessors, genutzter Hauptspeicher, genutzter “Raw Memory” und “Total Memory”.

- Memory Ushering
  Dabei handelt es sich um die Migration der Prozesse von einem Knoten auf einen anderen, um Swapping und Überlastung des Hauptspeichers zu vermeiden. Normalerweise ist nur der Lastenausgleichs-Algorithmus aktiv. Wenn sich der freie Hauptspeicher unter einer bestimmten Schwelle befindet, startet der Memory-Ushering-Algorithmus und löst den Lastenausgleichs-Algorithmus ab. Dieser Algorithmus versucht den kleinsten laufenden Prozess eines Knotens auf einen Knoten mit mehr freien Hauptspeicher zu migrieren. Eventuell werden die Prozesse zwischen Remote-Knoten migriert um nötigen Platz auf einen einzelnen Knoten zu erlangen.

- High-I/O-Performance
  Dies wird durch die Migration eines intensiven I/O-Prozesses auf einen Fileserver, anstatt wie üblich die Daten zum Prozess zu transferieren, erreicht.

- paralleles I/O
  Durch Migrieren paralleler I/O-Prozesse vom Clientknoten zu den Fileservern wird paralleles I/O realisiert.

Um die Load-Informationen zwischen den Knoten auszutauschen und die Prozessmigration zu verhandeln sowie auszuführen, werden die MNPs (Mosix Network Protocols) benutzt. Mosix benutzt UDP, um die Load-Informationen zu übertragen, und TCP, um die Prozesse zu migrieren. Diese Kommunikationen sind nicht verschlüsselt, nicht authentisiert und nicht geschützt, deshalb ist es besser ein privates Netzwerk für MNP zu benutzen (Unglücklicherweise werden die von Mosix benutzten Sockets nicht durch Standardtools wie netstat dokumentiert.). Die Algorithmen von Mosix sind dezentralisiert, jeder Knoten ist gleichzeitig Master für die Prozesse, welche lokal erzeugt werden und Server für die Prozesse, die von anderen Knoten migriert werden. Die Mosix-Algorithmen sind für maximale Performance, overheadfreie Skalierbarkeit und einfache Benutzung ausgelegt.
Trends im Cluster und Grid Computing, in welchen viele Anwendungen aus einer großer Anzahl von Prozessen und/oder großen Datenmengen bestehen (z.B. für genetische und finanzielle Modellierungen) und der dramatischer Abfall der Kosten für handelsübliche Hardware, veranlasste die Entwicklung vom Mosix Parallel I/O (MOPI) Paket. MOPI erweitert Rechenleistung von Mosix mit der Fähigkeit massive parallele I/O zu unterstützen. Dies zielt auf Anwendungen, welche große Datenmengen produzieren müssen, angefangen bei einigen Gbytes bis hin zu Terabytes/s.
Skalierbarkeitsbetrachtungen werden in fast allen Mosix-Algorithmen in Betracht gezogen, z.B. der Probabilistic Information Dissemination Algorithmus. Jeder Knoten sendet in regulären Intervallen Informationen über seine Ressourcen an eine zufällige Anzahl von Knoten. Zur gleichen Zeit hält jeder Knoten einen kleinen Puffer für die zu empfangenen Informationen bereit. Das Ergebnis ist, dass Mosix Konfigurationen mit einer großen Anzahl von Knoten mit minimalem Overhead unterstützen kann. Zum Beispiel das Mosix Parallel I/O (MOPI) Paket kann skalierbare I/O Performance liefern, indem parallele Prozesse auf Knoten migriert werden, die ihre zugehörigen Daten halten, z.B. verschiedene Segmente von einer vorpartitionierten Datei. Zusätzlich zum maximalen I/O-Durchsatz sättigt das Mosix-System nicht das Netzwerk, so dass es fast unendlich skaliert werden kann.
Die weltweite Produktion von Mosix-Systemen reicht von kleinen Clustern, z.B. einigen PCs, Workstations und Servern, welche über das Ethernet verbunden sind, bis hin zu High-End-Clustern mit mehreren hundert Knoten, z.B. SMP und non-SMP Servern, die über das Gigabit-

