DIPLOMARBEIT

Lars Uhlemann

Hochschule für Technik, Wirtschaft und Kultur Leipzig
Fachbereich für Informatik,
Mathematik und Naturwissenschaften

Diplomarbeit

Planung und Aufbau einer auf Linux basierenden Fileserverinfrastruktur am Umweltforschungszentrum Leipzig

vorgelegt von Lars Uhlemann

Leipzig, 26. November 2003

Betreut von: Prof. Dr. Klaus Hänßgen (HTWK-Leipzig)
und Dr. Thomas Wieser (UFZ-Leipzig)

Ich versichere wahrheitsgemäß, die Diplomarbeit selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderung entnommen wurde.

Lars Uhlemann Leipzig, den 26. November 2003

Inhaltsverzeichnis

Einleitung

Umfangreiche Bilddatenbanken, Digitales Audio/Video und wissenschaftliche Messaufzeichnungen sind fester Bestandteil der Informationsverarbeitung geworden. Diese Arbeitsgebiete zeichnen sich durch einen enormen Bedarf an Speicherplatz aus. Laut Storage Magazin (Ausgabe 1/2003) verdoppelt sich die allgemein benötigte Speichermenge jedes Jahr. Der stetig ansteigende Speicherbedarf erfordert neuer Denkweisen in der Administration und Hardwareentwicklung.

Auch am Umweltforschungszentrum Leipzig (UFZ) wird die Anpassung der Speicherkapazität an den Bedarf der Benutzer thematisiert, da die vorhandene Fileserverinfrastruktur nicht mehr in der Lage ist diese Anforderung zu erfüllen. Somit wurde durch die Beschaffung zusätzlicher Hardware neue Speicherkapazitäten geschaffen. Wie wird diese zusätzlich geschaffene Speicherkapazität in die bestehende Fileserverinfrastruktur integriert und wie wird sie den Benutzern zur Verfügung gestellt? Diese beiden Fragen werden im zu erstellenden neuen Betriebskonzept beantwortet.

Die stetig steigenden Kosten der Programmlizenzen und der Produktunterstützung stellen immer häufiger ein Problem dar. Dies wird um so deutlicher in Zeiten des immer präsenten Zwangs zum Sparen. Hinzu kommen neuen Lizenzmodelle, welche zu immer stärker werdenden Abhängigkeiten von einzelnen Software-Herstellern führen. Das Umweltforschungszentrum Leipzig als Mitglied der Helmholtz Gemeinschaft deutscher Forschungszentren wird zu 100 Prozent durch die öffentliche Hand finanziert. Es ist somit zum effektivem und sparsamen Umgang mit dem ihm zur Verfügung gestellten Mitteln verpflichtet. Wie können Kosten vermindert und Abhängigkeiten von Software-Herstellern vermieden werden? Als Antwort auf diese Frage ergibt sich der Einsatz von freier Software als fester Bestandteil des zu entwickelnden neuen Betriebskonzeptes. Die Umstellung der bisher eingesetzten kommerziellen Software auf freie Software, sowie deren spezifische Anpassung an die Gegebenheiten und Anforderungen des Umweltforschungszentrums stellen ein weitere zu lösende Aufgabe dar.

Im ersten Kapitel werden die nötigen Grundlagen zum Verständnis des Themas Fileserver vermittelt. Das zweite Kapitel befasst sich mit den Gründen für die Ablösung des alten Fileserverkonzeptes und geht anschließend ausführlich auf die Erstellung des neuen Betriebskonzeptes

ein. Im folgenden Kapitel wird auf die Anpassung der eingesetzten Software an die spezifischen Gegebenheiten am Umweltforschungszentrum eingegangen. Im vierten Kapitel wird die Überlegenheit der neuen Fileserver-Hardware im Vergleich zur alten Fileserverstruktur mit einem Test bewiesen. Mit einer Zusammenfassung der Ergebnisse und einem Ausblick wird die Arbeit mit dem fünften Kapitel abgeschlossen.

Die beschriebenen Probleme und gestellten Aufgaben werden in dieser Arbeit ausführlich behandelt und entsprechend gelöst.

Konventionen

Alle relevanten Fachbegriffe und Abkürzungen werden bei ihrem ersten Auftreten kurz erläutert. Zusätzlich sind im Glossar die wichtigsten Fachbegriffe aufgeführt. Englische Fachbegriffe werden in die deutsche Sprache übersetzt und wenn nötig erklärt. Alle relevanten Bezeichnungen und Begriffe sind im Index aufgeführt. Existiert zu einer Bezeichnung oder zu einem Begriff ein Glossareintrag, ist der entsprechende Seitenverweis im Index kursiv dargestellt.

Im hinteren Umschlag befindet sich eine CD-Rom mit der elektronischen Version dieser Arbeit. Zum Lesen dieses elektronischen Dokuments wird die Software Adobe Acrobat Reader vorausgesetzt. Diese kann für eine Reihe von Betriebssystemen von http://www.adobe.de bezogen werden.

Kapitel 1

Theoretische Grundlagen

Dieses Kapitel befasst sich mit den Grundlagen aller Themengebiete, welche direkt zum Umfeld eines Fileservers gehören und für dessen Betrieb nötig sind. Es stellt die Grundlage für die nachfolgenden praktisch bezogenen Kapitel dar. In Abschnitt 1.1 wird auf die grundsätzlichen Aufgaben, die ein Fileserver zu bewältigen hat, eingegangen. Im darauf folgenden Abschnitt 1.2 werden die verschiedenen Arten der Lokalisierung von Daten-Speichern erläutert. Anschließend (Abschnitt 1.3) werden verschiedene Dateisysteme, unter Berücksichtigung auf deren Eignung für einen Fileserver, angesprochen. In Abschnitt 1.4 wird ein Überblick auf die verschiedenen Fileserversoftwarelösungen gegeben und deren Vor- bzw. Nachteile unter verschiedenen Gesichtspunkten betrachtet. Anschließend (Abschnitt 1.5) werden Mechanismen und Verfahren zur Benutzer-Verwaltung und zur Zugriffssteuerung angesprochen und diskutiert. Im letzten Abschnitt (1.6) wird auf das Thema RAID und Backup, im Zusammenhang mit dem Oberbegriff Datensicherheit, eingegangen.

1.1 Aufgaben eines Fileservers

> Der Begriff Fileserver bedeutet sinngemäß übersetzt: ein Rechnersystem welches Dateien (Daten) [1] oder Programme bereithält, um diese lokal oder hauptsächlich über ein Netzwerk [2] zu verteilen, anzubieten oder auszuführen.

Daten und Programme werden zentral auf dem Fileserver gehalten und gespeichert. Benutzer, welche über die benötigten Zugriffsrechte verfügen (siehe 1.5), sind in der Lage von ihren Rechnern, welche mit dem Fileserver über ein Rechner-Netzwerk verbunden sind, auf die dort abge-

[1] Dateien dienen als Container für die verschiedensten Arten von Daten z.B.: ausführbare Programme, Texte, Tabellen, Videos, Musik usw.

[2] Unter *Netzwerk* wird hier ein Rechner-Netzwerk verstanden

legten Daten zuzugreifen und mit diesen zu arbeiten. Dadurch ergeben sich vielfältige Möglichkeiten. Zum Beispiel kann der Benutzer dort seine Arbeitsdaten ablegen und anderen Benutzern zur Verfügung stellen. Er kann sie dort sichern (ein Backup erstellen, siehe 1.6) oder sie ausschließlich nur auf dem Fileserver speichern und diese dort direkt bearbeiten! Weitere Möglichkeiten sind das direkte Ausführen von Programmen. Somit müssen sich die Programme nicht mehr lokal auf den Rechnern der Benutzer befinden und belegen dort keinen Speicherplatz. Ein Vorteil ist auch das Ablegen aller persönlichen Daten auf dem Fileserver. Der Benutzer kann von einen beliebigen Rechner aus, welcher mit dem Fileserver über das Netzwerk verbunden ist, seine Daten bearbeiten und ist somit unabhängig von einem einzelnen Rechner.

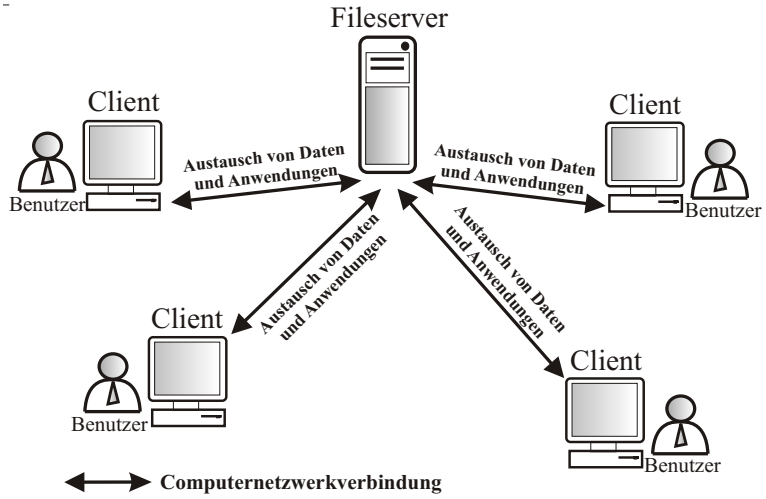

Abbildung 1.1: Beziehung zwischen Fileserver und Clients

In Abbildung 1.1 steht der Fileserver im Zentrum des Rechner-Netzwerkes. Er liefert von den Clients[1], angeforderte Dateien aller Art an diese aus und bietet diesen somit seine Dienste an. Der Begriff *Dienste* wird im weiteren als Oberbegriff für alle Aufgaben, die ein Server bearbeitet, eingeführt. Ein Fileserver bietet somit einen Fileserverdienst an.

Die Voraussetzung für einen Fileserver ist eine Software, welche die oben genannten Aufgaben erfüllen kann. Diese Software wird als *Fileserver-Software* bezeichnet. Hier gibt es die verschiedensten Lösungen im Bezug auf die Portabilität[2] , Performance, Sicherheit und Administrierbarkeit. Alle diese einzelnen Punkte spielen bei der Auswahl der geeigneten Fileserver-Software eine tragende Rolle. In Abschnitt 1.4 wird auf das Thema Fileserver-Software eingegangen.

[1] Ein Rechner über den der Benutzer angebotene Dienste in Anspruch nimmt.

[2] Portabilität ist die Verfügbarkeit einer Software für unterschiedliche Betriebssysteme. Ist sie auf vielen Betriebssystemen verfügbar, so wird von einer hohen Portabilität gesprochen.

1.2 Lokale Speicher und Speichernetze

Ein Speicher[3], der Nutzdaten enthält (ohne Betriebssystem, Fileserversoftware und sonstiger Anwendungen), kann sich an verschiedenen Orten im Netzwerk befinden. Die in Abschnitt 1.1 gezeigte Abbildung 1.1 ist nur eine vereinfachte Darstellung. Sie entspricht, bezogen auf die Lokalisierung des Speichers, der von Abbildung 1.2. Hier liegt eine zentralisierte Speicherung der Nutzdaten vor. Ein Beispiel ist in dieser Abbildung 1.2 dargestellt. Client A bezieht (durch eine gepunktete Linie dargestellt) Daten über das Netzwerk direkt von dem an den Fileserver I angeschlossenen Speicher (mit einem Verbund aus Fest-Speichern dargestellt). Dieser Speicher wird hier als serverzentrierter Speicher bezeichnet.

