Verlässliche, adaptive Informationssysteme: Adaptierung durch dynamische Bereitstellung von Ressourcen

Sven Breuner

Technische Universität Kaiserslautern, Germany

Abstract. Die zunehmende Leistungsfähigkeit von Clients gegenüber Servern, vor allem jedoch die sehr hohe Anzahl von gleichzeitigen Nutzern einzelner Dienste, die sich durch die weite Verbreitung des Internets ergeben hat, sind nur zwei von vielen Beispielen, die neue Umgangsweisen mit der so entstehenden Last erfordern. Aufgrund der zum Teil massiven Schwankungen, denen die Nutzerzahlen solcher Dienste unterliegen, besteht ein großer Bedarf an Systemen, die sich derartigen Schwankungen anpassen können und gleichzeitig ein möglichst hohes Maß an Zuverlässigkeit bieten.
Das Problem der Last wird in verschiedensten Ausprägungen betrachtet, wobei jeweils in Frage kommende Lösungsmöglichkeiten vorgestellt und diskutiert werden. Neben Standardverfahren wie der lokalen Lastverteilung durch Gateways werden so auch momentan noch nicht voll etablierte, aber vielseitig einsetzbare Konzepte wie Virtualisierung und Grids behandelt.

1. Einleitung

Last kann bei Computern in verschiedensten Formen entstehen. Sie kann sich beispielsweise in einem hohen Verbrauch von Netzwerkbandbreite aufgrund extensiver Kommunikation mehrerer Prozesse miteinander äußern, durch einen zeitkritischen Bedarf an großen Mengen gespeicherter Daten auf einem Fileserver oder einfach durch komplexe Programme, die eine hohe Beanspruchung von CPU-Zeit ergeben. Alle diese Arten von Last existieren zwar schon mehrere Jahrzehnte, spätestens jedoch durch die weite Verbreitung des Internets wurden sie in eine vollkommen neue Größenordnung gebracht.

Zur Einstellung auf diese neuen Größenordnungen der Last wurden bereits diverse Lösungen entwickelt, von denen jedoch keins der Systeme allgemeinen Ansprüchen gerecht werden kann. Obwohl der Begriff der Last selbst im Folgenden immer möglichst abstrakt benutzt wird, ist es daher erforderlich, die verschiedenen Ausprägungen von Last zu erkennen und genauer zu betrachten, um anhand dieser jeweils angemessene Systeme zur Lastbewältigung zu diskutieren. Einige wirksame Möglichkeiten zur simplen Verteilung von Last werden im nächsten Kapitel vorgestellt.

Zusammen mit der Benutzerzahl von Systemen haben aber nicht nur die Anforderungen an die Antwort- und Bearbeitungszeiten, sondern auch die Ansprüche

an die Verfügbarkeit der Systeme zugenommen. Zur Gewährleistung der Verfügbarkeit ist es damit keine ausreichend befriedigende Lösung mehr, Dienste lediglich durch mehrere Fallback-Server zu sichern. Stattdessen werden an die Dienste selbst neue Qualitäts- und Stabilitätsansprüche gestellt, die sich zum Teil nur durch die Verwendung genau abgestimmter oder im Dauerbetrieb bewährter Konfigurationen erreichen lassen. Diesem Thema widmet sich daher das dritte Kapitel mit einer Einführung und den möglichen Anwendungsgebieten von Virtualisierung.

Das Internet hat aber nicht nur neue Arten von Last aufgeworfen. Dass in diesem besonderen Netzwerk hingegen sogar ein vorher nicht vorstellbares Potential zur Bewältigung von Last liegt, haben wissenschaftliche Einrichtungen bereits vor einigen Jahren erkannt und erfolgreich genutzt. Vor diesem Hintergrund und dem Bedarf nach einem vollkommen dynamischen Weg, diverse Arten von Last zu bewältigen, wird im vierten Kapitel die neue Infrastruktur „Grid“ vorgestellt, die speziell auf die weltweite Vernetzung durch das Internet zur dynamischen Bewältigung von Last aufbaut.

2. Load balancing

Das klassische Client-Server-Modell, bei dem ein Server die Anfragen mehrerer Clients behandelt, wird aufgrund seiner Einfachheit bezüglich der klaren Strukturierung und Aufgabenverteilung oft zunächst anderen Modellen, wie etwa Peer-to-Peer-Netzwerken, vorgezogen. Es brachte jedoch von Anfang an einen entscheidenden Nachteil mit sich: die schlechte Skalierbarkeit.

Während die Anzahl der gleichzeitig aktiven Clients in den ersten Jahren dieses Modells jedoch aufgrund der Gegebenheiten lokaler Netzwerke immerhin noch überschaubar geblieben ist, hat sich die Situation spätestens seit der weiten Verbreitung des Internets entscheidend geändert. Hochgradig gefragte Internetseiten etwa verzeichnen heutzutage mehrere hundert Millionen Zugriffe täglich. Einer der Spitzenreiter ist hierbei *Ebay*[1] mit bis zu 889 Millionen Zugriffen am Tag. Ein solches Szenario kann nicht von einem einzelnen Server bewältigt werden. [Höv05]

Hinzu kommt außerdem, dass inzwischen das Leistungsverhältnis von Client und Server in der Regel wesentlich ausgeglichener ist. Aus den ehemals verhältnismäßig schwachen Clients sind leistungsfähige PCs und Workstations mit entsprechend schnellerer Netzanbindung geworden, die in wesentlich kürzerer Zeit komplexe Anfragen stellen und deren Benutzer angemessen kurze Bearbeitungs- bzw. Antwortzeiten erwarten. Einen der ungünstigen Fälle bilden hier die Webservices, bei denen die Anfragen auf der für die Systemunabhängigkeit und die menschliche Lesbarkeit optimierten *Extensible Markup Language*[2] (XML) basieren. Hier müssen die Anfragen zunächst erst einmal geparst und in eine „Maschinen-verständliche“ Form gebracht werden, bevor überhaupt die eigentliche Verarbeitung beginnen kann.

---

[1] http://www.ebay.de, Stand Dezember 2005

[2] http://www.w3.org/XML, Stand Dezember 2005

Ein ebenfalls sehr wichtiger Punkt, der dennoch nicht von allen Serverbetreibern berücksichtigt wird, wie immer wieder Beispiele zeigen, ist die Ausfallsicherheit eines Systems. Einzelne Server stellen einen „Single Point of Failure“ dar, bei deren Ausfall das gesamte System funktionsunfähig wird.

Ein möglicher Lösungsansatz für diese Art von Problemen ist die Vermeidung des klassischen Client-Server-Modells und damit die Wahl einer besser skalierenden Architektur, wie beispielsweise einem Peer-to-Peer-Netzwerk.

2.1. Lastverteilung durch besondere Softwarearchitektur: Peer-to-Peer-Netzwerke

Die Aufgabe des Servers wird in einem Peer-to-Peer-Netzwerk prinzipiell von jedem der Peers übernommen, wodurch das Problem des einzelnen Servers, der eine sehr großen Anzahl von Clients bedienen muss, zunächst aufgehoben ist. Die gute Skalierbarkeit eines solchen Systems liegt also bereits in seiner Natur begründet, denn mit jedem neuen „Client“, der Anfragen generiert, kommt zugleich auch ein neuer „Server“ zum System hinzu, der einen Teil zur Bewältigung der Gesamtlast des Systems mit beiträgt. Ein solches System funktioniert gerade deswegen überhaupt, weil die Clients bzw. Peers ein so großes Potenzial besitzen, dass sie ohne weiteres nebenbei die Anfragen anderer Peers verarbeiten können.

Der zweite große Vorteil, den solche Netzwerke mit sich bringen, ist die Ausfallsicherheit. Da die Teilnehmer für gewöhnlich Computer sind, die entweder über keine permanente Internetverbindung verfügen oder bei denen die jeweilige Software nicht ununterbrochen läuft, wird das ständige Wegfallen und Hinzukommen von Peers von Anfang an bei der Entwicklung berücksichtigt und die verwalteten Daten dementsprechend redundant, oder der Zugriff auf die Daten fehlertolerant gestaltet.

Erfolgreiches Beispiel für solche Netzwerke sind die unter dem Begriff „Internettauschbörsen“ bekannten Programme wie *eMule*[3] oder *Kazaa*[4]. Sie stellen im Grunde verteilte Dateiarchive, also Dateisysteme mit nicht veränderbaren Dateien, dar. Der Wegfall von Peers wirft hier kein Problem auf, da Dateien gegebenenfalls einfach von anderen Peers kopiert werden können oder anderenfalls unbegrenzt gewartet wird, bis ein Peer mit einer Kopie der entsprechenden Datei wieder im System auftaucht. Verzeichnisinformationen zum Auffinden von bestimmten Dateien werden zur Performancesteigerung in dedizierten Servern gespeichert, liegen aber zusätzlich noch in verteilten Hash-Tabellen und sind somit ebenfalls unempfindlich gegenüber dem Ausfall von Servern oder Peers.

Obwohl diese Art von Netzen für gewisse Aufgaben ideal zu sein scheint, bringen sie jedoch auch entscheidende Nachteile mit sich. So ist der Versuch, solche Systeme flexibler für andere oder gar diverse Arten von Aufgaben zu konstruieren, aufgrund ihrer Komplexität kaum empfehlenswert. Auch die Zurückverfolgung oder die Korrektur eines Fehlers gestaltet sich aus diesem Grund als äußerst schwierig. Hinzu

[3] http://www.emule-project.net, Stand Dezember 2005
[4] http://www.kazaa.com, Stand Dezember 2005

kommt, dass nicht garantiert werden kann, dass Peers auch wirklich die Serveraufgaben übernehmen und sich nicht nur unfair als ausschließliche „Clients“ am System beteiligen. Dadurch entsteht häufig ein erhebliches Ungleichgewicht und es bilden sich Warteschlagen bei den korrekten Peers, in denen die Teilnehmer lange auf die Beantwortung ihrer Anfragen warten müssen [Ada00]. Nicht zuletzt spielt auch die mangelnde Anonymität eine Rolle, denn durch das Auslesen der üblicherweise offenen Verzeichnisinformationen kann der Datenaustausch leicht von Dritten überwacht werden [Hei06]. Die folgende Abbildung stellt ein solches Netzwerk stark vereinfacht dar.