LAN verbunden sind.
Mit Mosix erfolgt eine Abstraktion der physikalischen Verteilung der verschiedenen Rechner und die laufenden Programme, die mehrere Prozesse benutzen, bleiben bezüglich ihrer Verteilung unverändert. Gegenüber Legion besitzt Mosix als gut ausgestattetes Middleware-System eine einfachere Handhabung, da die Verwaltung der Meta-Objekte entfällt. Das Formieren eines Clusters mit Mosix funktioniert aber nur dann, wenn alle Teilnehmer dieselbe Betriebssystemversion und Mosix-Version besitzen.

openMosix Das openMosix-System ist eine Weiterentwicklung des bestehenden Mosix-Systems unter der GNU General Public License (GPL). Realisiert wird es mit Hilfe einer Erweiterung des Linux-Kernels für Single-Image-Clustering.

3.4 High-Availability-Cluster

3.4.1 Allgemeines

Da immer mehr die Abhängigkeit von Computersystemen zunimmt und z.B. Cluster für viele Probleme Lösungen anbieten können, steigt die Nachfrage nach hochverfügbaren Systemen. Meistens bestimmt die Art der Anwendung wie das High-Availability(HA)-Cluster aussehen soll. Ein HA-Cluster setzt sich aus mindestens zwei Knoten zusammen, die miteinander kommunizieren. Dabei werden ständig Informationen über deren Zustand ausgetauscht und damit überprüft, ob alle Services (Daemons) laufen. Die Kommunikation erfolgt über eine dedizierte Ethernetverbindung oder via serieller Schnittstelle oder eine andere Art der Zweiwege-Verbindung. Es existieren zwei typische Arten:

- Beide Knoten laufen und auf beiden laufen verschiedene Dienste. Somit teilen sich beide Server die Arbeiten und es werden beide Rechenkapazitäten ausgenutzt. Falls ein Rechner ausfällt, muss der andere Rechner seine Arbeit übernehmen und wird langsamer, aber das System funktioniert immer noch. Der Nachteil dieser Variante ist der Abfall der Geschwindigkeit, denn somit können zeitkritische Probleme nicht gelöst werden.

- Ein Rechner ist online, das heißt alle Dienste laufen auf ihm, und der andere Rechner befindet sich im idle-Modus (Leerlauf) oder im Standby. Falls beide Rechner identisch sind, wird immer das gleiche Potenzial erbracht. Der “schlafende” Rechner wird für den Ausfall des laufenden Rechners benötigt. Hier erweisen sich die doppelten Computerkosten eindeutig als Nachteil, denn zusätzliche Computerkapazitäten werden nicht gewonnen.

Für den Zugriff auf den Datenbestand ist ein externes Speichersystem notwendig, z.B. ein SAN oder ein externes SCSI-RAID. Wichtig ist, dass dieser externe Speicher auch im Fall des Versagens einer der beiden Rechner voll einsatzfähig sein sollte, und daher auch betrachtet werden muss. Es ist auch möglich die Anzahl der Rechner von zwei auf mehrere zu erweitern. Wenn die Mittel dazu existieren dies zu realisieren, dann ist es möglich mehrere Knoten im HA-Cluster zu haben. Bei dem erstem Typ sollte aber beachtet werden, dass nicht zwei Rechner auf ein gleiches Volume zugreifen. Des Weiteren erweist es sich nicht einfach die Koordination so zu regulieren, dass beim Ausfall eines Services genau dieser und von genau einem Rechner übernommen wird. Dagegen ist es beim zweiten Typ weniger kompliziert und “nur” abhängig von der Finanzierung. Es erweist sich aber nicht als effizient mehrere “schlafende” Rechner in ein HA-Cluster einzuhängen. HA-Cluster sind dadurch gekennzeichnet, dass nie ein “Single-Point-of-Failure” auftritt. Um eine Hochverfügbarkeit zu liefern, sollten alle Verbindungen, Datenspeicher, Stromversorgung und auch das Netzwerk redundant vorhanden sein.