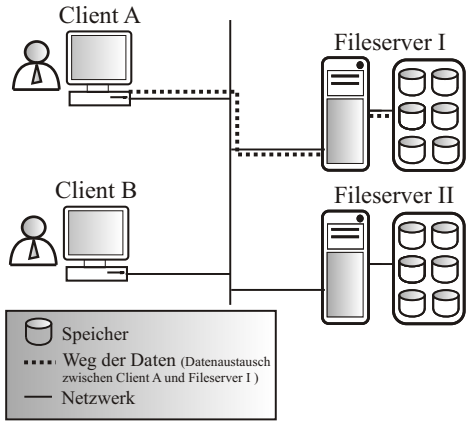

Abbildung 1.2: Lokale Speicherung der Daten auf dem Fileserver

1.2.1 Storage Area Network (SAN)

Speichernetze dienen in erster Linie dazu die Verfügbarkeit des Speichers zu erhöhen. SAN's sind unabhängige Speicherverbünde, welche sich außerhalb der eigentlichen Fileserverhardware befinden und über ein spezielles Netzwerk, dem Speichernetzwerk, mit den Fileservern verbunden sind.

In Abbildung 1.3 ist ein Storage Area Network zu sehen (eingerahmt durch eine gestrichelte Linie). Es ist über ein spezielles Netzwerk direkt mit den Fileservern verbunden. In Abbildung 1.3 bezieht (durch gepunktete Line dargestellt) beispielsweise Client A die Daten über das Netzwerk von Fileserver I. Dieser erhält die Daten über die Steuerungslogik des Speichernetzwerkes von dem dahinter befindlichen Speicherverbund. Über die Steuerungslogik wird die Verteilung

[3] Dieser Speicher (englisch: Storage) besteht nur aus Festplattenverbünden (Lesen und Schreiben) oder auch Compact-Disk-Verbünden (nur Lesen) und der dazu notwendigen Logik, um diese anzusprechen und zu betreiben.

der Speicherkapazität geregelt (jeder Fileserver erhält exklusiv einen Teil der Speicherkapazität), die Zugriffe auf den Speicher koordiniert und dieser verwaltet.

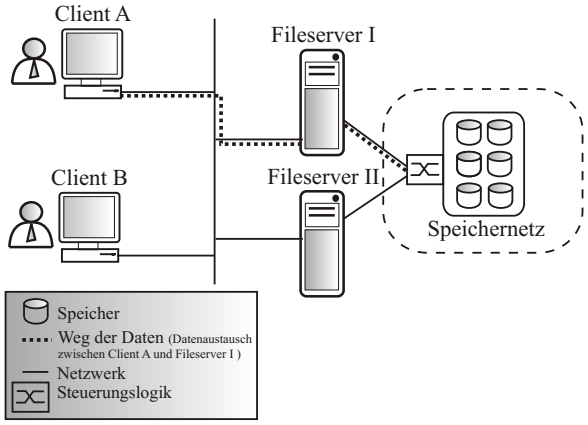

Abbildung 1.3: Storage Area Network (SAN)

Als Technologie des Speichernetzwerkes werden zwei blockorientiert arbeitende Netzwerk-Protokolle genutzt. Der Fibre Channel Standard (Abkürzung: FC) wurde ursprünglich als Ablösung für Fast-Ethernet (100 MBit/s) entwickelt. In Bezug auf die Transferraten wurde er aber von Gigabit-Ethernet (1 GBit/s) und 10Gigabit-Ethernet (10 GBit/s) überholt[Pec02]. Er besitzt aber positive Eigenschaften für den Einsatz in Speichernetzen. Das sind zum Beispiel die serielle Übertragung für weite Entfernungen und hohe Geschwindigkeiten, eine geringe Wahrscheinlichkeit an Übertragungsfehlern und die geringe Verzögerung (Latenz) der übertragenen Daten. Die Übertragungsgeschwindigkeiten betragen 100 MByte/s (800 MBit/s) oder 200 MByte/s (1600 MBit/s). Übertragungsmedien sind sowohl Kupferkabel als auch Glasfaserkabel[Ste03]. Das neue InfiniBand stellt wie Fibre Channel eine serielle Übertragung dar. Es ist aber von Beginn an speziell auf Speichernetze optimiert. Die Geschwindigkeiten reichen hier, unter Einsatz von Übertragungskanalbündelung, von 10 GBit/s bis zu maximal 30 GBit/s. Als Übertragungsmedien werden Kupferkabel und Glasfaserkabel eingesetzt.

1.2.2 Network Attached Storage (NAS)

In Abbildung 1.4 ist ein Network Attached Storage (NAS) zu sehen. Dieser unabhängige Speicher wird über das normale Netzwerk mit den Fileservern verbunden und arbeitet dateiorientiert. In Abbildung 1.4 bezieht (durch gepunktete Line dargestellt) Client A Daten über das Netzwerk von Fileserver I. Dieser erhält die angeforderten Daten über das gleiche Netzwerk vom NAS-Speicher.

Das genutzte Protokoll iSCSI wurde als Standard im Jahre 2002 verabschiedet. iSCSI ist eine Implementierung des bekannten SCSI-Protokolls[Fie01] zur Übertragung von Daten über das Internet Protokoll (IP)[Kau97]. Die Funktionen und Eigenschaften entsprechen dem normalen SCSI-Standard, zusätzlich angepasst auf die Übertragung über ein IP-Netzwerk. Weiterhin wird in [Ste03] als zukünftiger Standard iFCP genannt. Dieses Protokoll ist eine Implementierung des Fibre Channel-Protokolls für die Übertragung von Daten über ein IP-Netzwerk. Die Übertragungsgeschwindigkeiten beider Protokolle sind vom eingesetzten Netzwerk abhängig (100 MBit/s bis 10 GBit/s).

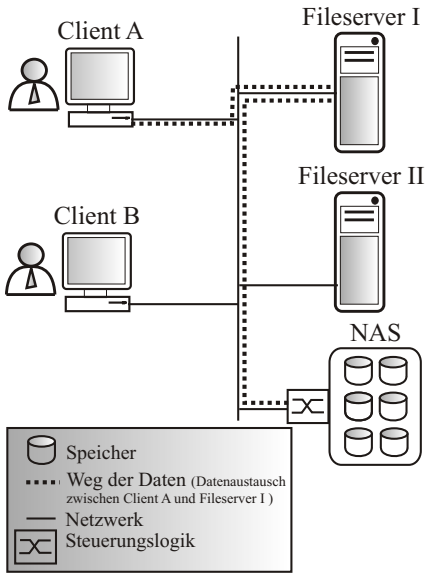

Abbildung 1.4: Network Attached Storage (NAS)

1.2.3 Vergleich der Speicherstrategien

Serverzentriert Der Vorteil dieser Art der Speicheranbindung sind die geringen Kosten gegenüber den anderen Speicherstrategien. Der Speicher wird über die interne Hardware lokal an den Fileserver angeschlossen. Der Nachteil ist die geringere Verfügbarkeit. Fällt der Fileserver aus, so sind auch die Daten auf dem Speicher nicht mehr erreichbar. Erst nach der Beseitigung der Fileserver-Störung kann auf die Daten wieder zugegriffen werden.

SAN Fibre Channel und der neue InfiniBand-Standard setzen kostenintensive Spezialnetzwerkinfrastrukturen voraus. Fibre Channel wird aber in Zukunft durch seinen Nachfolger iFCP (Übertragung über Standard-Netzwerkinfrastruktur) und dessen deutlich höhere Übertragungsgeschwindigkeit abgelöst. Bei einem Ausfall eines Fileservers wird mittels der intelligenten

Steuerungslogik, welche sich zwischen dem Speicher und den Fileservern befindet, auf den zweiten Fileserver oder einen Ersatz-Server gewechselt. Der Speicher ist so unmittelbar wieder verfügbar.

NAS Für die Verfügbarkeit treffen hier die gleiche Aussagen wie auf die SAN's zu. Durch die Nutzung von konventionellen und damit preisgünstigen Standard-Netzwerken wird sich auch Network Attached Storage mit den beiden neuen Standards iSCSI und iFCP am Speichernetzmarkt etablieren[Ste03].

1.3 Dateisysteme

Daten sind ein wesentlicher Bestandteil der heutigen elektronischen Datenverarbeitung. In Rechnern verarbeiten Anwendungen Daten und sie selbst bestehen aus Daten. Dateisysteme organisieren, verwalten, speichern, bearbeiten und laden Daten aller Art auf physikalischen- oder auch logischen Datenträgern. Die Begriffe physikalische und logische Datenträger werden in Abschnitt 1.3.4 erläutert.

Dateisysteme ermöglichen den Zugriff auf nichtflüchtige Speicher[4]. Dieses ist notwendig, da der Rechner bei der Ausführung von Anwendungen Daten von nichtflüchtigen Datenträgern in seinen Arbeitsspeicher lädt, um mit diesen zu arbeiten. Der Arbeitsspeicher des Rechners ist flüchtig und in seiner Größe[5] aus Kostengründen und durch seine Technologie beschränkt. Ist der Arbeitsspeicher belegt, lagert der Rechner (genauer die Zentrale Recheneinheit) durch entsprechende Mechanismen Teile von Daten, die er im Moment nicht benötigt, auf den nichtflüchtigen Speicher aus, um diese bei Bedarf von dort wieder zu laden.

1.3.1 Die Ebenen des Dateisystems

In Abbildung 1.5 ist die Architektur eines Dateisystems dargestellt, welches sich in drei Ebenen mit verschiedenen Aufgaben aufteilt. An die unterste Ebene sind die physischen Geräte, welche die Daten in Blöcken[6] bereithalten, angeschlossen. Die oberste Ebene (Logische Ein-/Ausgabe) ist mit der Benutzer-/Anwendungsschnittstelle des Betriebssystems verbunden. Diese Darstellung wird als allgemeines Modell genutzt, um die Funktionsweise eines Dateisystems

[4] was bedeutet, nach dem Ausschalten des Rechners bleiben die Daten erhalten, im Gegenteil zum flüchtigen Speicher, der nach Ausschalten des Rechners seine Inhalt (Daten) verliert. Anmerkung: Dateisysteme können auch auf flüchtigen Speichern angelegt werden !

[5] Heutige Rechner besitzen im Durchschnitt 128 MByte bis einige Gigabyte große Arbeitsspeicher.

[6] Der Block ist die kleinste Daten-Einheit auf einem Datenträger. Blockgrößen von 512 Byte bis 4096 Byte sind heutzutage üblich.

zu erläutern. Die Ebenen des Dateisystems sind fest in das Betriebssystem integriert. Es ist aber möglich mehrere unterschiedliche Dateisysteme innerhalb eines Betriebssystems zu nutzen. Allerdings ist immer nur ein Dateisystem pro physikalischem oder logischem Datenträger möglich (siehe 1.3.4).

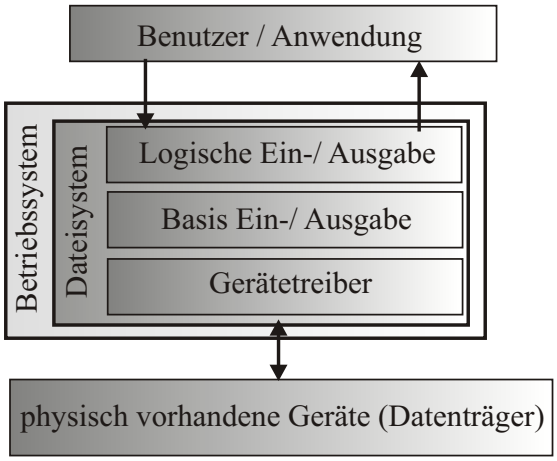

Abbildung 1.5: Dateisystem Ebenen

Gerätetreiber-Ebene

Gerätetreiber sind Softwareroutinen, welche auf der niedrigsten der drei Ebenen arbeiten. Sie haben direkten Zugriff auf die physikalischen Geräte. Sie sind verantwortlich für das direkte Lesen und Schreiben der auf den Geräten enthaltenen Daten. Die Übertragung der Daten findet hier in Blöcken fester Größe statt. Gerätetreiber sind an ein Betriebssystem gebunden und nur für ein oder mehrere Geräte eines Herstellers funktionsfähig. Jedes Gerät benötigt somit seinen eigenen speziellen Gerätetreiber.