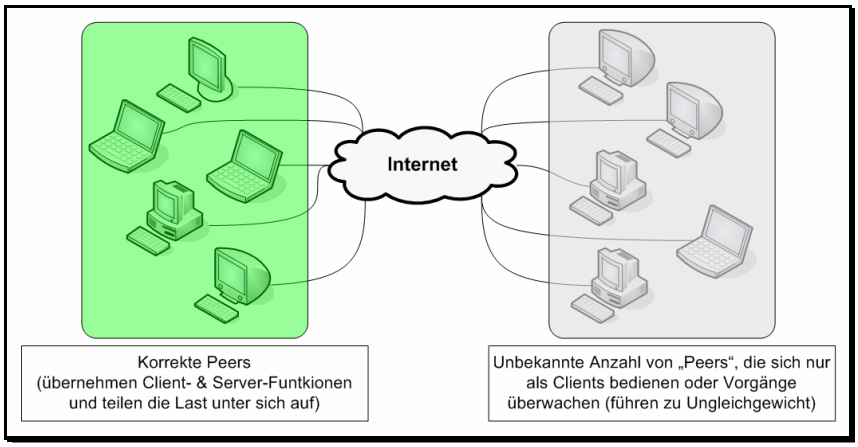

Abb. 1. Lastverteilung im Peer-to-Peer-Netz

Alle diese Umstände machen ein derartiges System für kommerzielle Zwecke mit hohen Qualitätsansprüchen kaum nutzbar. Dennoch stellt die Aufteilung der Last gemäß dem Leitsatz „Teile und herrsche“ die einzig sinnvolle Art dar, mit den eingangs erwähnten Problemstellungen umzugehen. Dazu wird jedoch eine hierarchischere Struktur der Systeme benötigt, in denen Aufgaben klarer verteilt und Vorgänge automatisiert oder von Administratoren und Entwicklern zuverlässig und nachvollziehbar überwacht werden können. So können Informationen für unbefugte unzugänglich gehalten werden und zugleich die geforderten Qualitätsansprüche an das Gesamtsystem gewährleistet werden.

Wie ein solches System massiv verteilt aussehen kann, wird im Kapitel „Grids“ genauer erläutert. Bereits innerhalb einer lokalen administrativen Domäne, also in einem LAN, gibt es jedoch wirksame Möglichkeiten zur Verteilung von Last, wie das folgende Kapitel zeigen wird.

2.2. Lastverteilung durch zentrale Instanz: Gateways

Auch wenn zahlreiche Organisationen, wie etwa Universitäten oder große Unternehmen, das Recht auf die Benutzung ganzer IP-Adressräume des Internets haben und somit nicht auf einen zentralen Internet-Zugangspunkt zur *Network Address Translation*[5] (NAT) angewiesen sind, so gibt es auch bei diesen Organisationen üblicherweise einen Gateway: einen zentralen Punkt bzw. Host im Netzwerk, über den der Internet-Traffic läuft.

Ein solcher Gateway ist aus Sicherheitsgründen üblicherweise eine Hardware- oder Software-Firewall, die aufgrund von definierbaren Regeln Entscheidungen trifft, ob und über welche Routen Pakete weiterzuleiten sind. Hierzu findet also bereits eine Analyse des Traffics in Echtzeit statt. Diese Tatsache kann man sich zu nutze machen, um eine Lastbalancierung zu erzielen.

Dafür ist es wichtig, dass der Datenaustausch über das Internet häufig eine bestimmte Eigenschaft hat, die bei dem oft verwendeten *Hypertext Transfer Protocol*[6] (HTTP) per Definition gegeben ist: gemeint ist das zustandslose Anfrage/Antwort-Schema. Hierbei besteht der Datenaustausch zwischen Client und Server lediglich aus einer Folge voneinander unabhängiger Anfragen an den Server (z.B. dem Aufruf einer bestimmten Webseite durch Übertragung der URL) und einer direkt darauf folgenden Antwort des Servers (z.B. durch das Senden der angeforderten Webseite). Die nächste Anfrage, die von demselben Client kommt, kann damit prinzipiell problemlos von einem anderen Server beantwortet werden, denn die jegliche vorherige Kommunikation des Clients ist für die Beantwortung der nächsten Anfrage irrelevant.

Werden also statt einem einzelnen HTTP-Server vom Betreiber mehrere HTTP-Server benutzt, kann in diesem Szenario somit bereits eine Lastverteilung dadurch herbeigeführt werden, dass der Gateway jeden HTTP-Request an einen anderen Server weiterleitet. Dieses Verfahren lohnt sich aber kaum bei statischen Webseiten, da hier die Anbindung an das Internet bzw. der Dateisystemzugriff die beschränkenden Faktoren sind. Es wird daher bei rechenintensiveren Anwendungen eingesetzt. Beispiele sind dynamische Webseiten, die auf Skriptsprachen basieren und zur Laufzeit erst übersetzt werden oder auch Online-Routenplaner. Es empfiehlt sich hierbei, die Datenquelle von der Geschäftslogik zu trennen, damit alle Server auf dieselben Daten zugreifen können. Die folgende Abbildung stellt einen solchen Aufbau schematisch dar.

[5] http://www.ietf.org/rfc/rfc1631.txt, Stand Dezember 2005
[6] http://www.ietf.org/rfc/rfc1945.txt, Stand Dezember 2005

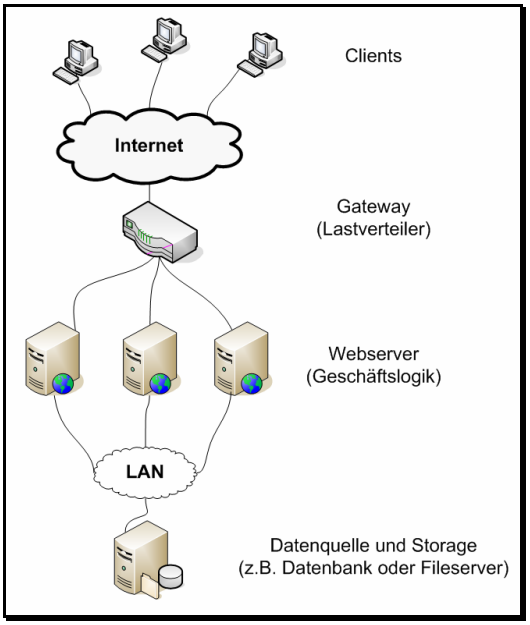

Abb. 2. Lastverteilung über einen Gateway

Oft genügt die serverseitige Zustandslosigkeit jedoch nicht. Sobald es auf einer Webseite einen Warenkorb oder einen anderen Grund dafür gibt, weswegen Daten über eine einzelne Anfrage hinaus einem Client zugeordnet werden müssen, wird eine Möglichkeit benötigt, diese Daten auf der Serverseite etwas längerfristig zu erhalten.

Eine Lösung wäre hier, die Session-Daten in der gemeinsamen Datenquelle zu speichern. Damit sind sie unabhängig davon, welcher Server als nächstes den HTTP-Request beantwortet und können über eine vom Client in der URL übertragene Session-ID wieder gefunden werden. Das belastet allerdings verstärkt die Datenquelle.

Die Alternative existiert gewissermaßen in abgewandelter Form bereits wieder im Gateway, denn Gateways operieren bereits mit Hilfe von Zuständen. Sie merken sich beispielsweise die variablen Zuordnungen von den in OSI[7]-Layer 3 befindlichen IP-Adressen, die zum Routing im Internet verwendet werden, zu den entsprechenden Interfaces auf der LAN-Seite und den OSI-Layer 2-Adressen, die zur Adressierung im LAN verwendet werden. Ein weiteres Beispiel sind NAT-Gateways, die dynamisch in der Lage sein müssen, der vorherigen ausgehenden Kommunikation entsprechend eintreffende Pakete an den richtigen Empfänger im LAN weiterzuleiten. Auch hier werden wieder Zustände im Gateway benötigt.

---

[7] *Open Systems Interconnection Reference Model*, http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip, Stand Dezember 2005

Erweitert man den Gateway dahingehend, dass er den Traffic auf einem höheren OSI-Layer versteht oder einfach in einem bestimmten Zeitraum aufeinander folgende Anfragen desselben Clients immer wieder an denselben Server weiterleitet, so können in diesem System problemlos serverseitige Sessions verwendet werden. Risiko hierbei ist allerdings ein eventuell entstehendes Last-Ungleichgewicht auf den Servern, da gewisse Clients mehr Last erzeugen als andere.

Die Fähigkeiten des Gateways lassen sich aber zusätzlich noch auf diverse Arten erweitern, aufgrund derer auch die Lastverteilung im System entscheidend verbessert werden kann. Kommuniziert der Gateway beispielsweise mit den Servern über ein gesondertes Protokoll, so kann er von ihnen Statusinformationen bezüglich ihrer Auslastung erfahren und mit diesem Wissen gezielt Anfragen an weniger belastete Server weiterleiten. Bei einem Serverausfall würde er damit auch keine Anfragen mehr an diesen Server weiterleiten, was ein wichtiger Schritt in Richtung Robustheit bzw. Ausfallsicherheit ist.