3.4.2 Ausgewählte Clustersysteme

LinuxVirtualServer (LVS)

Das LinuxVirtualServer-Projekt steht unter GPL und ist ein wenig mehr als ein reiner Hochverfügbarkeitscluster. Neben der reinen Steigerung der Serververfügbarkeit, bietet dieser Cluster auch eine Lastverteilung der Dienste. Die Architektur des Clusters ist für den Endbenutzer transparent, nur ein virtueller Server ist sichtbar. Dieser virtuelle Server unterteilt sich in einen Lastverteilungsserver als Frontend (den man aus Verfügbarkeitsgründen ebenfalls redundant auslegen sollte) und mehreren Clientmaschinen, auf denen die entsprechenden Dienste laufen. Die Clientrechner sind untereinander über ein Hochgeschwindigkeits-LAN oder ein geografisch verteiltes WAN (Wide Area Network) verbunden. Alle eingehenden Anfragen kommen bei den Lastverteilungsserver an und werden von diesen an die Dienstserver verteilt. Er lässt dabei die parallel aufgesetzten Dienste auf den Clientrechnern als virtuellen Dienst auf einer einzelnen IP-Adresse erscheinen. Durch das transparente Hinzufügen/Entfernen von Knoten in dem Cluster wird Skalierbarkeit realisiert. High-Availability wird durch die Lokalisation von Knoten- und Daemonfehlern und der entsprechenden Rekonfiguration des Clusters möglich. Viele Lastverteilungssysteme basieren auf dem LVS.

Suchmaschine Google

Um mehr als 200 Millionen Suchen pro Tag durchführen zu können, benötigt Google entsprechende Technologien die Schnelligkeit und auch Ausfallsicherheit anbieten. Deshalb steckt hinter Google ein Linuxcluster mit mehr als 10000 Servern (PCs, single-processor, Red Hat Linux, 256MB RAM, 80GB HD), der weltweit der größte kommerzielle Linuxcluster ist. Google besitzt drei Datenzentren, jedes davon besitzt ein identisches Image der Daten. Google benutzt RoundRobin-DNS, um den Traffic unter den Datenzentren zu lenken. Für ein besseres Trafficmanagement wird das Border Gateway Protocol verwendet. Google benutzt sein eigenes Trafficmanagement und seine eigene Loadbalancing-Software, um den Traffic an den besten Server zu lenken. Der Index des Webs (beträgt mehr als 3 Milliarden Einträge) wird in einzelne Teile zerlegt, wobei jeder Teil des Indexes für Redundanz und Ausfallsicherung zu einem Cluster von ca. 40 Servern verteilt wird. Die Trafficmanagementtools haben ihre größte Ausarbeitung einmal im Monat, wenn Google seinen Index updatet und 10 Terabytes Daten über das LAN zu jeden Server übertragen muss. Die Server sind mit 100Mbit/s verbunden (FastEthernet) innerhalb eines Racks. Diese Racks sind dann untereinander mit 1Gbit/s verbunden. Alle Google-Server besitzen die selbe Software-Konfiguration, daher ist das gesamte System leicht zu administrieren, denn jeder Rechner kann jede beliebige Aufgabe übernehmen. Da das System von Google ständig wächst, wird auch für die Rechner jede Menge an Platz gebraucht. Die Server von Google werden von zwei kleinen Herstellern geliefert: Rackable Systems und King Star Computer.

Zwei Rackable-Maschinen passen in eine einzelne 1U-Rackeinheit. Hier wird die Flexibilität und Erweiterbarkeit solch eines Clustersystems deutlich gemacht.