Basis-Ein-/Ausgabe Ebene

Die Basis-Ein-/Ausgabe ist die Ebene des Dateisystems, welche über die Gerätetreiber mit den einzelnen Geräten kommuniziert. In dieser Ebene wird die Datenintegrität, also die fehlerfreie Transferierung der Daten, sichergestellt. Ein Mechanismus zur Sicherstellung der Datenintegrität wird in Abschnitt 1.3.2 erläutert. Weiterhin werden hier die verschiedenen Aufträge (auszuführende Transfers) koordiniert (korrekte Reihenfolge). Die Basis-Ein-/Ausgabe Ebene schafft eine einheitliche Schnittstelle für den blockorientierten Datentransfer. Die spezifischen Geräteeigenschaften bleiben vor den höheren Schichten verborgen.

Logische Ein-/Ausgabe Ebene

Diese Ebene ermöglicht den Zugriff auf die eigentlichen logischen Datensätze[7] und nicht mehr auf die einzelnen Datenblöcke. Eine Datei stellt damit eine Sammlung von Datensätzen dar, auf welche der Anwender, ohne Rücksicht auf die beteiligten Blöcke, zugreifen kann. Ein Datensatz kann sich je nach Größe über mehrere Blöcke verteilen.

Es existieren drei Organisations-Strukturen für Datensätze in Dateien. Alle Dateisysteme bauen auf einer dieser Strukturen auf. Welche Dateiysteme welche Strukturen verwenden wird später in der speziellen Betrachtung der einzelnen Dateisysteme erläutert (siehe 1.3.2).

Die erste Struktur ist die sequentielle Anordnung der Datensätze. Diese besitzt den Vorteil, daß sie auf allen Speichern (Bandlaufwerke sind typische Vertreter) eingesetzt werden kann. Die Länge der Datensätze wird nur durch die Speicherkapazität der Geräte begrenzt. Nachteile der sequentiellen Anordnung sind die eingeschränkten Zugriffsmöglichkeiten, da auf die Datensätze nicht wahllos zugegriffen werden kann. Es gibt keine Schlüsselinformationen, wo sich welche Daten befinden. Die nächste Struktur sind die relativen Dateien, welche über einen ganzzahligen Schlüssel pro Datenblock verfügen und somit den direkten Zugriff auf einen Datensatz ermöglichen. Die modernste Struktur sind die direkten Dateien. Sie sind eine Erweiterung der relativen Dateien. Hier sind beliebige Schlüssel erlaubt, deren Reihenfolge keine Rolle mehr spielen. Der Zusammenhang von Datensätzen wird hier durch einen Zeiger, der auf den nächsten Schlüssel (Datensatz) zeigt, gewährleistet. In modernen Dateisystemen sind diese direkten Dateien in Baumstrukturen organisiert, was zu einer Optimierung der Zugriffszeiten führt (Komplexität des Suchalgorithmus [Sed92]).

Die Logische Ein-/Ausgabe Ebene transferiert mit Hilfe der Basis-Ein-/Ausgabe die Datenblöcke in den Ein-/Ausgabepuffer im Arbeitsspeicher. Es wird für jede Datei ein eigener Puffer angelegt. Nur die Größe des Arbeitsspeichers begrenzt die Menge der möglichen Ein-/Ausgabepuffer. Durch diese Pufferung auf Blockebene kann eine erhebliche Leistungssteigerung erzielt werden, indem die Arbeit der Basis-Ein-/Ausgabe verringert wird. Sind die angeforderten Datenblöcke schon im Puffer vorhanden, müssen diese nicht neu von den Geräten transferiert werden. Durch das verzögerte Schreiben eines Datenblockes wird dieser erst nach einer vom Dateisystem festgelegten Zeitspanne (nach dem kein Zugriff mehr auf diesen stattgefunden hat) auf das Gerät zurückgeschrieben.

Benutzer- / Anwendungsschnittstelle

Über der Logischen Ein-/ Ausgabe befindet sich diese Schnittstelle, welche dem Nutzer oder der Anwendung eine Datei als abstrakten Datentyp zur Verfügung stellt. Das Dateisystem stellt

[7] Datensatz, Menge zusammengehöriger Informationen, welche logisch als eine Einheit behandelt werden.

über Systemaufrufe Operationen, welche im folgenden beschrieben werden, zur Verfügung. Über diese Aufrufe kommuniziert das Betriebssystem mit dem Dateisystem.

Erzeugen und Entfernen von Dateien

Bevor mit einer Datei gearbeitet werden kann, muss sie angelegt werden. Es wird deren eindeutiger Name in einem Verzeichnis eingetragen, in welchem alle im System existierenden Dateien einen Eintrag besitzen. Die Verzeichnisse sind meist selber spezielle Dateien und sequentiell oder baumartig aufgebaut. Beim Anlegen der Datei werden, je nach Dateisystem, verschiedene Attribute festgelegt. Das sind zum Beispiel Zugriffsrechte, Anzahl der belegten Datensätze und Informationen über die Dateistruktur. Beim Entfernen wird der Verzeichnisseintrag mit den gesamten Informationen gelöscht und die belegten Ressourcen (Speicherplatz) wieder freigegeben.

Öffnen und Schließen einer Datei

Um einen Datei-Zugriff zu ermöglichen, muss die Datei beim System angemeldet werden. Als eineindeutiger Identifikator dient der Name, welcher ihr bei der Erstellung gegeben wurde. Während des Öffnens führt das Dateisystem eine Reihe von Tests durch, wie zum Beispiel ob der Prozess[8], welcher auf die Datei zugreifen will, die dafür nötigen Zugriffsrechte besitzt. Der Prozess bekommt nach dem erfolgreichen Bestehen aller Tests einen eindeutigen Dateibezeichner zurück, der bei allen Operationen auf die Datei verwendet wird. Weiterhin wird ein Ein-/Ausgabepuffer mit den Inhalten der Datei gefüllt. Hat ein Prozess die Bearbeitung einer Datei abgeschlossen, so wird diese vom System abgemeldet. Jetzt wird der für diese Datei benutze Ein-/Ausgabepuffer geleert, die Inhalte auf die Datenträger zurückgeschrieben und die benutzten Ressourcen (Arbeitsspeicher) wieder freigegeben.

Der Zugriff auf die Datensätze

Je nach der Struktur, wahlfrei oder sequentiell, können Prozesse entweder lesend oder schreibend auf die Datensätze zugreifen. Wahlfrei bedeutet, es wird über einen eindeutigen Schlüssel auf die gewünschten Datensätze direkt zugegriffen. Bei einem sequentiellen Zugriff werden die Datensätze nacheinander gelesen oder geschrieben. Das Dateisystem benötigt dabei nur die Informationen an welcher Position des Datensatzes der Zugriff beginnt und wie groß dieser ist.

Für das Dateisystem ist es daher wichtig, die Position des aktuellen Datensatzes zu kennen. Ohne dieses Wissen ist kein sequentieller Zugriff möglich. Daher wird die aktuelle Position, innerhalb einer Datei, durch einen *Dateizeiger* markiert. Dieser wird bei allen Lese- und Schreibzugriffen aktualisiert (Dateizeiger wird verschoben). Für den Fall das mehrere Prozesse auf die gleiche Datei zugreifen wird ein sogenannter *privater Dateizeiger* benutzt. Jeder dieser Prozes-

[8] mit Prozess ist hier eine Anwendung oder eine Benutzeraktion gemeint , der Prozess ist eine eigene Einheit innerhalb eines Betriebssystem, welcher Aktionen aller Art ausführt, z.B. Programm, Dienst

se besitzt einen eigenen *privaten* Dateizeiger mit dem er auf die Datei zugreift. Im weiteren existieren noch *gemeinsame Dateizeiger*, welche von mehreren Prozesse gemeinsam benutzt werden[Lam97].

Das Entfernen einzelner Datensätze wird durch die Logische Ein-/Ausgabe realisiert. In einer Schlüssel-basierten Struktur werden die Datensätze unter Angabe ihres Schlüssels entfernt. Bei sequentiellen Strukturen ist es nicht möglich Datensätze in der Mitte einer Datei zu löschen. Es muss eine Verkürzung stattfinden, wobei nur Datensätze am Ende der Datei gelöscht werden können[Lam97].

Bei jedem Dateizugriff muss die Konsistenz[9] der Daten erhalten und garantiert werden. Greifen beispielsweise zwei Prozesse auf eine Datei zu, der Erste lesend und der Zweite schreibend, muss der Erste entweder nach dem Schreiben oder vor dem Schreiben des Zweiten die Datensätze lesen. Andernfalls würde der erste Prozess inkonsistente Daten lesen. Das Dateisystem stellt dazu Mechanismen zur Verfügung, welche die Unterbrechung eines Dateizugriffes verhindern[Her99]. Als Beispiel-Mechanismus seien hier Semaphoren genannt. Können Zugriffe auf Dateien nicht unterbrochen werden, so wird von *atomaren Zugriffsoperationen* gesprochen. Einige Dateisysteme sind sogar in der Lage einzelne Datensätze innerhalb einer Datei zu sperren.

1.3.2 Serverdateisysteme

Dateisysteme sind hochoptimierte Schnittstellen zwischen den eigentlichen Geräten (Festplatten, Bandlaufwerken, CD-Brennern usw.) und dem Betriebssystem sowie verschiedener Anwendungen. Sie ermöglichen die Darstellung der einzelnen Blöcke, welche sich auf den Geräten befinden, als abstrakten Datentyp, der Datei. In diesem Abschnitt werden Dateisysteme betrachtet, die für einen Einsatz auf einem Fileserver in Betracht kommen. Die Kriterien für eine solche Eignung wurden wie folgt definiert:

1. Dateisystemgröße von mindestens 16 TeraByte
2. Dateigrößen von mindestens 10 TeraByte
3. Unterstützung von Dateiberechtigungen (Rechtevergabe, ACL's)
4. Unterstützung von Journaling

Das erste Kriterium ist berechtigt, da sich laut [Ric03] die momentane verfügbare und benötigte Speicherkapazität pro Jahr verdoppelt. Dieser hohe Zuwachs an Speicherbedarf resultiert aus

[9] Datenkonsistenz bedeutet: Die Daten befinden sich in einem geordneten, fehlerfrei lesbaren und definierten Zustand

der zunehmenden Nutzung von speicherintensiven Anwendungen wie Videobearbeitung, Audiobearbeitung und Wissenschaftlichen Daten (zum Beispiel Satellitenfotos (größer 1 GByte)). Auch beim zweiten Punkt sind die gerade aufgeführten Anforderungen für diese Mindestgröße verantwortlich. Umfangreiche Wissenschaftliche Messreihen, welche in Dateien mit mehreren Gigabyte Umfang gespeichert werden, sind in der Forschung keine Seltenheit.

Der dritte Punkt bezieht sich auf die Rechtevergabe auf Dateien und Verzeichnisse innerhalb des Dateisystems. Diese Funktionalität ist eine der Kernanforderungen für ein Serverdateisystem. Der Grund dafür liegt in der Rolle, die ein Server darstellt. Er stellt seine Dienste immer mehr als einem Benutzer zur Verfügung. Die Daten der unterschiedlichen Benutzer müssen vor dem Einblick durch nicht autorisierte Benutzer geschützt werden. Zum besseren Verständnis wird diese Problematik am Beispiel der Linux/Unix Rechtevergabe näher betrachtet. [Sie99].