Die fortgeschrittene Lastbalancierung ist vor allem bei rechenintensiveren Anfragen effektiv. In einem solchen Fall kann der Lastausgleich aber auch auf einer anderen Ebene erfolgen, wie das nächste Kapitel verdeutlichen wird.

2.3. Lastverteilung durch nachträglichen Ausgleich: openMosix

Soll Last in einem Verbund von Computern verteilt werden, wird die Entscheidung der Zuteilung von Aufgaben häufig im Voraus getroffen. Sei es, wie im letzten Kapitel, auf Basis einer vorgeschalteten Instanz, die Anfragen bei ihrem Eintreffen unmittelbar an bestimmte Server weiterleitet, oder, wie bei einem High-Performance-Computing-Cluster, durch einen Scheduler, der Jobs zu gegebener Zeit an freie Nodes verteilt.

Dieses Vorgehen hat einen Nachteil: Die zuteilende Instanz kann im Voraus nicht wissen, wie viel Last der von ihr zugeteilte Job zur Laufzeit später tatsächlich verursachen wird. Das Projekt *openMosix*[8] verfolgt daher einen anderen Lösungsansatz, der zu einer effizienteren Ausnutzung der zur Verfügung stehenden Ressourcen, vor allem bei rechen- oder hauptspeicherintensiven Anwendungen, führen soll. [BarM03]

Die Grundidee bei openMosix ist, dass Prozesse erst zur Laufzeit zeigen, welche Last sie verursachen. Daher kann auch erst zur Laufzeit entschieden werden, auf welchem Host ein Prozess am sinnvollsten bearbeitet werden kann – nämlich auf dem Host, auf dem die aktuell laufenden Prozesse momentan die geringste Last erzeugen. Voraussetzung für die Ausnutzung dieser Erkenntnis ist also, dass Prozesse zur Laufzeit auf einen angemessenen Host verschoben werden können.

Zu diesem Zweck ist openMosix als Patch für den Linux-Kernel umgesetzt worden. Das ermöglicht genau die beschriebene Migration von Prozessen, abhängig von der momentanen Auslastung der einzelnen Hosts. Diese Auslastung kann zum einen eine hohe Beanspruchung der CPU sein, zum anderen aber auch ein großer Verbrauch von Arbeitsspeicher und daraus resultierendem Swapping, was ebenfalls

[8] http://openmosix.sourceforge.net, Stand Dezember 2005

ein Grund zur Migration wäre, sofern ein anderer Host über mehr freien Arbeitsspeicher verfügt und damit den Prozess besser bearbeiten könnte. openMosix übernimmt hierbei die gesamte Clusterbildung und auch automatisch die Entscheidung zur Migration, bietet aber gleichzeitig die Möglichkeit der Überwachung und des manuellen Eingriffs an. Die folgende Abbildung verdeutlicht die Lastverteilung in einem solchen System.

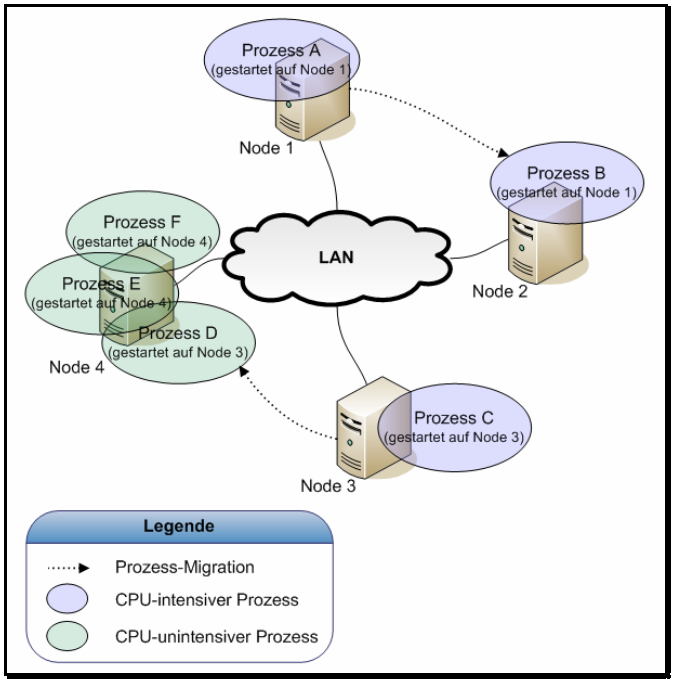

Abb. 3. Lastverteilung bei openMosix

Es gibt allerdings auch Situationen, in denen sich die Verschiebung eines Prozesses ungünstig auswirken würde. Das ist beispielsweise der Fall, wenn ein Prozess auf lokale Rechnerressourcen, wie die Festplatte oder andere Hardware, zugreift. openMosix kann dann die Zugriffe zwar an den ursprünglichen Host weiterleiten, so dass dem Prozess stets sein ursprüngliches Environment zur Verfügung steht, bei zahlreichen Zugriffen dieser Art wirken sich diese Weiterleitungen aber nachteilig auf die Performance aus, und der Prozess muss wieder auf seinen ursprünglichen Host migrieren.

Laufen auf einem Rechnerverbund für den Einsatz von openMosix günstige, also länger andauernde, gleichbleibend CPU- oder arbeitsspeicherintensive Prozesse, dann bietet openMosix eine sehr geschickte, elegante und vor allem effiziente Ausnutzung

der zur Verfügung stehenden Nodes, die aufgrund der Voraus-Entscheidung bei anderen Ansätzen kaum erreicht werden kann.

2.4. Zusammenfassung

In diesem Kapitel wurden drei grundlegend verschiedene Arten der Lastbewältigung vorgestellt, von denen jede wichtige Vor-, aber auch Nachteile mit sich bringt. Jede Möglichkeit lässt sich also sinnvoll für bestimmte Zwecke einsetzen. Diese Zwecke sind jedoch so unterschiedlich, dass es nicht angebracht wäre, die Lösungen direkt miteinander zu vergleichen. Dennoch ist die Gemeinsamkeit aller Lösungen eine Verteilung der Last auf mehrere Hosts, wobei zum einen die Organisation der Hosts, zum anderen aber auch die Qualität der Auswertung von entstandener und zu erwartender Last eine entscheidende Rolle spielt.

Ein weiterer wichtiger Faktor ist jedoch auch die resultierende Zuverlässigkeit des Gesamtsystems, denn ein unzuverlässiges System macht auch bei einer guten Skalierbarkeit oder Lastverteilung keinen Sinn. Mit dieser Erkenntnis wird im nächsten Kapitel ein Verfahren vorgestellt, das eine solide Grundlage für zuverlässige Systeme bildet und damit im letzten Kapitel als ideale Basis für massiv verteilte und zugleich zuverlässige Systeme dienen kann.

3. Virtualisierung

Große Last kann durch eine hohe Anzahl von Benutzern entstehen, oder durch die Abwicklung einer zeitlich dringenden Aufgabe, von der unter Umständen mehrere weitere Systeme und letztendlich auch wieder Benutzer abhängen. Solche Abhängigkeiten erfordern eine angemessene Verteilung der Last, da es weder in Frage kommt, Benutzer lange Zeit auf relativ triviale Ergebnisse (wie z.B. das Anzeigen einer angeforderten Webseite von einem Internet-Buchhandel) warten zu lassen, noch darf die größtenteils automatisierte Abwicklung verteilter Geschäftsprozesse (etwa finanzielle Transaktionen) unnötig in die Länge gezogen werden.

Einige Möglichkeiten zur Verteilung verschiedener Arten von Last wurden bereits im letzten Kapitel gezeigt. Durch sie lässt sich die Reaktionszeit von Systemen unter hoher Last entscheidend verbessern. Die Anwendungsmöglichkeiten solcher Ansätze werden anhand der in diesem Kapitel vorgestellten Konzepte später noch einmal aufgegriffen und vertieft.

Zunächst ist jedoch ein anderer Aspekt relevant, der eine zweite wichtige Anforderung an Systeme zur Bewältigung großer Lasten darstellt. Gerade weil diese Art von Systemen einen hohen verwaltungstechnischen, gesellschaftlichen oder finanziellen Stellenwert haben und entsprechend viel von ihrem Funktionieren abhängt – sonst würde man nicht die Zeit und das Geld investieren, diese Systeme zu bauen – ist die ständige Verfügbarkeit der vom System bereitgestellten Dienste eine zwingend notwendige Voraussetzung. Die zentralen Ansprüche sind hierbei also Stabilität und Hochverfügbarkeit und somit ein nicht nur vom Ansatz her solides

Gesamtkonzept, bei dem ausschließlich Hard- und Software zum Einsatz kommt, die besonderen Qualitätsansprüchen genügt, sondern darüber hinaus auch die Fähigkeit, auf auftretende Fehler schnell und flexibel zu reagieren um dadurch solange wie möglich den entsprechenden Dienst weiter zur Verfügung zu stellen.

Bei einigen Systemen dieser Art ist außerdem eine weitere Erkenntnis wichtig, die sich unter entsprechend gegebenen Voraussetzungen besonders günstig ausnutzen lässt: Große Last besteht häufig nicht permanent, sondern bildet sich in der Regel zu besonderen Zeiten, die oftmals sogar vorhersehbar sind. Im Falle eines fiktiven nationalen Internet-Auktionshauses bedeutet das beispielsweise, dass große Last durch die Nutzer vor allem mittags (in der Mittagspause) und abends (nach der Arbeit) entsteht, denn nachts schlafen die meisten potentiellen Nutzer und während der Arbeitszeit haben sie andere Dinge zu erledigen.