Kapitel 4

Vergleich und Fazit

Einen Vergleich zwischen Cluster und Grid aufzubauen, erweist sich nicht als einfach, da es zwischen beiden "Klassen" einige Überschneidungen gibt. Grob gesagt ist ein Grid ein Cluster im großen Umfang. Hauptelemente des Grids sind die Ressourcen und bei den Cluster spielen die Rechnerknoten (Nodes) die Hauptrolle.
Grids ermöglichen das Suchen nach freien Ressourcen, auch "Scavenging" genannt, analog zu den Clustern, jedoch in einem größeren Ausmaß. Da Grids mehrere Ressourcen verbinden und mehrere Anwendungen zur gleichen Zeit laufen lassen, ist es ihnen möglich so genannte "Hot Spots" von verschiedenen Anwendungen auszugleichen. Im Vergleich zu Clustern ist der Grad der Kopplung und die Kommunikationsgeschwindigkeit der Grids niedriger. Nach der Definition von Kesselman (siehe Abschnitt 2.1) bietet ein Grid eine nahtlose Integration von Ressourcen, so dass das Grid ähnlich dem SSI von Clustern als eine einheitliche Ressource dem Benutzer erscheint. Jedoch wird dies durch die vorwiegend heterogene und dezentrale Struktur des Grids, im Gegensatz zur oft homogenen und zentral gesteuerten Clusterstruktur, zunehmend schwieriger. Die vorgenommene Klassifizierung in High-Performance- und High-Availability-Cluster ist nur eine Möglichkeit. Es zeigt sich in der Realität, dass oft auch eine Komposition aus beiden existiert. Bei der Verwendung von Clustern kann der Benutzer meist bei seiner gewöhnlichen Rechnerumgebung bleiben. Oft können für den Aufbau von Clustern gebräuchliche Komponenten, sogenannte Commodity-Sets, genutzt werden, so dass es für den Clusterbenutzer kostengünstiger ist. Damit eine Aufgabe mittels eines Clusters bearbeitet werden kann, sollte vorher geprüft werden, ob sie parallelisierbar ist. Ein limitierendes Element der Klasse Cluster ist die Übertragungsrate im Netz, denn für ein Cluster spielt die Geschwindigkeit der Datenübertragung eine größere Rolle als bei Grids. Die Ressourcen des Grids sind global bzw. geografisch verteilt, so dass der dezentrale Charakter des Grids die Kommunikationsgeschwindigkeit sowieso begrenzt.
Nicht zu übersehen sind auch die Faktoren Energieaufnahme und der Platzbedarf für eine effiziente Nutzung des Clusters. Im Gegensatz dazu arbeiten Grids ressourcensparender, denn mit Hilfe der Bildung von virtuellen Organisationen existiert keine unnötige Redundanz an Ressourcenan-

geboten. Die hier vorgestellten Gridsysteme sind meist durch eine wissenschaftliche Zielstellung entstanden und besitzen jetzt noch wissenschaftlichen Charakter. Zum Beispiel das System Legion setzt im objektorientierten Bereich an, ist aber aufgrund der Tatsache, dass alles als Objekt modelliert werden muss, viel zu aufwändig. Das Globus-Projekt ist als Grid-Middleware-System aufgrund seines modularen Aufbaus sowie seinen geringen Prämissen an die interne Struktur und an das Programmiermodell universell einsetzbar.
Bei der Gegenüberstellung der beiden Klassen Grid und Cluster wird ersichtlich, dass sich Grid Computing noch in der Entwicklung befindet. Die Entscheidung der Verwendung eines Grids oder eines Clusters hängt vorwiegend von der Art und dem Umfangs der zu lösenden Aufgabe sowie von der vorhandenen technischen Umgebung ab.