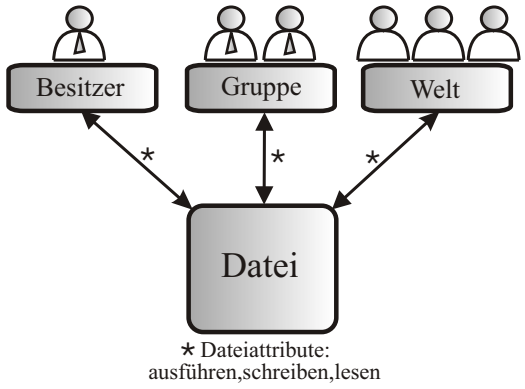

Abbildung 1.6: Datei-Rechte

In Abbildung 1.6 ist die Datei[10] als zentraler Punkt zu sehen. Die Pfeile stellen die Wechselbeziehungen, im Sinne der Rechtevergabe, mit den verschiedenen Benutzer-Gruppen dar. Sie verfügt über drei Rechte-Attribute:

- **ausführbar**, die Datei ist ausführbar/startbar, zum Beispiel eine Anwendung
- **schreiben**, in die Datei darf geschrieben, der Inhalt darf verändert werden
- **lesen**, der Inhalt der Datei darf gelesen werden

Diese drei Datei-Attribute werden jeweils an drei Benutzergruppen vergeben. Die erste stellt den Besitzer oder Ersteller der Datei dar. Die zweite ist die Gruppe welcher die Datei gehört[11].

[10] Rechte werden auch auf Verzeichnisse angewendet

[11] In einer Gruppe können zum Beispiel alle Mitarbeiter einer Abteilung enthalten sein, welche Zugriff auf Abteilungsdokumente benötigen. Ein Benutzer kann auch Mitglied in mehreren Gruppen sein.

Die dritte Gruppe stellt die Benutzer, welche weder die Besitzer/Ersteller noch in der Gruppe der Datei Mitglied sind. Diese dritte Gruppe wird als die restliche *Welt* bezeichnet.

Ein einfaches Beispiel für eine Rechtevergabe auf eine Datei könnte dem Besitzer/Ersteller die vollen Schreib- sowie Leserechte geben. Die Gruppe in welcher er Mitglied ist dürfte die Datei nur Lesen. Die letzte Benutzergruppe, die Welt, darf die Datei weder verändern noch Lesen ! Ein Praxis-Beispiel für eine derartige Rechtevergabe ist die Finanzabteilung einer Firma. Dort werden Lohnscheine in Dateien gespeichert. Nur der Sachbearbeiter darf die Lohnscheindatei verändern, andere Sachbearbeiter aus der Finanzabteilung dürfen diese nur lesen. Mitarbeiter aus anderen Abteilungen dürfen die Lohnscheindatei weder lesen noch in diese schreiben (geheimer Inhalt).

Der im dritten Punkt der Kriterien genannte Begriff *ACL's* [12] bezieht sich auf ein Rechtevergabesystem auf Dateien und Verzeichnisse welches eine feinere detailliertere Vergabe von Rechten vorsieht als im vorher genannten Unix-Rechtesystem Beispiel. Mit ACL's ist es möglich Zugriffsrechte einer Datei oder eines Verzeichnisses auf mehrere Benutzer oder Gruppen zu setzen. Weiterhin stehen zusätzliche Rechteattribute zur Verfügung[Die03]. In Tabelle 1.1 wird anhand des NTFS-Dateisystems (siehe Seite 16) ein Beispiel für erweiterte Attribute und deren Bedeutung aufgezeigt.

Attribut	Bedeutung
Attribute lesen	die Attributliste wird angezeigt, kann gelesen werden
Erweiterte Attribute lesen	die erweiterte Attributliste wird angezeigt, kann gelesen werden
Attribute schreiben	Attribute können verändert, geschrieben werden
Erweiterte Attribute schreiben	Erweiterte Attribute können verändert, geschrieben werden
Unterverzeichnisse und Dateien löschen	Verzeichnisse und Dateien können gelöscht, werden
Berechtigungen lesen	die einfachen Berechtigungen können gelesen werden (siehe Linux/Unix-Rechte)
Berechtigungen ändern	die einfachen Berechtigungen dürfen verändert werden
Dateibesitz übernehmen	dieses Attribut erlaubt das Verändern des Erstellers/Besitzers einer Datei

Tabelle 1.1: Erweiterte Attribute, NTFS 5.0 (Auszug) [Kup00]

Als vierter und letzter Punkt der Serverdateisystem-Kriterien wird die Unterstützung von Journaling angeführt[Die02]. Nach einem Systemabsturz des Servers, zum Beispiel durch Stromausfall, befindet sich das Dateisystem in einem inkonsistenten Zustand. Dies resultiert aus den

[12] *ACL:* Access Control Lists, deutsch: Zugriffs Kontroll Listen

in Abschnitt 1.3.1 auf Seite 8 angesprochenen Pufferspeichern, welche Daten erst verzögert auf die Datenträger schreiben. Stürzt das System ab, ist es möglich, daß einige Daten schon auf den Datenträger geschrieben wurden und andere sich noch im Puffer befinden. Das Dateisystem ist inkonsistent.

Die Konsistenz eines Dateisystems hängt weiterhin von den Metadaten, den internen Strukturen der auf den Datenträgern gespeicherten Daten, ab. Diese Metadaten legen unteranderem die Beziehung zwischen Dateien und den zugehörigen Datenblöcken fest. Zu diesen gehören auch die Bitmaps. Das sind Abbildungen in welchen die freien und belegten Blöcke des Datenträgers verzeichnet sind. Verweisen zum Beispiel die Metadaten einer Datei auf nicht existierende Datenblöcke oder gilt laut Datenträgerbitmap ein Block als leer und wird mit anderen Daten beschrieben, obwohl dieser schon belegt war, ist die Konsistenz nicht mehr gewährleistet. Ohne eine Journaling-Funktion im Dateisystem wird beim Systemstart auf fehlerhafte Beziehungen zwischen Dateien und zugehörigen Datenblöcken geprüft und diese gegebenenfalls wiederhergestellt. Dieser Vorgang nimmt bei den heutigen Datenträgergrößen (n TeraByte) mehrere Stunden bis Tage in Anspruch. In dieser Zeit ist der Server nicht verfügbar und kann somit seine Dienste nicht anbieten.

Die Konsistenz eines Dateisystems wird in den häufigsten Fällen von äußeren Einflüssen negativ beeinflusst. Dazu zählen zum Beispiel defekte Kabelverbindungen in den Rechnern, ungeplante Stromausfälle und weitere widrige Umstände.

In diesen Fällen sorgt das Journaling dafür, daß die Konsistenz des Dateisystem schnell wieder hergestellt wird. Das Journaling arbeitet dabei mit einer doppelten Buchführung. Stehen Veränderungen am Dateisystem an (Daten werden geschrieben oder verändert) lässt es zuerst die alten konsistenten Metadaten des Datenträgers unangetastet. Die neuen Metadaten schreibt es in einen reservierten Bereich (Journal oder auch Log genannt) des Datenträgers. Es fasst dabei einzelne Aktionen, wie Dateien anlegen, kopieren, löschen, zu einer Transaktion zusammen und vermerkt deren erfolgreiche Ausführung im Log.

In diesem Journal stehen dann ein gültiger Satz neuer Metadaten, sowie alle Transaktionen durch welche diese entstanden sind. Sind die Veränderungen am Dateisystem abgeschlossen, markiert das Journaling die neuen Metadaten als gültige Metadaten des Dateisystems. Nach einem Systemabsturz muss das Dateisystem das Transaktions-Log wieder abspielen und prüfen ob alle Operationen erfolgreich abgeschlossen wurden. Ist dies der Fall sind die neuen Metadaten konsistent und können benutzt werden. Ist nur eine Operation durch den Systemabsturz nicht erfolgreich abgeschlossen worden, werden die alten noch gespeicherten konsistenten Metadaten genutzt.

Die Integrität der Daten selber ist aber nicht gewährleistet, da diese nicht mit in das Journal gesichert werden (Ausnahme: ext3, siehe Seite 18). Nach einem Systemabsturz kann somit ein

Gemisch von alten und neuen Daten vorliegen. Ein Journalingdateisystem schützt somit nicht vor Datenverlust (siehe 1.6). Der große Vorteil ist die schnelle Verfügbarkeit eines konsistenten Dateisystems nach einem Systemabsturz.

Im Anschluss wird auf die Funktionsweise, Vor-/Nachteile und Besonderheiten ausgewählter Serverdateisysteme eingegangen. Bewusst werden hier nur Dateisysteme angesprochen, welche sich in einem stabilen Entwicklungszustand befinden. Das NTFS-Dateisystem, welches vollständig implementiert nur für die Microsoft Betriebssysteme (Windows NT, Windows 2000, Windows XP und deren jeweilige Servervarianten) verfügbar ist, wird hier nur deswegen behandelt da es das eingesetzte Dateisystem auf den Microsoft-Windows Clients (Benutzerrechnern) des UFZ ist.

NTFS

Die Abkürzung NTFS steht für **N**a**T**ive **F**ile **S**ystem. Es wurde 1993 erstmals vom Entwickler Microsoft in seinem Serverbetriebssystem Windows NT 3.1 eingeführt. NTFS verwaltet jede Datei und jedes Verzeichnis als separates Objekt, welches jeweils eigene Rechte sowie Attribute besitzt[Sha00]. Es gibt zwei Arten von Dateien. Die ersten enthalten Metadaten (Informationen über den Datenträger) und die zweiten bestehen aus den eigentlichen Daten.

Abbildung 1.7: NTFS Dateisystem Datenstruktur

In Abbildung 1.7 ist die Struktur der Daten, welche NTFS auf dem Datenträger anlegt, dargestellt [Rus03]. Der erste Abschnitt stellt den Boot-Teil[13] dar. Nach dem Start des Rechners und dem erfolgreichen Ablauf der BIOS[14]-Startsequenz lädt der Rechner den Bootteil des ausgewählten Datenträgers. Nach dem Bootteil folgt bei NTFS die MFT [15], deren Inhalt in Tabelle 1.2 aufgeführt ist.

[13] Im Bootteil stehen die Positionen der Programmteile welche das Betriebssystem des Rechners starten. Jedes Betriebssystem speichert im Bootteil seine Bootsequenz. Der Bootteil selbst ist nur auf den Datenträgern oder Partitionen (siehe 1.3.4) vorhanden welche ein Betriebssystem enthalten.

[14] *BIOS:* mit dessen Hilfe werden die gesamten Komponenten des Rechners (CPU, Speicher, Laufwerke, Grafikkarten etc.) nach dem Einschalten initialisiert und geprüft und in Betriebsbereitschaft versetzt. Es verfügt über eine grafische Konsole mit deren Hilfe umfangreiche Konfigurationseinstellungen an den Rechnerkomponenten möglich sind.

[15] *MFT:* Master File Table, deutsch Haupt Datei Tabelle

[16] siehe Seite 12

MFT Datum	Bedeutung
$MFT	Master File Table, Verzeichnis aller Dateien auf dem Datenträger Inhalt siehe Tabelle 1.3
$MFTMirr	Sicherheitskopie der MFT
$Volume	Bezeichnung des Datenträgers, Datenträger ID
$Logfile	Zeiger auf die Logdatei [16]
$Boot	Zeiger auf den Bootbereich des Datenträgers
$AttrDef	Attribute des Datenträgers (lesender,schreibender Zugriff)
$BadClus	Liste der defekten Blöcke/Bereiche des Datenträgers
$Quota	Für den Datenträger festgelegte Kapazitätsbeschränkung (siehe 1.3.3)
$Secure	Sicherheitsinformationen des Datenträgers, Zertifikate, Schlüssel
$Extend	Reserviert für weitere Metainformationen des Datenträgers
$Bitmap	kennzeichnet die belegten Blöcke des Datenträgers

Tabelle 1.2: Daten der MFT [Rus03]

Die in Tabelle 1.2 aufgeführten Inhalte beziehen sich auf den jeweiligen Datenträger, auf welchem sich die MFT befindet. Das MFT Datum $MFT ist das Verzeichnis aller auf dem Datenträger befindlichen Dateien. Jede existierende Datei besitzt in $MFT die in Tabelle 1.3 aufgeführten Metadaten.