Ist das Serversystem des Auktionshauses statisch darauf ausgelegt, die zu den Stoßzeiten entstehende Last problemlos zu bewältigen, dann sind die eingesetzten Server zu den restlichen Tageszeiten nur zu einem geringen Bruchteil beansprucht. Die offensichtlichen Nachteile sind hier zum einen die hohen Anschaffungskosten für ein System, das einen Großteil des Tages kaum ausgelastet ist sowie die laufenden Kosten für Wartung, Strom etc. Anzustreben ist also eine Lösung, die es erlaubt, Ressourcen dynamisch nach Bedarf bereitzustellen und sie in der übrigen Zeit freizugeben und damit für andere Zwecke nutzbar zu machen.

openMosix bildet bereits einen Ansatz in diese Richtung, indem beispielsweise CPU-intensive Prozesse automatisch auf Hosts mit einer unausgelasteten CPU verschoben werden. So lassen sich verschiedene Arten von Last (repräsentiert durch die verschiedenen Prozesse) dynamisch in einem Serverpool verteilen und damit die zur Verfügung stehenden Ressourcen deutlich besser ausnutzen. Zu einem zuverlässigen Server gehört aber nicht nur ein einzelner Prozess, sondern ein ganzes Environment. Dazu können beispielsweise ganz besondere Versionen installierter Zusatzsoftware gehören oder eine festgelegte Verzeichnisstruktur auf der Festplatte. Sogar ein bestimmtes Betriebssystem mit einem besonderen Kernel oder einer Konfiguration, die sich für andere Servertypen nicht eignet, ist hierbei in Betracht zu ziehen.

Solche Anforderungen lassen sich durch die reine Verschiebung von Prozessen wie bei openMosix nicht mehr erfüllen. Stattdessen ist eine Verschiebung des gesamten „abstrakten Servers“ mitsamt seinem Environment erforderlich. Um das zu erreichen, muss der Server mit seinem Environment vollständig von einem konkreten Host und dessen Hardware unabhängig gemacht werden, denn üblicherweise sind nicht alle Hosts, die in einem Netzwerk prinzipiell zur Übernahme von Last geeignet wären, absolut identisch.

Zur Herbeiführung so einer sauberen Trennung von Serverfunktionalität und konkretem Host eignet sich das Mittel der Virtualisierung. Virtualisiert wird in diesem Fall die Hardware des Hosts und damit letztendlich der konkrete Host selbst. Diese Trennung ermöglicht es dann, eine bestimmte Serverfunktionalität von einem beliebigen Host übernehmen zu lassen, unabhängig davon, welches Betriebssystem oder sonstigen Gegebenheiten momentan auf ihm zur Verfügung stehen, denn das

Betriebssystem des Servers und alles weitere lässt sich dann in vollem Umfang auf diesem Host bereitstellen.

Mit der Virtualisierung sind allerdings auch Einschränkungen verbunden. Um also zu verstehen, wie man dieses Mittel geschickt zur optimalen Verteilung von Last nutzen kann, ist es daher erforderlich, zunächst einmal zu verstehen, wie die Virtualisierung funktioniert. Diesem Thema widmet sich das folgende Kapitel, an dessen Anschluss dann die Anwendungsmöglichkeiten solcher Techniken behandelt werden.

3.1. Virtualisierungstechniken

Prinzipiell bietet die Hardware eines Standard-Servers bislang keine besondere Unterstützung an, um eine Virtualisierung zu ermöglichen. Aufgrund der zunehmenden Verbreitung dieser Methode sind allerdings die beiden größten Hersteller für x86-kompatible Prozessoren, *Intel* und *Advanced Micro Devices* (AMD), bereits im fortgeschrittenen Stadium der Entwicklung entsprechender Technologien (namentlich *Silvervale*[9] und *Pacifica*[10]) und wollen noch im Jahr 2006 Produktionsreife erreichen. In Ermangelung entsprechender Technologien wurde die Virtualisierung daher bislang üblicherweise vollkommen durch Software realisiert.

Da das Ziel darin besteht, das gesamte Betriebssystem, welches normalerweise direkte Hardwarezugriffe ausführt, von der Hardware zu lösen, muss also in jedem Fall eine neue Hardware-unabhängige Abstraktionsschicht zwischen Hardware und Betriebssystem eingeführt werden, die Zugriffe vom Betriebssystem zunächst entgegen nimmt und dann auf einer konkreten Hardware umsetzen kann. Dazu gibt es zwei Möglichkeiten: Entweder man verändert das Betriebssystem dahin gehend, dass es der Virtualisierung entgegen kommt und direkt die virtuelle Hardware anspricht, oder man schafft eine vollständig virtuelle Umgebung, so dass ein in dieser Umgebung gestartetes Betriebssystem gar nicht mehr feststellen kann, dass es nicht direkt mit der Hardware kommuniziert. Das Betriebssystem sieht dann nur noch die virtuelle Hardware, wobei Zugriffe auf diese bei Bedarf abgefangen und entsprechend auf die konkrete Hardware übertragen werden.

Da letzterer Ansatz eine deutlich komplexere Abstraktion darstellt, ist leicht nachvollziehbar, dass der erste Ansatz, bei dem das Betriebssystem zugunsten der Abstraktion angepasst wird, einen signifikanten Performancevorteil mit sich bringt.

Das Open-Source-Projekt *Xen virtual machine monitor*[11] (Xen oder auch Xen vmm), das von der Universität in Cambridge betreut wird, verfolgt daher einen solchen Ansatz, der auch als Paravirtualisierung bezeichnet wird. [BarP03] Ein auf die virtuelle Hardware zugreifendes Betriebssystem mitsamt den installierten Programmen und dem restlichen Environment (also das gesamte System ohne konkreten Hardwarebezug) wird dabei „virtual machine“ genannt, Xen selbst ist hingegen der „virtual machine monitor“.

---

[9] http://www.intel.com/business/bss/products/server/virutalization.htm, Stand Januar 2006
[10] http://enterprise.amd.com/Enterprise/serverVirtualization.aspx, Stand Januar 2006
[11] http://www.cl.cam.ac.uk/Research/SRG/netos/xen, Stand Januar 2006

Um die angesprochene Zwischenschicht zur Hardwareabstraktion zu realisieren, wird auf einem Host zunächst statt des eigentlichen Betriebssystems der Xen-Kernel gestartet, der diese Aufgabe übernimmt. Xen selbst ist aber noch nicht als vollständiges Betriebssystem zu verstehen und bietet daher keine Möglichkeit zur direkten Interaktion. Um eine Steuerung zu ermöglichen, wird als nächstes eine besonders privilegierte virtuelle Maschine gestartet, von der aus dem Benutzer durch Zugriff auf ein spezielles Interface die Möglichkeit geboten wird, Xen zu kontrollieren und so beispielsweise den Start beliebiger weiterer virtueller Maschinen zu veranlassen, denen diese Kontrollmöglichkeit dann aus Sicherheitsgründen nicht mehr gewährt wird. Wie sich der Sachverhalt, dass auf einem einzelnen physikalischen Host mehrere virtuelle Maschinen laufen können, nutzen lässt, wird im nächsten Kapitel genauer behandelt.

Zunächst einmal ist hier die Frage relevant, für welche Systeme bzw. Betriebssysteme sich Xen eignet, denn Einschränkungen in diesem Bereich schränken unweigerlich auch die Nutzbarkeit einer solchen Lösung ein. Prinzipiell kann jedes gängige Betriebssystem für x86-kompatible Prozessoren so angepasst werden, dass es Xen unterstützt. Das rechtliche Problem dabei ist allerdings die hierfür notwendige Änderung des Betriebssystems, denn diese erfordert bei proprietären Betriebssystemen zum einen den Quellcode und zum anderen das Einverständnis des Herstellers. Die Anpassung von Linux hingegen ist bereits geschehen. Linux ist damit das bislang einzige Xen vollständig unterstützende Betriebssystem. Eine Windowsversion ist in Arbeit, steht allerdings momentan nicht der Öffentlichkeit zur Verfügung. Ein portiertes Betriebssystem kann dann letztendlich ohne weitere Änderungen privilegiert mit den entsprechenden Tools zur Steuerung von Xen laufen oder als rein virtuelle Maschine.

Eine schematische Darstellung der Funktionsweise von Xen bietet die folgende Abbildung.

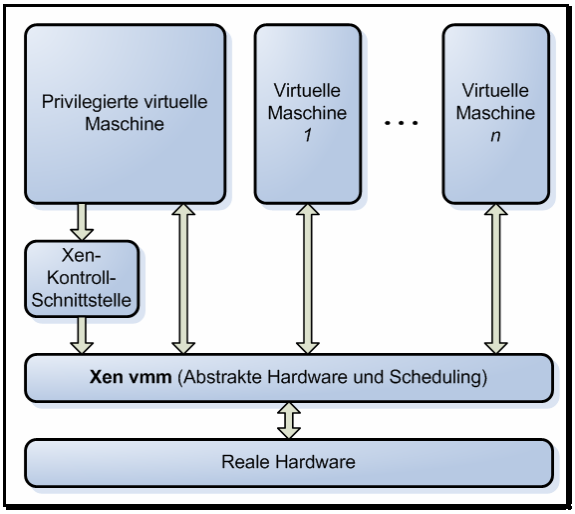

Abb. 4. Virtualisierung mit Xen

Nach dem Start stehen, je nach Konfiguration, jedem virtuellen Host eine eigene virtuelle Netzwerkkarte, ein eigener Speicherbereich, eine eigene virtuelle Festplatte usw. zur Verfügung, er ist also von den anderen virtuellen Hosts, die auf demselben realen Host laufen, vollkommen isoliert und von eventuellen Abstürzen oder anderen Fehlern damit vollkommen unbeeinflusst.