Literaturverzeichnis

[Beowulf Cluster1] Beowulf Allgemeines -
URL: http://www.beowulf.org/

[Beowulf Cluster2] Beowulf FAQ -
URL: http://www.canonical.org/~kragen/beowulf-faq.txt

[Beowulf Cluster3] Tina: a Beowulf-Super-Computer -
URL: http://tina.nat.uni-magdeburg.de

[Beowulf Cluster4] Heiko Bauke: Linux-Beowulf-Cluster: Herausforderung im High Performance Computing and Networking(2001) -
URL: tina.nat.uni-magdeburg.de/heiko/Publications/linuxtag2001_folien.pdf

[Cluster1] Gregory F. Pfister. In Search Of Clusters. Prentice Hall, 1995

[Cluster2] LinuxCluster im Auftrieb - ein Überblick (2000) -
URL: http://www.linuxinfo.de/de/db/lnxi-shkommentar.php3?eid=63

[Cluster3] Linux Clustering Software (2002) -
URL: http://freshmeat.net/articles/view/458

[Condor1] What is Condor? -
URL: http://www.cs.wisc.edu/condor/description.html

[Condor2] How did the Condor project start? -
URL: http://www.cs.wisc.edu/condor/background.html

[DataGrid] What is DataGrid? -
URL: http://web.datagrid.cnr.it/LearnMore/LearnMore1.jsp

[Globus Project1] Ian Foster, Carl Kesselman: The Globus Project: A Status Report (1998) -
URL: ftp.globus.org/pub/globus/papers/globus-hcw98.pdf

[Globus Project2] Globus 2.0 Overview -
URL: http://www.globus.org/gt2/admin/guide-overview.html

[Globus Project3] The Globus Project: Frequently Asked Questions -
URL: http://www.globus.org/about/faq/general.html

[Globus Project4] White Paper GridFTP: Universal Data Transfer For The Grid (2000) -
URL: www.globus.org/datagrid/deliverables/C2WPdraft3.pdf

[Google1] Linux Cluster Google -
URL: http://www.google.com/press/highlights.html

[Google2] Google Bets The Farm On Linux(2001) -
URL: http://www.internetwk.com/lead/lead060100.htm

[Grid Computing1] Ian Foster, Carl Kesselman, Steven Tuecke: The Anatomy Of The Grid: Enabling Scalable Virtual Organizations (2001) -
URL: www.mcs.anl.gov/globus/research/papers/anatomy.pdf

[Grid Computing2] Ian Foster, Carl Kesselman: Computational Grids (1985) -
URL:dsl.cs.uchicago.edu/Courses/CS347/../../papers/gridbook_chapter2.pdf

[Grid Computing3] IBM: The Era Of Grid Computing: A New Standard For Successful IT Strategies (2003) -
URL: www.ibm.com/grid/pdf/it_exec_brief.pdf

[Grid Computing4] IBM: The Era of Grid Computing: Enabling New Possibilities For Your Business (2003) -
URL: www.ibm.com/grid/pdf/business_exec_brief.pdf

[Legion] Legion: A Worldwide Virtual Computer -
URL: http://www.cs.virginia.edu/~legion/

[LinuxVirtualServer] Linux Virtual Server Project -
URL: http://www.lvserver.de/german/index.html

[Mosix1] Mosix -
URL: http://page.inf.fu-berlin.de/~materlik/mosix/praesentation/html/slide_1.html

[Mosix2] Automatic Load Balancing And Transparent Process Migration (2000) -
URL: http://www.sissa.it/~inno/pubs/mosix.pdf

[Mosix3] openMosix FAQ General -
URL: `http://howto.ipng.be/openMosixWiki/index.php/FAQ`

[Paralleles und Verteiltes Rechnen] - Claudia Leopold: Parallel And Distributed Computing. Wiley-Interscience Publication, 2001

[SunGridEngine] Sun ONE Grid Engine Software -
URL: `http://wwws.sun.com/software/gridware/`

[SunClusterGrid] Sun Cluster Grid Architecture: A technical white paper describing the foundation of Sun Grid Computing (2002) -
URL: `wwws.sun.com/software/grid/SunClusterGridArchitecture.pdf`