Metadatum	Bedeutung
$FILE_NAME	Name der Datei
$ATTRIBUTE_LIST	Liste der Datei-Attribute, siehe Tabelle 1.1
$STANDARD_INFORMATION	Liste mit Informationen wie Erstellungsdatum, einfache Zugriffsberechtigungen, Datum des letzten Lesens, Versionsnummer
&DATA	enthält die Zeiger auf die eigentlichen Daten der Datei (Position auf dem Datenträger)
$OBJECT_ID	eindeutige Identifikation der Datei
$SECURITY_DESCRIPTOR	enthält den Namen des Besitzers und dessen Gruppenmitgliedschaft
$BITMAP	zeigt an welche Bereiche der Datei momentan in Benutzung sind

Tabelle 1.3: MFT-Metadaten jeder Datei [Rus03]

Nach der MFT folgen die eigentlichen Daten. Werden die Metainformationen der MFT im reservierten Bereich zu groß, kann zusätzlich eine MFT auf einer freien Stelle des Datenträgers angelegt werden (angedeutet in Abbildung 1.7).

NTFS verfügt über mehrere Eigenschaften die es als Serverdateisystem auszeichnen. Es ist in der Lage die verwalteten Daten in Echtzeit zu verschlüsseln/entschlüsseln. Jeder Benut-

zer verfügt über ein Schlüsselpaar (privat/öffentlich), mit dem er in der Lage ist seine Dateien/Verzeichnisse zu verschlüsseln[vOu96]. Andere Benutzer können diese verschlüsselten Dateien nicht mehr lesen oder beschreiben. Ein Journaling-Mechanismus der Firma Veritas ist implementiert und gewährt Dateisystemkonsistenz nach Systemabstürzen. Der Verweis auf die Log-Datei befindet sich in der MFT und trägt die Bezeichnung $Logfile (siehe Tabelle 1.3). Der in der gleichen Tabelle aufgeführte Eintrag $Quota bezieht sich auf die Möglichkeit eine Speicherplatzbeschränkung pro Benutzer zu realisieren. Damit wird administrativ/zentral festgelegt, wieviel Speicher der Benutzer auf dem Datenträger mit seinen Daten belegen darf (siehe 1.3.3).

Weiterhin unterstützt NTFS:

1. sowohl physikalische als auch logische Datenträger (siehe 1.3.4)
2. Datenträgergrößen sowie Dateigrößen von maximal 1,84 x 10^{10} Gigabyte
3. Access Control Lists (siehe Seite 14)
4. Erweiterte Attribute (siehe Tabelle 1.1)
5. variable Einstellung der Blockgrößen von 512 Bytes bis zu 64 KBytes.

Die Nachteile von NTFS liegen in seiner Portabilität. Vollständig implementiert ist es nur in den Microsoft Betriebssystemen. Ein Hindernis bei der objektiven Bewertung von NTFS ist die mangelhafte Dokumentation. Die Vorteile von NTFS sind die angesprochene Echtzeitverschlüsselung von Dateien und Verzeichnissen, der integrierte Journaling Mechanismus sowie die Verwendung von ACL's und erweiterten Attributen.

ext3

Das ext3 Dateisystem ist fester Bestandteil des Linux-Betriebssystemkernels. Somit ist es auf allen Rechnerplattformen verfügbar, auf welchen Linux als Betriebssystem lauffähig ist. Die Datenträgerstruktur von ext3 ist typisch für die meisten der existierenden Unix/Linux-Dateisysteme, wie zum Beispiel: ext2, ufs (UNIX File System), s5fs (System V File System), advfs (Digital UNIX File System), XFS (SGI) [17] und ReiserFS. Somit sind die hier getroffenen Erläuterungen (betreffend der Datenträgerstruktur), bis auf die spezifischen Eigenheiten, auch für die genannten Unix/Linux-Dateisysteme gültig.

Ext3 ist der einzig um einen Journaling-Mechanismus erweiterte Nachfolger des ext2-Dateisystems. Somit treffen die Erläuterungen von ext2 auch auf ext3 zu. In Abbildung 1.8 ist die Datenstruktur von ext3, welche sich deutlich von der des NTFS-Dateisystems unterscheidet, zu sehen.

[17] **SGI:** Silicon Graphics Incorporated, Rechner Hersteller

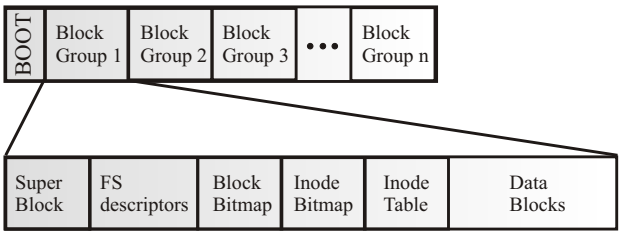

Abbildung 1.8: Ext3 Dateisystem Datenstruktur

Nach dem optional vorhandenen Bootbereich wird der gesamte Datenträger in sogenannte Block-Gruppen, welche durchgängig nummeriert sind, aufgeteilt. Im zweiten Rahmen der Abbildung 1.8 ist der Inhalt einer solchen Block-Gruppe dargestellt. [Die00]

Der erste Teil stellt den Superblock dar. In diesem sind Informationen des Datenträgers enthalten, wie zum Beispiel: Name, Seriennummer, Größe, Erstellungsdatum, defekte Blöcke, sowie die Anzahl der Blockgruppen und deren Größe. Dieser Superblock ist in jeder Blockgruppe als Kopie enthalten. Dies ist ein Vorteil von ext3. Bei Beschädigung eines Superblocks wird einfach eine andere Kopie verwendet. Im *FS-descriptors*-Teil[18] stehen Verweise auf momentan geöffnete Dateien. Über diese Deskriptoren greift das Betriebssystem direkt auf die Dateien zu.

Die nächsten zwei Teile (Block- und Inode-Bitmaps) halten fest, welche Datenblöcke und Inodes der jeweiligen Blockgruppe schon belegt sind. Ein Bit mit dem Wert 0 steht für einen freien Block oder Eintrag in der Inode-Tabelle, der Wert 1 für einen belegten Block/Eintrag. Auch bei Ext3 sind drei verschiedene Blockgrößen möglich: 1024 Byte, 2048 Byte und 4096 Byte. Block- und Inode-Bitmaps nehmen jeweils einen Block ein. Je nach Blockgröße verwalten sie also 8192 (1K-Blöcke) bis 32 768 Einträge (4K-Blöcke) pro Blockgruppe. Bei 32 786 Einträgen (4K-Blöcke) ergeben sich 128 MByte adressierbarer Speicher (32 768 Einträge multipliziert mit 4 KByte gewählter Blockgröße).

Der vorletzte Teil ist die Inode-Tabelle[19]. Sie enthält für jede in der jeweiligen Blockgruppe gespeicherte Datei einen Eintrag und ist in Form einer verketteten Liste implementiert. Der Vorgänger des ersten Eintrages ist der letzte Eintrag in der vorhergehenden Blockgruppe. Der Nachfolger des letzten Eintrages ist wiederum der erste Eintrag in der nachfolgenden Blockgruppe. So sind alle Inode-Einträge miteinander verkettet (siehe Abbildung 1.10 auf Seite 22). Ein Eintrag besitzt eine Länge von 128 Byte und enthält die folgenden Informationen:

- Der Typ der Datei (reguläre Datei, Verzeichnis, character device, block device, Pipe, Socket, symbolischer Link/Verknüpfung)

[18] FS steht für File System (deutsch: Dateisystem)
[19] Inode: Abkürzung für Index Deskriptor

- Besitzer der Datei, Gruppenmitgliedschaft der Datei;
- Zugriffsrechte (lesen, schreiben und ausführen für Besitzer, Gruppe und Welt);
- Zeitpunkte der letzten Änderung von Inode und Daten,
- Zeitpunkt des letzten Dateizugriffs sowie Zeitpunkt des Löschens;
- logische Dateigröße;
- Anzahl der von der Datei belegten Datenblöcke;
- Verweise auf Blocknummern der belegten Datenblöcke.

In Abbildung 1.9 ist der Teil des Inodes dargestellt, welcher die Blöcke mit den Verweisen enthält. Über dem dargestellten Teil befinden sich die oben genannten Datei-Informationen. Er besitzt zwölf direkte Zeiger auf die mit Daten belegten Blöcke. Damit wäre es lediglich möglich, je nach gewählter Blockgröße, auf maximal 48 KByte Daten zuzugreifen.

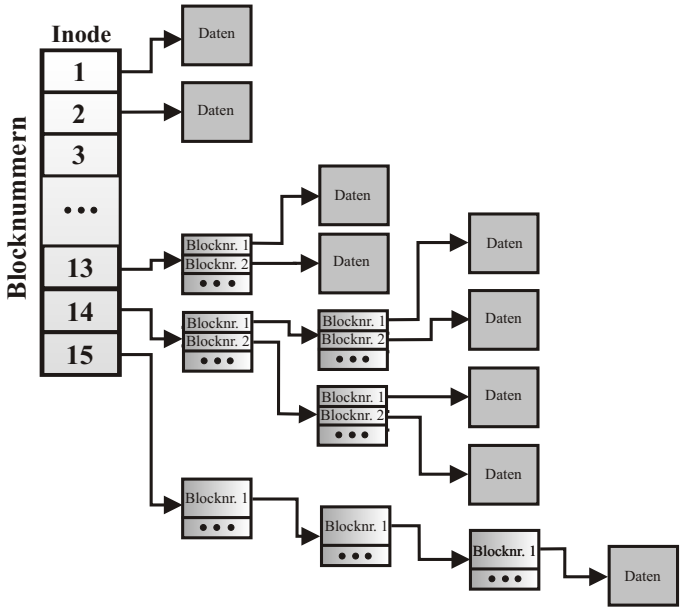

Abbildung 1.9: Verweis-Teil eines Datei-Inodes (Verweise auf die belegten Datenblöcke)

Durch die Blocknummern 13, 14 und 15 werden aber Blöcke adressiert die wieder Listen mit Verweisen auf Daten enthalten. Diese Adressierungsart wird *indirekte Adressierung* genannt

und ist einfach indirekt (Blocknummer 13), zweifach indirekt (Blocknummer 14) und dreifach indirekt (Blocknummer 15) implementiert.

Unter maximaler Ausnutzung der indirekten Adressierung ist es möglich (abhängig von Blockgröße) 16 GByte bis 4 TByte anzusprechen. Das wird aber nicht genutzt, da die maximale Blockgruppengröße nur 128 MByte beträgt. Überschreitet eine Datei diese Größe, so wird ein Verweis in deren Inode-Eintrag auf die nächste Blockgruppe eingetragen. Durch die zwölf direkten Blöcke sind die Daten kleiner Dateien (unter 48 KByte) schnell gefunden. Die indirekte Adressierung besitzt daher einen Nachteil. Je mehr Blöcke, bei der Suche nach indirekt abgelegten Daten, geprüft werden müssen, desto länger dauert es. Der Zeitaufwand bei der Suche in einer Datei steigt somit mit deren zunehmender Größe stetig an!

Die maximale Datenträgergröße und die maximale Dateigröße beträgt 16 TByte. Eine Besonderheit ist der Journaling Mechanismus. Mit diesem ist es möglich, neben der normalen Sicherung der Datenträgerkonsistenz auf Blockebene, die eigentlichen Datensätze konsistent zu halten, also zusätzlich alle Schreibvorgänge zu protokollieren und zu sichern. Es wird immer sichergestellt, daß sich gültige Datensätze auf dem Datenträger befinden. Es besteht nicht die Gefahr, wie bei Journaling Dateisystemen ohne diese Funktionalität, daß nach einem Systemabsturz eine Mischung von alten und neuen Daten vorliegt!

Nachteil ist die Organisation der Inodetabellen. Hier werden verkettete Listen benutzt. Wird eine Datei angefordert, so muss langwierig sequentiell gesucht werden, was sich negativ auf die Performance auswirkt. In anderen Dateisystemen werden B-Bäume als Organisationsstruktur für die Inodetabellen verwendet (siehe Seite 20 (ReiserFS) und Seite 21 (SGI-XFS)).