Zu bedenken ist dabei aber, dass trotz der beispielsweise eigenen virtuellen Netzwerkkarte, bei real nur einer einzelnen existierenden Karte die Bandbreite unter allen virtuellen Maschinen aufgeteilt werden muss. Dasselbe gilt auch für Festplattenzugriffe, CPU-Nutzung, Hauptspeicher und alles weitere. Das in diesen Fällen notwendige Scheduling gehört daher zu den Aufgaben des virtual machine monitors, so dass die zur Verfügung stehenden realen Ressourcen angemessen unter den virtuellen Hosts aufgeteilt werden.

Zur Migration einer virtuellen Maschine auf einen anderen Host genügt es, die Konfigurationsdatei der virtuellen Maschine (diese enthält z.B. die Größe des zugeteilten Hauptspeichers) und den Inhalt der virtuellen Festplatte auf einen neuen Host zu verschieben und dort zu starten. Bei einem gemeinsamen Netzwerk-Dateisystem entfällt sogar dieser Vorgang. Überdies besteht zudem die Möglichkeit, eine virtuelle Maschine ohne vorheriges Herunterfahren auf einen neuen Host zu migrieren. Dazu wird der akutelle Zustand kurz eingefroren und dann auf dem neuen Host wieder geladen.

Trotz der eleganten und zugleich gegenüber anderen Lösungswegen überaus performanten Art, wie Xen die Virtualisierung umsetzt, sind jedoch die dazu notwendige Anpassung des Betriebssystems an den virtual machine monitor und die damit bislang herrschende Einschränkung des Betriebssystems für viele Einsatzzwecke inakzeptabel.

Daher verfolgt die Firma *VMware*[12] mit dem Produkt *ESX-Server* den zweiten möglichen Ansatz zur Virtualisierung. Durch eine vollständigere Hardwareabstraktion als bei Xen sind keine Änderungen an den Betriebssystemen der virtuellen Maschinen mehr nötig, eine spezielle Unterstützung zur Beschleunigung des Zugriffs auf die virtuelle Hardware wird, sofern für das jeweilige Betriebssystem verfügbar, ausschließlich über Treiber realisiert. Dabei legt sich der Kernel von ESX, ähnlich wie Xen, als eigentliches Betriebssystem zwischen die virtuellen Maschinen und die reale Hardware. [VMw06]

Da dieses Produkt speziell für Zwecke wie dynamische Migration zur Lastverteilung ausgelegt ist, bietet VMware außerdem das *VirtualCenter* an, mit dem sich solche Vorgänge komfortabel und ohne merklichen Ausfall eines Servers steuern lassen. Überdies lassen sich hiermit auch virtuelle Maschinen beliebig klonen und ihre Auslastung überwachen. Je nach Bedarf können also spontan mehrere virtuelle Instanzen eines Servers auf weiteren Hosts gestartet werden.

Die nächste Stufe der Virtualisierung wird beispielsweise von dem Produkt *GSX-Server* der gleichen Firma realisiert. Hierbei ist das Hardware-Abstraktions-Layer selbst nur noch ein Programm für ein bereits installiertes Betriebssystem, innerhalb

[12] http://www.vmware.com, Stand Januar 2006

dessen dann die virtuellen Maschinen geladen werden können. Im Gegensatz zu den anderen Lösungen, die eine Installation des virtual machine monitors als Basisbetriebssystem erfordern, lässt sich hier auf einem bereits eingerichteten Server beliebig die Fähigkeit zum Starten der virtuellen Server in Form eines Programms nachträglich einrichten. Da das Produkt selbst aber hierbei nicht mehr die volle Kontrolle über das System hat, verliert man damit zugleich die Fähigkeit einer Servermigration ohne kurzzeitigen Ausfall.

Alle diese Lösungen haben jedoch noch eine gemeinsame Einschränkung: Sie abstrahieren nicht die CPU-Architektur, da sich eine solche Funktionalität je nach Unterschied zur real existierenden CPU-Architektur äußerst negativ auf die Performance auswirkt. Dennoch gibt es Produkte wie *Microsoft VirtualPC*[13], die auch diese Funktionalität bieten. [Mic06] Sie stellen die höchste Stufe der Virtualisierung dar.

Dass sich mit jeder höheren Stufe der Abstraktion zwangsläufig die Performance einer virtuellen Maschine verringert, liegt in der Natur der Sache. Wenn möglich bietet es sich daher an, für einen gegebenen Zweck immer die geringste mögliche Stufe der Abstraktion zu wählen. Dass die Performance allerdings unter Umständen nicht immer das wichtigste Kriterium ist, wird das folgende Kapitel zeigen.

3.2. Anwendungen der Virtualisierung

Virtualisierung ermöglicht die Trennung der eigentlichen Serverfunktionalität von einer konkreten Hardware. Wenn man diesen Gedanken konsequent weiter verfolgt, werden gesamte Computer damit zu einer dynamisch nutzbaren Ressource, die nach belieben reserviert und wieder freigegeben werden kann. Der besondere Vorteil, dass ein realer Host zugleich von mehreren virtuellen Hosts genutzt werden kann, eröffnet dabei besonders interessante Möglichkeiten.

Als Beispiel sei hier noch einmal ein fiktives, national tätiges Internet-Auktionshaus mit entsprechend notwendiger Anzahl an Webservern aufgeführt. Während der normalen Arbeitszeiten benötigen die eigenen Mitarbeiter des Unternehmens selbst diverse Arten von Servern, um zu arbeiten. Das können Terminalserver zur gemeinsamen Nutzung von Software sein, Server zur Erstellung und Auswertung von Statistiken oder anderen Kalkulationen und viele andere Arten von Servern. Außerhalb der normalen Arbeitszeiten sind hingegen solche Server dann weitestgehend ungenutzt - dafür haben jetzt aber die potentiellen Nutzer des Auktionshauses Zeit, so dass der Betrieb auf den Webservern des Unternehmens deutlich ansteigt.

Durch die Virtualisierung ist es möglich, dieselben Computer für solche unterschiedlichen Aufgaben zu nutzen, ohne dass eine gegenseitige Behinderung zu befürchten ist. Hierzu könnten auf einem realen Server je ein virtueller Webserver und ein interner Unternehmensdienst-Server bereitgestellt werden. Häufig bietet es sich zu einer klaren Trennung jedoch noch an, tagsüber alle Webserver beispielsweise auf einem Mehrprozessor-System parallel laufen zu lassen. Dieses System wird mit

[13] http://www.microsoft.com/germany/mac/virtualpc, Stand Januar 2006

der geringen Last, die während der normalen Arbeitszeiten entsteht, problemlos fertig. Die anderen realen Server sind damit frei für die intensive Nutzung durch die Mitarbeiter. Nach Ende der Arbeitszeit wird ohne Ausfall von Diensten migriert. Dabei werden die jetzt kaum noch genutzten internen Unternehmensserver auf das Mehrprozessorsystem umgezogen, wodurch die volle Leistung der übrigen realen Server uneingeschränkt für die Kunden zur Verfügung steht. Die folgende Abbildung verdeutlicht diesen Sachverhalt noch einmal schematisch.

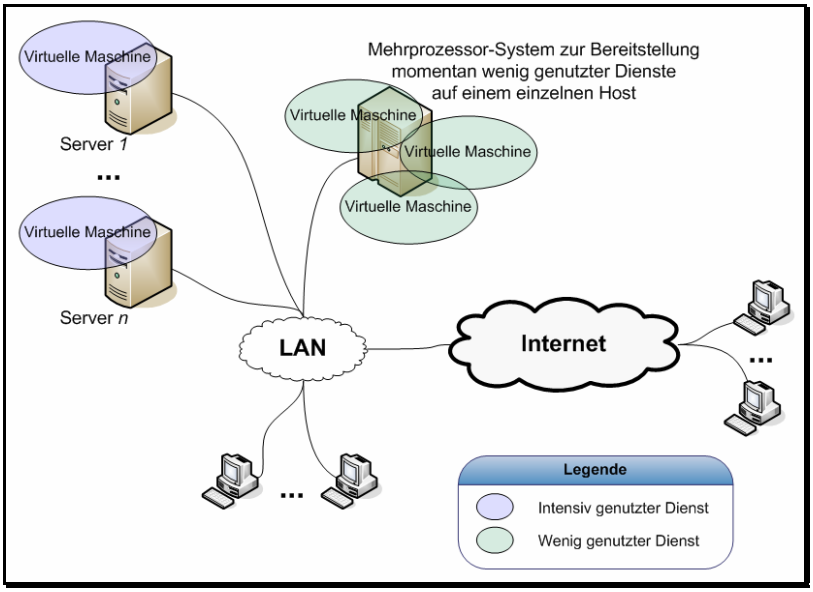

Abb. 5. Anwendungsbeispiel für virtuelle Maschinen

Wichtig in einem solchen Szenario sind zwei Punkte. Zum einen fällt zu keiner Zeit ein Dienst aus. Somit werden weder die Mitarbeiter, noch die Kunden etwas von dieser kostengünstigen Lösung merken. Zum anderen ändert sich trotz Migration auf einen einzelnen realen Host die Anzahl der virtuellen Server nie. Das ermöglicht den unkomplizierten Einsatz von Lastverteilungsverfahren wie Gateways, wobei darüber hinaus sogar noch Hochverfügbarkeit und Ausfallsicherheit erhalten bleiben, denn bei Absturz eines virtuellen Hosts laufen die anderen virtuellen Hosts ungehindert weiter. Außerdem bleibt immer die Möglichkeit, selbst bei massivem Ausfall mehrerer realer Server (z.B. durch einen Ausfall der Klimaanlage im Serverraum), mit Managementlösungen wie VirtualCenter ohne großen Aufwand sämtliche Dienste auf prinzipiell beliebigen Hosts im Netz wieder zur Verfügung zu stellen. Ein gemeinsam genutztes Netzwerk-Dateisystem, wie es ohnehin üblicherweise in Organisationen bereits existiert, ist hier also eine optimale Ergänzung zur Vereinfachung der Migration. Durch entsprechende Schnittstellen der Virtualisierungssoftware können dann solche Schritte sogar vollkommen automatisiert erfolgen.