Die Vorteile von ext3 sind die hohe Systemsicherheit (mehrfache Kopien des Superblocks), der Journaling-Mechanismus, eine gute Performance bei kleinen Dateien und kleineren Datenmengen (kleiner 1TByte) sowie die Unterstützung von ACLs[20]. Ein weiterer Vorteil ist die Möglichkeit ein bestehendes ext2 Dateisystem (Vorgänger von ext3) ohne Datenverlust in ein ext3 Dateisystem mit Journaling zu konvertieren.

ReiserFS

ReiserFS war das erste[21] unter Linux verfügbare Journaling Dateisystem. Der Name stammt von seinem Entwickler Hans Reiser ab. Die Datenstrukturen entsprechen bis auf den massiven Einsatz von Bäumen als Organisationsstruktur denen von ext3, weswegen sie hier nicht noch einmal explizit erläutert werden. Die balancierten und ausgeglichenen Bäume (B-Bäume)[Sed92] werden in den einzelnen Tabellen wie den Blockbitmaps, Inodebitmaps sowie der Inodetabelle

[20] Ausnahme: das in der RedHat-Linux-Distribution (www.redhat.com) enthaltene ext3-Dateisystem besitzt aus vermeintlichen Stabiltätsgründen keine ACL-Unterstützung

[21] Implementierung im Jahre 2001

eingesetzt.

Zum Vergleich der beiden Organisationsstrukuten werden diese jetzt näher betrachtet. In Abbildung 1.10 ist eine Inode-Tabelle von ext3 zu sehen. Die einzelnen Tabellenelemente sind untereinander verkettet und jedes Inode einer Datei zeigt auf die jeweiligen Daten (vereinfacht dargestellt). Wird die Datei H gesucht, so muß sequentiell die gesamte Liste durchschritten werden. Bei langen Listen erfordert dies einen hohen Zeitaufwand und führt somit zu Einschränkungen in der Performance.

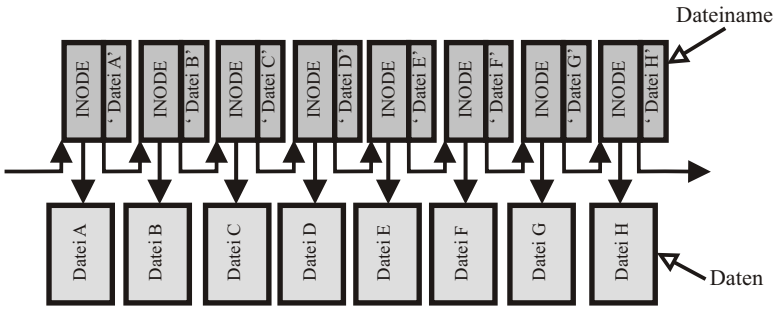

Abbildung 1.10: Beispiel einer verketteten Inode-Tabelle

Durch den Einsatz von Bäumen wird der Zugriff auf die Tabelle wesentlich beschleunigt. In Abbildung 1.11 ist eine als Baum organisierte Inode-Tabelle zu sehen.

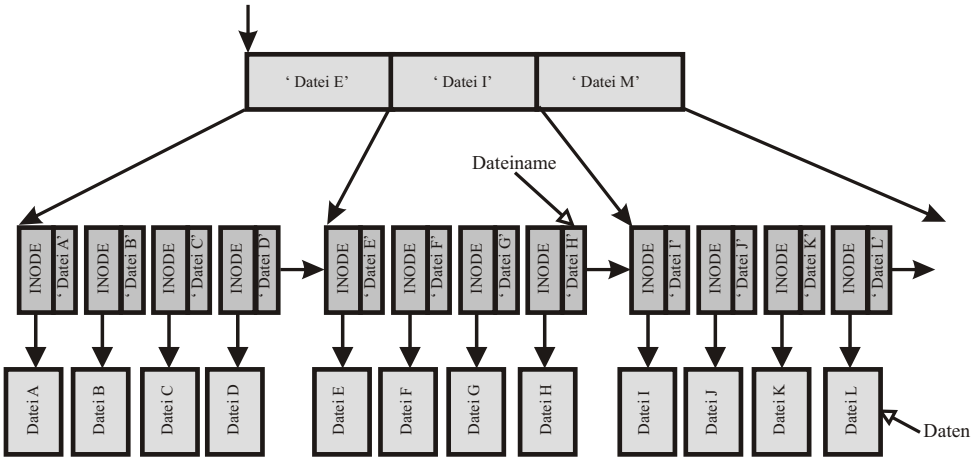

Abbildung 1.11: Beispiel einer als Baum organisierten Inode-Tabelle

Auf der Suche nach Datei H wird lediglich rechts über den Schlüssel *Datei E* in die zweite Teilliste gegangen, wo sich diese am Ende der Teilliste befindet. Dieser Weg wird gewählt, da

der Schlüssel *Datei H* in seinem Wert (alphabetische Sortierung) größer als der Schlüssel von *Datei E* und kleiner als der Schlüssel von *Datei I* ist. Somit liegt der gesuchte Schlüssel in der zweiten Teilliste. Dieser Art der Suche und der Sortierung der einzelnen Baumelemente entsprechen den Charakteristika von balancierten Bäumen[Sed92].

Wird die Komplexität betrachtet, benötigt eine Suche in einem B-Baum durchschnittlich $2 \ln N$ Zugriffe, um den entsprechenden Datensatz zu finden. Die Suche in einer verketteten Liste erfordert ungefähr $N/2$ Zugriffe bis der Datensatz gefunden ist [Sed92]. Ein Rechen-Beispiel, die Suche nach einem Datensatz in einem 1000 Elemente großen Baum würde ca. 14 Zugriffe benötigen, wobei in einer verketteten Liste (gleiche Anzahl von Elementen) ungefähr 500 Zugriffe nötig wären. Dieses Beispiel zeigt deutlich die Vorzüge des Einsatzes von Baumstrukturen in Dateisystemen.

ReiserFS legt die Blockgröße auf einheitlich 4096 Byte fest. Die maximale Dateisystemgröße und die maximale Dateigröße liegen bei 16 TByte. Es ist integraler Bestandteil des Linux-Betriebssystems und somit auf allen Linux-Plattformen verfügbar[Die02].

Der Einsatz von Bäumen als Organisationsstruktur für die verschiedenen Tabellen ist einer der größten Vorteile von ReiserFS. Von Vorteil ist auch die hohe Portabilität und die Journalingfunktionalität. Nachteile sind die fehlende Unterstützung von ACL's und erweiterten Attributen.

SGI-XFS

Die Firma SGI (Silicon Graphics Inc.) entwickelte 1994 ihr eigenes Dateisystem aus Unzufriedenheit mit dem Ur-Unix Dateisystem *UFS*. Diese Unzufriedenheit resultierte daraus, daß UFS aufgrund seiner Datenstrukturen (verkettete Listen etc.) ungeeignet war für große Datenmengen (größer 1 TByte) und damit auch ungeeignet für die enormen Datenmengen grafischer und multimedialer Anwendungen (Arbeitsgebiet von SGI). Der Grundaufbau entspricht dem von ext3 (siehe Seite 18). Es besteht aus Blockgruppen, welche den Superblock, Deskriptoren, Inode-Bitmap, Block-Bitmap und die Inode-Tabelle enthalten. Wie bei ReiserFS werden bei XFS B-Bäume als Organisiationsstruktur eingesetzt. Dieses verschafft ihm einen entsprechenden Performancevorteil, vor allen Dingen bei großen Dateisystemen (größer 1TByte), gegenüber den Dateisystemen mit verketteten Listen.

Zusätzlich verfügt XFS über sogenannte *extends*. Diese *extends* lösen die direkten und mehrfach indirekten Verweise in den Inode-Tabellen vollständig ab. Gerade bei großen Dateien, wird das Durchlaufen der mehrfach verketteten Listen zu einem Zeitproblem. Die Folgen sind Performanceeinbrüche. Die *extends* beschreiben jeweils nur noch eine Blockadresse (A) und eine Blockanzahl (n). Die Blockadresse (A) beschreibt den Anfang des Datenteils der Datei. Die Blockanzahl (n) ist die Anzahl der Blöcke, welche sich an diese erste Adresse anschließen. Somit werden n Blöcke Daten ab der Blockadresse (A) hintereinander sequentiell gelesen. Durch

das sequentielle Lesen wird ein weiterer Performance-Vorteil erreicht.

XFS verfügt über einen Journaling-Mechanismus, ACL-Unterstützung und erweiterte Attribute. Auf seiner Ursprungsplattform IRIX-Unix unterstützt es *Streaming Media*. Der Anwender oder die jeweilige Anwendung ist in der Lage bestimmbare Datenübertragungsraten anzufordern, welche dann durch XFS, so weit dies die momentane Systemauslastung zulässt, eingehalten werden. Als Beispiel einer Anwendung von *Streaming Media* werden Videostreams genannt, welche garantierte Datenraten benötigen, damit ein unterbrechungsfreies Abspielen gewährleistet wird. Datenträgergrößen und Dateigrößen bis zu $8,4 \cdot 10^6$ TByte werden unterstützt.

Der Nachteil von XFS war die schlechten Performance beim Löschen von Dateien. Dieses Problem wurde aber inzwischen von SGI beseitigt.

Vorteile sind die Unterstützung von ACL's, erweiterten Attributen, Journaling, Nutzung von *extends* sowie der breite Einsatz von B-Bäumen in den Organisationsstrukturen. Diese Vorteile qualifizieren XFS zu dem modernsten Dateisystem welches für Linux verfügbar ist.

1.3.3 Quota / Datenträgerkontingente

Die Bereitstellung von Speicherkapazität zählt zu den Haupteigenschaften eines Fileservers. Jeder Benutzer oder jede Gruppe kann seine/ihre Daten in einem für ihn/sie vorbereiteten Bereich (meist Verzeichnis) ablegen. Da aber die Speicherkapazität der Datenträger eine Ressource darstellt (nicht unbegrenzt verfügbar ist), muß diese pro Benutzer oder Gruppe begrenzt werden. Diese Begrenzung wird je nach der Gesamtkapazität des Datenträgers und den Anforderungen/Ansprüchen der Benutzer oder Gruppen festgelegt. Die Begrenzung der Kapazität wird mit einer sogenannten Quota-Software realisiert.

Diese Software prüft ständig ob die administrativ vorgegebene maximale Speicherkapazität überschritten wurde. Ist dies der Fall, so wird der entsprechende Bereich schreibgeschützt und somit für den Benutzer/Gruppe unbrauchbar/nicht mehr beschreibbar (weiterhin lesbar). Unterschieden wird zwischen dem sogenannten *Soft-Quota* und dem *Hard-Quota*. Das *Soft-Quota* stellt eine Speicherkapazitätsgrenze dar, welche unter der vom *Hard-Quota* liegt. Wird die Speicherkapazitätsgrenze des *Soft-Quotas* überschritten, erhält der Benutzer oder die Gruppe eine Warnung. In dieser Warnung[22] wird er aufgefordert Speicherplatz innerhalb einer administrativ festgelegten Zeit freizugeben. Ist diese Zeit abgelaufen, so wird wie oben beschrieben der Schreibzugriff auf die Daten gesperrt. Innerhalb dieser Zeit kann der Benutzer oder die Gruppe aber weiterhin Daten in seinen/ihren Bereich legen. Dies ist aber nur bis zum *Hard-Quota* möglich. Durch das Überschreiten dieser Grenze wird der Bereich (das Verzeichnis) für Schreibvorgänge gesperrt. Nur das Löschen sowie das Lesen von Daten ist in diesem Fall noch

[22] Die Warnung wird dem Nutzer oder der Gruppe auf unterschiedliche Weise mitgeteilt, zum Beispiel durch eine E-Mail, eine Fehlermeldung oder durch Sperrung des Zugangs zum Fileserver.

möglich. Erst nach unterschreiten der *Hard-Quota*-Grenze (durch Löschen von Daten) oder der Einrichtung einer höheren *Hard-Quota*-Grenze durch die System-Administration ist das Ablegen (Schreiben) von neuen Daten wieder möglich.