Noch flexibler macht einen die Virtualisierung bei der Bereitstellung eines Dienstes, der sich dynamisch mit einer variablen Anzahl von Hosts skalieren lässt. Hier bietet sich sogar die Möglichkeit, über Nacht ungenutzte Workstations mit einzubeziehen. Dass in einem derart flexiblen Ansatz ein sehr großes Potential zur dynamischen Bereitstellung und Nutzung von bislang ungenutzter Rechenleistung steht, wird im nächsten Kapitel noch einmal eingehender Behandelt. Durch solche Lösungen ist letztendlich nicht mehr die Performance bei der Umsetzung der Virtualisierung ausschlaggebend, stattdessen tritt der Bedarf für Dienste in den Vordergrund, die sich durch automatische Skalierung auf solche Gegebenheiten wie eine variable Anzahl von Hosts anpassen.

3.3. Zusammenfassung

Virtualisierung ist ein hohen Zuverlässigkeitsansprüchen genügendes Werkzeug zur effizienten Verteilung von Last in einem lokalen Netzwerk. Der entscheidende Vorteil ist dabei die Einfachheit, mit der ohne großen Aufwand (und bei Bedarf auch vollkommen automatisiert) virtuelle Hosts auf realen Hosts verschoben werden können. Damit kann zusätzlich auch die Ausfallsicherheit gewährleistet werden, indem im Fehlerfall ein Klon des virtuellen Servers auf einem beliebigen anderen Host, der die Virtualisierungsplattform unterstützt, gestartet wird. Das virtuelle System ist auf dem neuen Host sofort Einsatzbereit und erfüllt durch den Erhalt des kompletten Environments unverzüglich und uneingeschränkt seine volle Funktionalität, da spezielle Zusatzsoftware oder Betriebssystemkonfigurationen bereits Teil der virtuellen Maschine sind.

Die hier gezeigten Beispiele bezogen sich jedoch bislang vor allem auf Lastbalancierungen in lokalen Netzwerken. Um noch erheblich größere Rechenleistung durch massive Verteilung von Diensten zu nutzen, die aber nicht die Nachteile von Peer-to-Peer-Netzwerken mit sich bringt, existieren ebenfalls entsprechende Ansätze. Diese werden im folgenden Kapitel vorgestellt.

4. Grids

Virtualisierung und lokale Lastverteilung bilden eine sehr gut funktionierende Kombination, um mit regelmäßig auftretender Last umzugehen. Im bisher diskutierten Rahmen unterliegt diese Lösung jedoch noch einigen wichtigen Einschränkungen. Trotz der durch Virtualisierung ermöglichten Konsolidierung von verschiedenen virtuellen Serverdiensten auf einem einzelnen realen Server, die ihre Vorteile speziell bei weniger genutzten Diensten entfaltet, wird zur Behandlung größerer Last noch immer eine entsprechend hohe Anzahl physikalischer Hosts benötigt, die diese bewältigen kann.

Außerdem kann Last in vielen Formen entstehen und tritt nicht immer so „überschaubar“ und regelmäßig auf, dass ein lokaler Cluster mit einem vorgeschalteten oder integrierten Lastverteiler die geeignete Lösung darstellt. Häufig

gilt es daher, zwei weitere Arten der Last zu behandeln: einmalige oder selten wiederkehrende, zeitlich begrenzte Last und andauernde, extreme Last.

Das nächste Kapitel widmet sich daher zunächst diesen beiden Arten von Last und zeigt spezielle Ansätze und Lösungswege zu ihrer Behandlung auf. Diese speziellen Lösungswege erfüllen zwar nicht alle Kriterien für ein allgemein einsetzbares System, verdeutlichen jedoch bereits einige wichtige Anforderungen an ein solches System und bieten darüber hinaus zum Teil auch Möglichkeiten zur Erfüllung dieser Anforderungen.

4.1. Extremere Arten von Last und spezielle Behandlungsweisen

Die erste der beiden oben genannten Arten von Last, die sporadisch auftretende Last, ist vor allem relevant für kleinere Unternehmen oder auch wissenschaftliche Einrichtungen. Zwar werden hier von Zeit zu Zeit dringend Möglichkeiten zur verteilten Berechnung benötigt, etwa um detailgetreue Simulationen durchzuführen oder andere komplexe Probleme zu lösen, doch sind die einzelnen Ergebnisse häufig nicht wertvoll genug, um die Anschaffung und die laufenden Kosten für entsprechende Computing-Cluster zu decken. Üblicherweise schließen sich dann in solchen Fällen mehrere betroffene Organisationen zusammen, um durch die gemeinsame Nutzung von Ressourcen diesen Sachverhalt zu kompensieren.

Die ideale Lösung in einem solchen Fall wäre hingegen ein Dienst, der es erlaubt, überschüssige Ressourcen in einer Organisation nach Bedarf zur Nutzung durch andere zur Verfügung zu stellen. Damit könnten Organisationen mit entsprechendem Bedarf diese Ressourcen dann reservieren, nutzen und entsprechend der Dauer der Beanspruchung bezahlen. Reservierbare Ressourcen müssten sich hierbei sogar nicht nur auf reine Rechenleistung beschränken, sondern könnten auch aus Speicherplatz oder dem Zugriff auf andere Dienste bestehen.

Tatsächlich existieren in fast jeder Organisation solche ungenutzten Ressourcen, die hierfür in Frage kämen. Beispielsweise bei der Anschaffung neuer Fileserver wird vorausschauend davon ausgehend gekauft, dass der Speicherplatzbedarf der Organisation in den nächsten Monaten und Jahren stetig zunehmen wird. Daher wird der verfügbare Speicherplatz in der Regel mindestens einige Monate lang nur zu einem geringen Prozentsatz beansprucht.

Dass aber auch außerhalb von Organisationen eine bedeutende Anzahl von oftmals ungenutzten Ressourcen vorhanden ist, wurde bereits bei der Entwicklung der eingangs erwähnten Peer-to-Peer-Netzwerke erkannt und genutzt. Das theoretisch vorhandene Potenzial, das sich aus der Masse der auf globaler Ebene betrachtet existierenden, unausgelasteten Personal Computer ergibt, ist auch den hauptsächlich wissenschaftlichen Einrichtungen bewusst geworden, die mit der zweiten Art der Last, der extremen und andauernden Last, fertig werden müssen.

Eines der bekanntesten Projekte, bei dem es gelungen ist, diese Ressourcen nutzbar zu machen, ist *SETI@home*[14] (SETI steht für „Search for ExtraTerrestrial Intelligence“). [UnK06] Die Last besteht bei diesem Projekt darin, die täglich vom

[14] http://setiathome.ssl.berkeley.edu, Stand Januar 2006

weltweit größten Radioteleskop in Arecibo gelieferte Datenmenge von etwa 70GB zu analysieren, um so nach enthaltenen Signalen, die von einer außerirdischen Intelligenz stammen könnten, zu suchen. Dazu werden die Teleskopdaten in 200.000 etwa 350KB große Päckchen geteilt, die unabhängig von den restlichen Datenpäckchen in ein paar Stunden von einem PC analysiert werden können. Zur Teilnahme lädt sich ein registrierter Benutzer ein kleines Programm herunter, das als Bildschirmschoner läuft und sich automatisch über das Internet immer wieder neue Datenpäckchen vom SETI@home-Server holt, sie analysiert und die Ergebnisse anschließend an den Server übermittelt. Die Architektur dieses Systems wird in der folgenden Abbildung veranschaulicht.

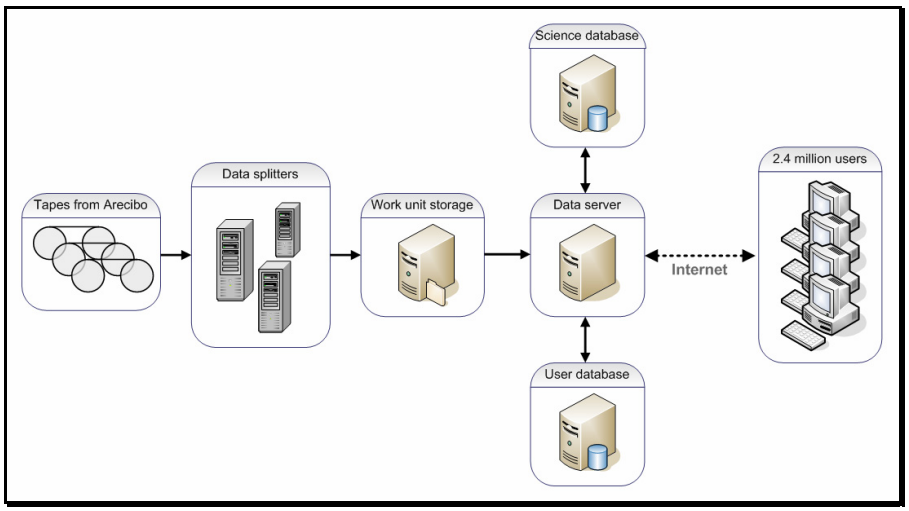

Abb. 6. Die Architektur von SETI@home
(nach: http://www.computer.org/cise/articles/seti.htm, August 2004)

Auffällig bei diesem System ist, dass trotz der beträchtlichen Zahl von Benutzern, im Prinzip ein simples, hierarchisches, Client-Server-artiges Modell gewählt wurde, bei dem ein über das Internet kontaktierter Data-Server den Computing-Nodes, also den PCs der Benutzer, als kontrollierende Instanz Jobs in Form von Datenpaketen zuteilt.