Im NTFS Dateisystem ist die Quota-Funktionalität integriert und wird dort aber als Datenträgerkontingent bezeichnet. In Abbildung 1.12 ist der Einstellungsdialog (Microsoft Windows Betriebssystem) für Datenträgerkontingente zu sehen. In diesem Fenster werden die jeweiligen Quotas (*Hard- und Soft-Quotas*), sowie die Art der Warnung festgelegt. Das Soft-Quota wird bei NTFS als Warnstufe bezeichnet. In einem weiteren Dialog, welcher über den rechts unten liegenden Knopf *Kontingenteinträge* zu erreichen ist, werden die gewählten Quotas auf auswählbare Benutzer und Gruppen angewendet.

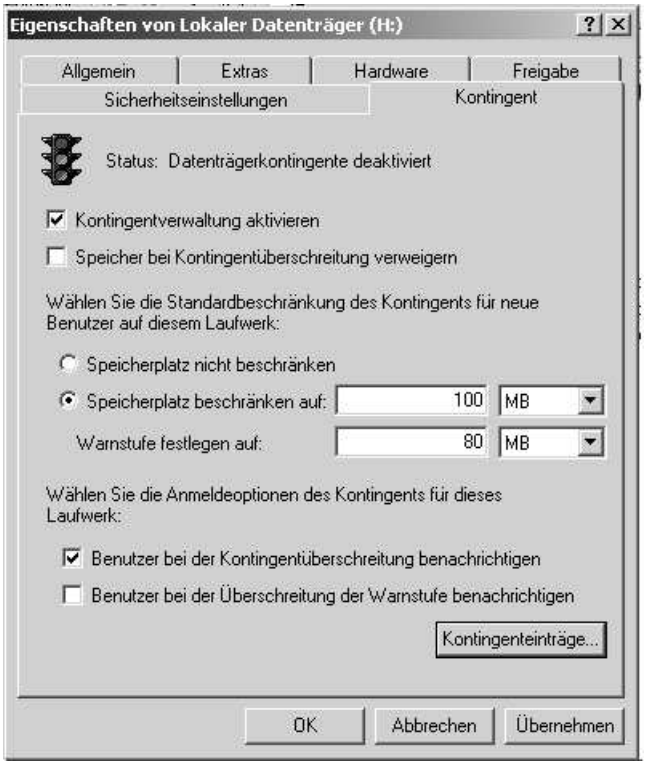

Abbildung 1.12: NTFS-Einstellungsdialog für Datenträgerkontingente

Bei den im Abschnitt 1.3.2 genannten Unix/Linux-Dateisystemen wird die Quota-Funktionalität über eine zusätzliche Software realisiert. Im Linux-Betriebssystem sind die entsprechenden Vorkehrungen für die Quota-Unterstützung schon implementiert. Nur eine Installation der Anwendungen für die Administration der Quotas ist notwendig. Die Implementierung der Quota-Unterstützung im Linuxkernel bringt den Vorteil unabhängig vom verwendeten Dateisystem Datenkontingente (Quotas) zu benutzen und einzurichten.

1.3.4 Physikalische-/ Logische Datenträger

Datenträger stellen in der Realität physisch vorhandene Geräte dar (zum Beispiel Festplatten, Disketten, Datenbänder). Diese werden als Physikalische Datenträger bezeichnet. Alle physikalischen Datenträger besitzen, vorgegeben durch die eingesetzte Technologie, vorbestimmte Speicherkapazitäten. Werden Speicherkapazitäten gefordert, welche über denen der physikalischen Datenträger liegen, ist es notwendig mehrere solcher parallel zu betreiben.

Das parallele Betreiben im Sinne des Verwaltens von mehreren Datenträgern bringt den Nachteil keine Gesamtsicht auf alle Daten zu haben. Somit erhöht sich der Verwaltungsaufwand, da immer verschiedene Datenträger angesprochen werden müssen. Das parallele Betreiben ist heutzutage unumgänglich, da die Speicherkapazitäten der einzelnen physikalischen Datenträger nicht die geforderten Gesamt-Speichermengen erreichen[Ric03]. Durch eine Zusammenfassung der Datenträger oberhalb der Geräteebene (siehe 1.3.1) wird eine Gesamtansicht auf alle Daten verwirklicht.

Diese Funktion realisiert eine Software welche auswählbare physikalische Datenträger zu einem *logischen* Datenträger zusammenfügt. Vertreter solcher Software sind LVM[23] unter Linux, NTFS (das dynamische Datenträger organisieren und anlegen kann) unter Microsoft Windows, Veritas Volume Manager und IBM-LVM (beide unter Unix und Microsoft Windows). Nach dem Anlegen der logischen Datenträger werden diese mit einem Dateisystem formatiert[24] und anschließend wie normale (physisch vorhandene) Datenträger behandelt und benutzt. Zum Erhöhen der Speicherkapazität ist ein späteres Hinzufügen weiterer physischer Datenträger möglich. Das Entfernen eines Datenträgers ist nur mit hohem Aufwand möglich. Daten, welche sich auf dem zu entfernenden physischen Datenträger befinden, müssen erst lokalisiert und dann gelöscht oder auf andere Datenträger[25] gesichert werden.

Eine weitere Organisationsart teilt einen physikalischen Datenträger in mehrere logische Datenträger auf. Dieser Vorgang der Aufteilung wird *Partitionierung* genannt und wird oberhalb der Geräteebene (siehe 1.3.1) des physischen Datenträgers vollzogen. Die einzelnen Partitionen werden dann vom Dateisystem als separate Datenträger erkannt und können zur Nutzung entsprechend formatiert werden. Diese Organisationsart wurde einerseits genutzt, um die in den neunziger Jahren eingesetzten Dateisysteme zu betreiben, welche damals nur bestimmte maximale Datenträgergrößen verwalten konnten. Solche Grenzen waren zum Beispiel 512 MByte, 2 GByte und 8 GByte. Durch die Partitionierung in kleinere Teilstücke wurden die Datenka-

[23] **LVM:** Logical Volume Manager

[24] Die *Formatierung* erstellt auf einem Datenträger, je nach den Eigenschaften des gewählten Dateisystems, eine neue Dateisystemstruktur, dabei wird die eventuell vorhandene alte Dateisystemstruktur und die jeweiligen enthaltenen Daten gelöscht.

[25] diese dürfen nicht Mitglied in der Menge der Datenträger sein, welche den betreffenden Logischen Datenträger stellen, da die Speicherkapazität durch das Entfernen aus dieser Menge geringer wird

pazitäten der über diese Grenzen hinweggehenden Datenträger nutzbar. Anderseits stellt die Partitionierung auch heute noch eine Möglichkeit dar bestimmte Bereiche auf dem Datenträger zu trennen. So ist es möglich eine Partition vor dem Beschreiben komplett zu schützen, während dieses auf einer andere Partition erlaubt ist.

1.4 Fileserversoftware

Die Möglichkeit Informationen und Anwendungen zu teilen, indem Dateien gemeinsam verwendet werden, ist der fundamentale Dienst in einem Rechnernetzwerk. Um diese Möglichkeit zu nutzen bedarf es einer Software, welche diesen Dienst erbringt, der Fileserversoftware. Es werden mehrere Bezeichnungen für den Begriff Fileserversoftware werden verwendet, wie zum Beispiel Netzwerkdateisystem oder das englische Wort*Filesharing* (deutsch: Dateien austauschen).

Historisch bezogen begann die Entwicklung der Fileserversoftware in den 60'er Jahren mit dem Mainframe-Modell. Ein einzelner Großrechner, an dem viele festplattenlose Clients über das Netzwerk angeschlossen waren, verfügte über die gesamten Daten und Anwendungen. Die Clients waren ständig mit dem Großrechner verbunden und teilten sich dessen Ressourcen (Rechenkapazität, Speicher, Daten, Netzwerkbandbreite). Durch den Preisverfall und der enorm angestiegenen Verarbeitungleistung der einzelnen Clients verschwand das Mainframe-Modell mit der Zeit.

Die Clients, welche inzwischen bezüglich ihrer Leistung und Ressourcen (in der Summe) den Mainframes weit überlegen waren, benötigten immer noch eine Möglichkeit ihre Daten auszutauschen. Es entwickelte sich das Client/Server Modell. Der Server, hier der Fileserver, dient als zentrale Ablage und Austauschstelle für Daten und Anwendungen. Diese Fileserver sind auf Eingabe/Ausgabe-Leistung (I/O Leistung) hochoptimierte Rechner, welche auf das Verteilen von großen Datenmengen spezialisiert sind.

In den späten achtziger und neunziger Jahren kristallisierten sich zwei grundlegende Protokoll-Standards für den Austausch von Dateien und Anwendungen über ein Netzwerk heraus. Das von der Firma SUN Microsystems entwickelte NFS (Network File System, deutsch: Netzwerk Datei System) sowie das von der Firma IBM entwickelte und von Microsoft erweiterte SMB-Protokoll (Server Message Block). Beide Firmen bieten auf dem jeweiligen Standard basierende Fileserversoftware an. Inzwischen sind aber auch Produkte anderer Entwickler verfügbar, welche diese Standards (Server- als auch Clientseitig) implementiert haben und somit deren Funktionalität besitzen.

Im Anschluss folgt eine kurze Erläuterung der beiden Standards. Grundlegende Eigenschaften und Besonderheiten werden aufgezeigt. Eine kurze Übersicht über Fileserversoftware, welche

diese Standards vertritt, folgt am Ende der Beschreibung.

1.4.1 Das Unix Netzwerk File System (NFS)

Der Soft- und Hardware-Hersteller SUN Microsystems stellte das verteilte Dateisystem NFS im Jahre 1984 der Öffentlichkeit vor und stellte Lizenzen für das Protokoll zur Verfügung[Hab03]. Durch diese Öffnung zählt es heute zum De facto Standard unter den Unix/Linux-Fileserveranwendungen.

NFS baut auf dem Client/Server Modell auf, wobei diese Rollen nicht fest verteilt sind und ständig wechseln können. NFS-Server und NFS-Client kommunizieren über ein eigenes Protokoll, welches in der Prozess/Anwendungsschicht innerhalb des TCP/IP[Mil99] Schichtenmodells liegt (siehe Abbildung 1.13).

Die Besonderheit liegt in der Nutzung des UDP Transportprotokolls[Mil99]. Das UDP-Protokoll setzt im Gegensatz zum TCP keine festen Verbindungen voraus (Verbindungslose Übertragung). Alle einzelnen Operationen werden einzeln bestätigt und nacheinander ausgeführt. Nachdem ein Client eine Anforderung an einen Server gerichtet hat, muß er zunächst auf die Antwort des Servers warten. Wenn der Server nicht antwortet (z.B. Systemabsturz) oder sich die Antwort verzögert, wiederholt der Client seine Anforderung. Somit hat ein Server-Systemabsturz keine Auswirkungen auf den Client. Hat der Client die Datei erhalten, ist die Client/Server-Kommunikation abgeschlossen. Der Vorteil dieses Arbeitsweise tritt vor allen Dingen im komplexen unzuverlässigen Gebilde Internet zu Tage.

In Abbildung 1.13 sind die drei NFS-Protokollschichten im TCP/IP Schichtenmodell[Mil99] zu sehen. Die erste Schicht des NFS-Protokoll-Stacks, in der Abbildung 1.13 mit NFS-Teil gekennzeichnet, bildet das RPC-Modul. Dieses stellt eine logische Verbindung zwischen Client und Server her. Das ist vergleichbar mit einem Telefonanruf, da der Client eine Nachricht an den Server sendet und dann auf dessen Antwort mit den Ergebnissen der angeforderten Prozedur wartet. Die zweite Schicht ist der XDR-Teil. Diese beschreibt und kodiert die zwischen Client und Server übertragenen Daten. Der NFS-Protokoll bildet die oberste Schicht des NFS-Stacks. Es definiert die Datei- und Verzeichnisstrukturen sowie die Prozeduren für Client und Server.