Allgemeiner formuliert liegt diesem System somit eine Architektur zugrunde, bei der über mehrere administrative Domänen verteilte Computing-Nodes dynamisch als Ressource zur Verfügung gestellt werden können. Ein zentraler Dienst übernimmt dabei mehrere Aufgaben, die zum Teil auch verteilt vorstellbar wären:

- *Authentifizierung* durch Kopplung an eine von außen nicht direkt zugängliche Benutzerdatenbank, wodurch der Sicherheitsaspekt des Gesamtsystems und auch die Anonymität gegenüber Dritten abgedeckt wird.

- Proxy zu einem *gemeinsam genutzten Speicher* für die Nodes, aus dem sie ihre Quelldaten beziehen und in dem sie Ergebnisdaten persistent hinterlegen können.
- *Scheduling* durch die Erfassung statistischer Daten. Hierbei ergibt sich die Möglichkeit, regelmäßig teilnehmende Benutzer von unzuverlässigeren Benutzern zu unterscheiden und auf dieser Basis z.B. wichtigere Jobs bevorzugt an Nodes von zuverlässigeren Benutzern auszugeben.
- *Abrechnung* durch analysierte Pakete. Wäre SETI@home ein kommerzielles System, könnte hier problemlos eine Abrechnung aufgrund der Anzahl der von jedem Benutzeraccount analysierten Pakete erfolgen.

Insgesamt bietet SETI@home damit bereits eine sehr gelungene Grundlage zur massiv verteilten Analyse. Nicht zu unterschätzen ist dabei auch die unkomplizierte Möglichkeit für die Benutzer, ihren PC gewissermaßen nach belieben freizugeben oder die Freigabe zu beenden. Dennoch funktioniert dieses System nur aufgrund des Kompromisses, dass zwar durch die Einbeziehung der privaten Nutzer sehr viele Pakete analysiert werden, aber nicht alle – Eine Tatsache, die man bei SETI@home einfach hinnehmen muss, da es aufgrund der aufwendigen Analyse keine Möglichkeit gibt, sämtliche Daten zu untersuchen; jedoch lässt sich dieser Umstand kaum auf allgemeinere Anwendungsszenarien übertragen. Außerdem erfordert das Projekt die Akzeptanz und Begeisterung der Privatpersonen, die ihre PCs für das Projekt zur Verfügung stellen. Nur wenige Projekte werden in der Lage sein, eine derartige Community für sich zu gewinnen.

Da SETI@home auf diesen sehr speziellen Voraussetzungen basiert, lässt sich das Verfahren kaum für beliebige Projekte einsetzen. Dennoch bildet es eine gut funktionierende Architektur, die bereits viele Aspekte der dynamischen Freigabe und Nutzung von Ressourcen abdeckt, und die somit im nächsten Kapitel als Grundlage genutzt werden kann für ein System, das auch allgemeineren Ansprüchen gerecht wird.

4.2. Systeme zur dynamischen Freigabe und Nutzung von Ressourcen

Aus den bislang betrachteten Arten von Last ergeben sich viele Anforderungen an ein System, das zur generellen Bewältigung von Last dienen soll. Wichtig ist bei dem Versuch, ein solches System zu konstruieren, daher die Einsicht, dass es nicht „das eine System“ gibt, das alle Anforderungen optimal erfüllt. Stattdessen ist es somit von großer Bedeutung, beim Design des Systems Prioritäten zu setzen und gewichtige Aspekte entsprechend herauszuarbeiten.

Ein vollkommen allgemein gehaltenes System hätte hingegen zwangsläufig zur Erfüllung sämtlicher Anforderungen – sofern man nicht sogar widersprüchliche Anforderungen berücksichtigt – eine so hohe Komplexität, dass es sich kaum zur Benutzung anbieten würde. Durch Spezialisierung jedoch können von einem solchen System verschiedene Varianten mit entsprechenden Ausprägungen abgeleitet werden, die ihren jeweiligen Benutzergruppen in Punkten wie einfacher Bedienung, Sicherheit oder Performance entgegen kommen und damit wesentlich nützlicher und erfolgreicher sind als „das eine System“.

In diesem Sinne stellt dieses Kapitel ein Konzept für eine Architektur vor, bei der man sich bemüht hat, möglichst viele allgemeine Anforderungen zu erfüllen – aber nicht alle. Demnach existieren bereits einige konkrete Projekte, die von diesem allgemeinen Konzept abgeleitet sind und sich entsprechend eigener Prioritäten spezieller entwickelt haben. Das Konzept selbst ist allerdings noch Gegenstand vieler wissenschaftlicher und kommerzieller Forschungsprojekte und befindet sich momentan in einer Phase der ständigen Weiterentwicklung.

Der Name für die Architektur, die sich hinter diesem Konzept verbirgt, lautet „Grid“. Der Begriff entwickelte sich in den frühen 90er-Jahren mit dem Hintergedanken, Computer-Ressourcen so einfach zugänglich zu machen wie Strom aus dem Stromnetz (engl. „power grid“). [Fos99]

Wesentlich bei der Umsetzung dieser Idee sind zunächst zwei Gruppen von Grid-Benutzern, die miteinander verbunden werden müssen. Das sind zum einen die Anbieter des Netzes, die ihre Ressourcen zur Nutzung freigeben, und zum anderen die Verbraucher, die sich bei diesen Ressourcen bedienen können. Dieses Konzept kommt zwar auch zur Verbindung mehrere Standorte oder Abteilungen eines Unternehmens in Frage, macht aber hauptsächlich erst auf überregionaler oder sogar internationaler Ebene Sinn, damit eine angemessene Menge und Bandbreite von nutzbaren Ressourcen im Netz vorhanden sind. Als Mittel zur Vernetzung bietet sich daher derzeit hauptsächlich das Internet an.

Um sich von dem Begriff der einzelnen Benutzer zu lösen, der sich für ein System dieser Größenordnung oft zu nah an einzelnen Personen orientiert, wurde außerdem der Begriff der „virtuellen Organisationen“ eingeführt. Somit geht es bei einem Grid nicht mehr nur um vereinzelte Teilnehmer, die ihren PC als Ressource zur Verfügung stellen, sondern viel mehr um tatsächliche Organisationen, die sich dynamisch nach Bedarf (z.B. bei gemeinsamen Projekten) mit anderen Organisationen zusammenschließen, um so zusammen die Ressourcen des Grids zu nutzen oder auch ganze Cluster als Grid-Ressourcen zur Verfügung zu stellen. Die folgende Abbildung veranschaulicht dieses Konzept noch einmal schematisch.

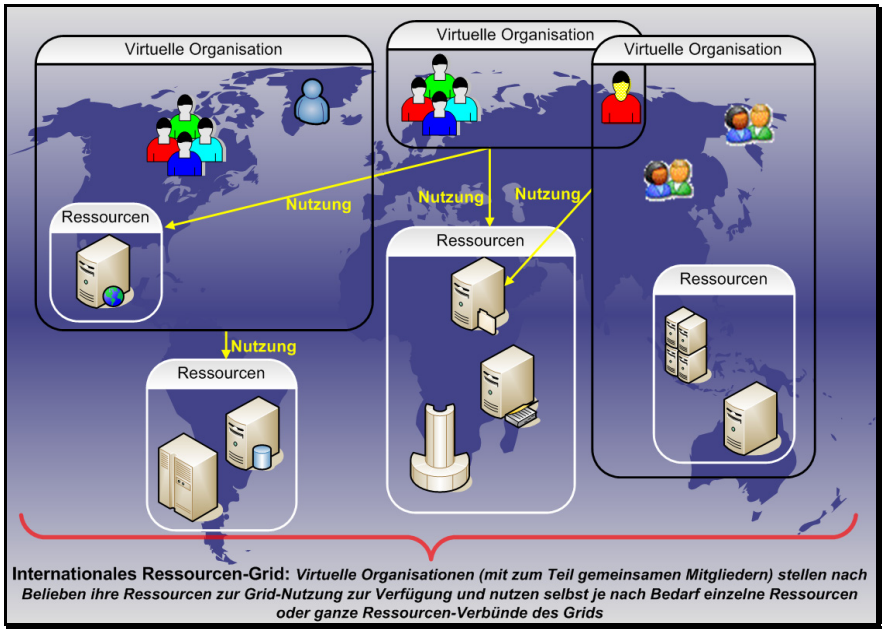

Abb. 7. Nutzung eines Grids durch virtuelle Organisationen

Ein Grid ist somit, vereinfacht gesagt, ein Zusammenschluss von diversen Ressourcen verschiedener Organisationen, die durch beliebige Anwender nach Bedarf genutzt werden können. Durch diesen Zusammenschluss entstehen im Rahmen von einzelnen Organisationen bislang nicht vorstellbare Kapazitäten, vergleichbar mit SETI@home. Tatsächlich gilt SETI@home sogar bereits als eine sehr einfache Grid-Variante.

Bei Grids kann man aufgrund der massiv verteilten Struktur Parallelen zu Peer-to-Peer-Netzen sehen, doch liegt der Grid-Software aus Gründen der Sicherheit und Zuverlässigkeit eher eine mehrschichtige, hierarchische Organisation und Kommunikation zugrunde, wodurch beispielsweise die Anonymität der virtuellen Organisationen gewährleistet werden kann. Darüber hinaus existieren Gemeinsamkeiten mit Computing-Clustern. Einer der Hauptunterschiede zu diesen ist dabei, neben der mehrere administrative Domänen umfassenden Organisation, die Vielfältigkeit der Ressourcen in einem Grid, die eine ganz neue Art des Schedulings erfordert. Hierzu gehören sowohl die Auffindung von geeigneten Ressourcen für spezielle Jobs, als auch eine Berücksichtigung der Möglichkeit, dass unterschiedliche Anbieter von Grid-Ressourcen verschiedene Preise für diese verlangen können, und dem Nutzer damit eine Auswahl von Prioritäten (Kosten/Dauer/…) gewährt werden muss.

Die für ein Grid benötigte, verständlicherweise sehr aufwendige Middleware soll hier aber nicht im Detail erläutert werden (Hinweise auf einige der benötigten

Komponenten fanden sich bereits im vorigen Kapitel). Stattdessen ist viel mehr die Frage relevant, für welche Anwendungszwecke im Sinne der Lastbewältigung sich solche Grids letztendlich eignen.

Aufgrund der hohen verfügbaren Rechenleistung bei parallelen Berechnungen, kristallisiert sich gleich eine der wichtigsten Anwendungsmöglichkeiten heraus. Den Nutzern solcher Systeme wird damit eine Möglichkeit geboten, Kosten und Nutzen viel direkter in Relation zu setzen. Abgesehen davon lässt sich in einem Grid eine Rechenleistung erschließen, die ohne einen derartigen Verbund niemals hätte genutzt werden können. Speziell im Vergleich zu Clustern ergeben sich jedoch auch neue Einschränkungen, die bei der Nutzung eines Grids zu berücksichtigen sind. Hierzu zählt beispielsweise die niedrigere Geschwindigkeit der über das Internet geleiteten Datenübertragungen gegenüber einem LAN. Grids eignen sich daher eher für sehr CPU-intensive, weniger jedoch für I/O-intensive Berechnungen. Ein Grid allerdings auch einfach zur Speicherung von großen Datenmengen zu nutzen, wäre ebenfalls vorstellbar.

Letztendlich hängt bei den Anwendungsmöglichkeiten alles von den Diensten und Sicherheitsrichtlinien ab, die durch die Middleware und die teilnehmenden Anbieter vorgegeben werden. Für eine sehr flexible Nutzungsmöglichkeit ohne eine dafür notwendigerweise komplexe Middleware, könnte man auf die Verwendung von Virtualisierungstechniken zurückgreifen, wie sie im dritten Kapitel beschrieben wurden. Auf diese Weise könnten Nutzer statt nur spezieller Anwendungen ihr gesamtes, speziell konfiguriertes Environment auf Computern im Grid einsetzen, indem das Grid einfach nur die Ausführung von virtuellen Maschinen zulässt. Dieses Verfahren bringt darüber hinaus noch weitere Vorteile für beide Seiten: Durch die Virtualisierung wird der Grid-Job einerseits vollkommen von der physikalischen Hardware isoliert, woraus sich ein erhöhtes Maß an Sicherheit für den Betreiber der Ressource ergibt; auf der anderen Seite ist der Nutzer innerhalb seiner virtuellen Maschine voll privilegiert und kann beispielsweise Netzlaufwerke mounten, was sonst nur einem Administrator vorbehalten wäre. Durch den Einsatz von Virtualisierungsplattformen auf Programmebene (vgl. VMware GSX-Server) könnte ein Computer im Grid die Ausführung von virtuellen Maschinen sogar problemlos als nur einen von mehreren Diensten anbieten. [Fig03]

Eine weitere Anforderung, die häufig an Grids gestellt wird, ist die vorherige Reservierung einer Ressource, gegebenenfalls sogar mehrerer Ressourcen gleichzeitig. Im Falle des bereits mehrfach angesprochenen, fiktiven Internet-Auktionshauses, für das bislang die lokale Lastverteilung optimal zu sein schien, würde sich mit einem derartigen Scheduling und einem direkten Internetzugang hier die Möglichkeit ergeben, die Webserver in das Grid zu verlegen und somit zugleich überregional zu verteilen. Während der Arbeitszeit reserviert das Unternehmen dann eine kleinere Anzahl von Hosts im Grid, nach der Arbeitszeit eine entsprechend größere. Nimmt die Last einmal ein ungewöhnlich hohes Ausmaß an, können dynamisch weitere Hosts im Grid zusätzlich genutzt werden. Schwerwiegendes Gegenargument für eine derartige Nutzung eines Grids ist die geringere Bandbreite und die höhere Latenz bei der zur Synchronisation der Daten notwendigen Kommunikation der Webserver mit einer gemeinsamen Datenbank. Durch die

Einführung von *Quality-of-Service*-Mechanismen[15] lässt sich aber auch dieses Hindernis beseitigen.

4.3. Zusammenfassung

Grids in ihren verschiedenen Ausprägungen bilden ihren Nutzern vormals nicht existierende Möglichkeiten zur dynamischen oder auch extensiven Nutzung diverser Arten von Ressourcen und eröffnen damit nicht nur einen ganz neuen Markt (durch Spezialisierung von Anbietern auf die Bereitstellung solcher Ressourcen), sondern ermöglichen auch die Bearbeitung derart komplexer Probleme, wie man sie vorher kaum hätte lösen können. Im aktuellen Stadium lässt sich jedoch nur erahnen, welches Potenzial in der Verbreitung von Grids liegt, denn noch ist ihre Nutzung eher experimentell, und somit sind sie der breiten Masse noch nicht zugänglich.

Ob eines Tages wirklich beliebige Personen bei Bedarf an größerer Rechenleistung „mal eben“ 50 Computer im Grid oder auch 2 TerraByte an Grid-Speicherplatz reservieren, bleibt daher abzuwarten. Dafür muss zum einen die Akzeptanz und das Vertrauen der potentiellen Nutzer geschaffen werden, ihre eventuell kritischen Jobs von nicht näher bekannten Computern irgendwo in der Welt bearbeiten zu lassen, zum anderen müssen die Zugriffsmöglichkeiten auf Grids letztendlich so einfach sein, dass sowohl eine spontane Bei-Bedarf-Nutzung durch die breite Masse als auch zusätzlich eine (eventuell automatisierte) Nutzung für komplexe Anwendungsszenarien ermöglicht wird.

5. Fazit

Mittel zur Bewältigung von Last müssen anhand des vorliegenden Szenarios jeweils neu gewählt werden. Dabei gilt es nicht nur, die Art der Last zu berücksichtigen, sondern immer auch ihre spezielle Ausprägung. Zudem bietet es sich an, Prioritäten wie benötigte Ausfallsicherheit oder Performance zu setzen, um den Kreis der in Frage kommenden Lösungen weiter einzugrenzen.

Ist am Ende ein konkretes System gefunden, das den Anforderungen gerecht zu werden scheint, muss es sich dennoch erst in der Praxis bewähren. Legt man zum Beispiel großen Wert auf Stabilität, sollte zur Qualitätssicherung zunächst ein geeigneter Stresstest gefunden werden, mit dem das System vor dem Produktivbetrieb überprüft werden kann, um sicherzugehen, dass es den Ansprüchen gerecht wird.

Grids scheinen im Moment zwar als Hoffnungsträger geeignet, um einen erheblichen Teil der lokalen Computer-Ressourcen in absehbarer Zeit abschaffen zu können und sich damit zugleich auch von administrativen Tätigkeiten zu befreien, doch werden sie bislang noch nicht durch die breite Masse genutzt. Das bedeutet, ihr eigentlicher Stresstest steht ihnen erst noch bevor – und solange kann niemand mit Sicherheit sagen, ob wir direkt vor einem neuen Zeitalter der Ressourcennutzung

[15] http://www.cisco.com/warp/public/cc/so/neso/vpn/vpne/qsvpn_wp.pdf, Stand Januar 2006

stehen oder die Verwirklichung dieser Vision vielleicht doch noch in unerreichbarer Ferne liegt.

Quellenverzeichnis

[Höv05] Hövel, J. a. d.: IT-Architektur für E-Commerce-Riesen: Ebay. Computerwoche Ausgabe 18/2005 (2005). http://www.aufdemhoevel.de/ebay.html, Stand Januar 2006

[Ada00] Adar, E. and Huberman, B. A.: Free riding on gnutella. First Monday, 5(10). (2000)

[Hei05] Heise online: Stranfanzeigen-Maschine gegen Tauschbörsen-Benutzer. (2005). http://www.heise.de/newsticker/meldung/63635, Stand Januar 2006

[BarM03] Bar, M. et al: openMosix. Linux-Kongress in Saarbrücken, Paper (2003) http://openmosix.sourceforge.net/linux-kongress_2003_openMosix.pdf, Stand Januar 2006

[BarP03] Barham, P. et al: Xen and the Art of Virtualization. (2003)

[VMw06] VMware ESX/GSX-Server Spezifikationen: http://www.vmware.com/pdf/esx_specs.pdf, http://www.vmware.com/pdf/gsx_specs.pdf, Stand Januar 2006

[Mic06] Microsoft VirtualPC for Mac: http://www.microsoft.com/mac/products/virtualpc/virtualpc.aspx?pid=virtualpc, Stand Januar 2006

[UnK06]: Universität von Kalifornien: SETI@home. http://setiathome.ssl.berkeley.edu, Stand Januar 2006

[Fos02] Foster, I.: What is the Grid? A Three Point Checklist. (2002)

[Fos99] Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, San Francisco (1999)

[Fig03] Figueiro, R. J. et al: A Case for Grid Computing on Virtual Machines. (2003)