Die in Abbildung 1.13 enthaltenen RFC (*Request For Comments*) Bezeichnungen beschreiben die jeweiligen Protokollschichten in allen ihren Eigenschaften und Arbeitsweisen. Die RFC's werden von der Network Working Group als Gremium verwaltet und herausgegeben. Die für NFS relevanten RFC's sind die folgenden: RFC 1057 Remote Procedure Call[Mic88], RFC 1014 External Data Representation[Mic87a], RFC 1094 NFS Version 2 [Mic87b], RFC 1813 NFS Version 3 [Mic95] und für die darunter liegenden Schichten RFC 768 für das UDP Protokoll[ISI80] und RFC 791 für das IP-Protokoll[oSC81a].

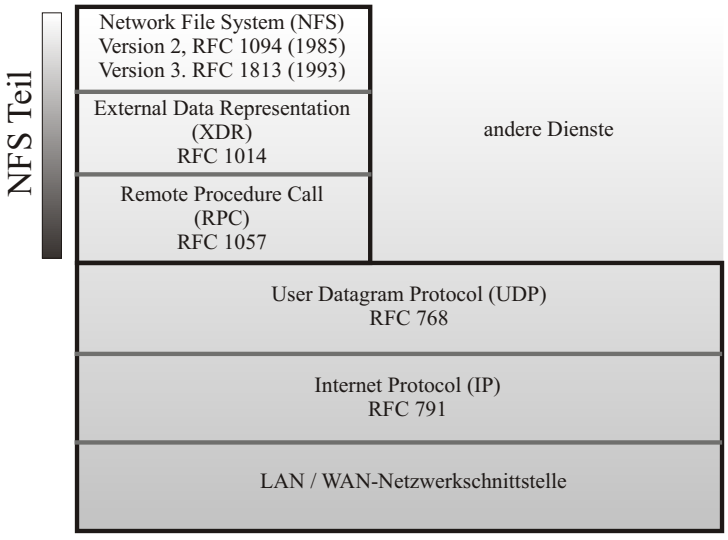

Abbildung 1.13: Die NFS-Protokolle im TCP/IP Schichtenmodell[Mil99]

Die Integration der Daten, welche sich nicht lokal auf dem System befinden, wird mit NFS nahtlos vollzogen. Der Anwender bemerkt keinen Unterschied zwischen den lokal vorhandenen Daten und den Daten, welche über das Netz zur Verfügung gestellt werden[26]. Das Betriebssystem sorgt für die jeweiligen nötigen Systemaufrufe. Es stellt entweder Anfragen an das lokale Dateisystem oder an NFS.

Von NFS existieren heute mehr als 300 Implementierungen verschiedenster Entwickler, was durch die Offenlegung des Protokoll möglich wurde. Die letzte Version von NFS (Version 4) ist im Jahre 2002 erschienen und ist in RFC 3010 [u.a00], [Hab03] beschrieben. NFS 4 baut nicht mehr UDP auf, sondern nutzt TCP als Kommunikationsprotokoll in der Transportschicht. Unterstützung von verschlüsselter Kommunikation und ACL's sind die wichtigsten Neuerungen.

Linux verfügt über eine vollständige NFS Client/Server-Implementierung aller Versionen (einschließlich NFS 4). Die Microsoft Windows Betriebssysteme (NT,2000,XP) verfügen auch über die Fähigkeit NFS Freigaben[27] zu integrieren. Die Microsoft Implementierung stellt allerdings nur den NFS-Client zur Verfügung.

[26] Die durch Zeitverzögerung und Bandbreite der Netzwerkverbindung möglichen Verzögerungen werden hier vernachlässigt.

[27] Eine Freigabe ist der Teil von Dateien und Verzeichnissen, welcher sich auf dem Fileserver befindet und für den Zugriff über das Netzwerk auf dem Client zur Verfügung gestellt wird. Eine Freigabe (*homes*-Freigabe) ist in Abbildung 1.14 zu sehen.

1.4.2 Das Microsoft SMB - Modell

Das Server Message Protokoll wurde 1985 von IBM für kleinere Netzwerke (max 10-20 Rechner) entworfen. Es baut wie das NFS-Protokoll auf dem Client-Server Modell auf, wobei der Server Dateien und Verzeichnisse in Form von Freigaben (siehe Abbildung 1.14) dem Client zur Verfügung stellt. In Abbildung 1.14 ist die SMB-Freigabe *homes* eines Server-Verzeichnisses auf einem Client-Rechner mit Microsoft Windows Betriebssystem dargestellt.

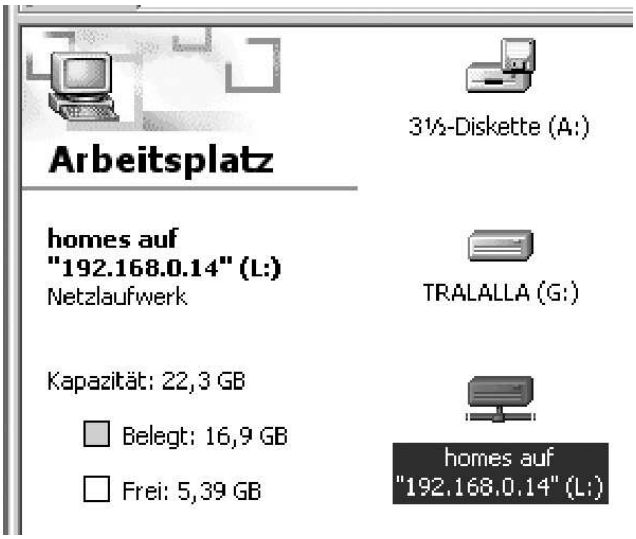

Abbildung 1.14: Eine SMB-Freigabe in Microsoft Windows

Im Jahre 1986 griffen Microsoft, Intel und weitere Firmen das Protokoll auf und entwickelten es ständig weiter. Daher existieren eine große Anzahl verschiedener Varianten des Protokolls, wobei die Microsoft Implementierung die am weitesten verbreitete ist. Microsoft implementierte das Protokoll erstmals mit dem Erscheinen von Windows for Workgroups (1993) in einem seiner Betriebssysteme. Alle weiteren Erläuterungen beziehen sich ausschließlich auf das Microsoft SMB-Protokoll.

SMB setzte als erstes auf die von IBM entwickelte Netzwerkschnittstelle NetBIOS[28] auf [Kau97]. Microsoft erweiterte seine Betriebssysteme, um mittels SMB und NetBIOS lokale Dateien und Verzeichnisse für andere Rechner zugänglich zu machen. Durch die begrenzten Möglichkeiten der NetBios Schnittstelle wurde die Schnittstelle später auf das Standardprotokoll TCP/IP aufgesetzt. Das SMB Protokoll setzte somit auf *NetBIOS over TCP/IP* auf und konnte dessen erweiterte Eigenschaften (größere Anzahl von Rechnern) nutzen. Als Name

[28] Network Basic In and Output System, Erweiterung des Rechner Bios'es um Netzwerkfähigkeiten (nur für kleine Netzwerke geeignet (10-50 Rechner)).

der Microsoft NetBIOS Implementierung wurde NetBEUI (NetBIOS Enhanced User Interface) gewählt.

In Abbildung 1.15 ist das SMB-Protokoll im TCP/IP Schichtenmodell dargestellt. Im Gegensatz zu NFS (Version 2 und 3) setzt das darunterliegende Zwischenprotokoll *NetBIOS over TCP/IP* auf das TCP-Protokoll [Mil99] auf. Bei TCP wird vor der Datenkommunikation ein sichere Verbindung zwischen dem Client und Server aufgebaut. Über diese Verbindung laufen dann alle Anfragen und Daten. Das Zwischenprotokoll NetBIOS over TCP/IP sorgt für die Übersetzung zwischen SMB und dem TCP Protokoll. Der SMB-Teil vereint alle drei (RPC,XDR,NFS) bei NFS bestehenden Schichten. Er ist für die Darstellung von Dateien und Verzeichnissen, der Kodierung der Daten und derren Weiterleitung an die unter ihm liegende Schicht verantwortlich.

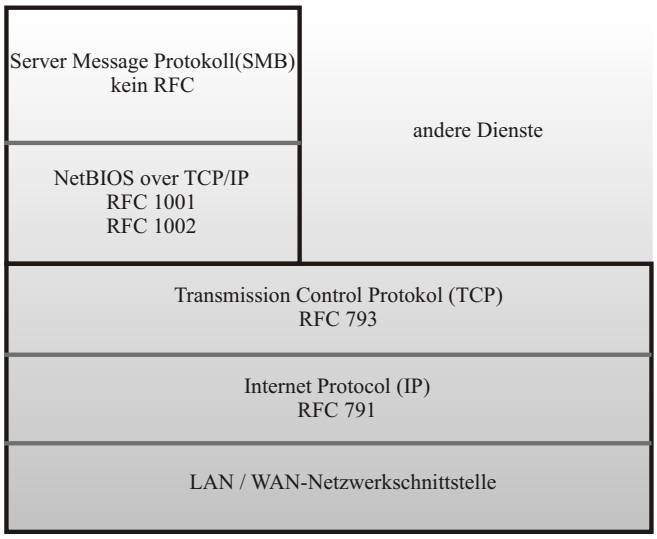

Abbildung 1.15: Das SMB-Protokoll im TCP/IP Schichtenmodell[Mil99]

Das SMB-Protokoll selbst wurde bis heute in keinem RFC-Dokument beschrieben. Die gesamte Microsoft SMB Implementierung ist nicht offen verfügbar und nicht vollständig dokumentiert. Die Funktionsweise des heutigen SMB-Protokolls musste durch Network Reverse Engenieering[29] ermittelt werden, um auch anderen Betriebssystemen eine Anbindung an SMB-Freigaben zu ermöglichen oder um einen eigenständigen SMB-Server zu entwickeln. Die NetBIOS over TCP/IP Implementierung ist in RFC 1001/1002 [wG87a], [wG87b] und die darunter liegenden Protokolle TCP und IP jeweils in RFC 793 [oSC81b] und RFC 791 [oSC81a] beschrieben.

Der heutige Stand der Microsoft SMB-Implementierung setzt komplett auf TCP/IP auf und wurde von der alten NetBIOS-Schnittstelle entkoppelt. Weiterhin wurde das Protokoll in CIFS

[29] Network Reverse Engenierung bezeichnet eine Methode, Verfahren und Arbeitsweisen von Protokollen mittels Abhören der Netzwerk-Kommunikation zu ermitteln

(Common Internet File System), in Anlehnung an NFS, umbenannt. Aktuell ist laut Microsoft die Version 1 des Protokolls.

Name	Hersteller/Entwickler	Kommentar
alle Windows Betriebssysteme Versionen nach Windows for Workgroups	Microsoft	kommerzielle Referenz-Implementation
Samba	Andrew Tridgell u.v.a.	Open Source (freie Software) alle Unix/Linux-Plattformen
PC NetLink	Sun Microsystems	kommerzielle Implementation SUN OS Betriebssystem
TAS (Total NET Adv. Server)	LSI Logic	kommerzielle Implementation verschiedene Unixe

Tabelle 1.4: Verfügbare SMB Server (Auszug)

Um dieses Protokoll auch unter anderen Betriebssystemplattformen verfügbar zu machen, entwickelten verschiedene Hersteller (siehe Tabelle 1.4) , mittels des weiter oben genannten Network Reverse Engeniering, eigene Server und Clients. In Bezug auf das Verteilen von Freigaben ersetzten diese die Microsoft Referenz Implementation vollständig. In Tabelle 1.4 sind eine Reihe von Serveranwendungen und deren Entwickler, welche das SMB-Protokoll implementiert haben, aufgeführt.

1.5 Benutzerverwaltung

Die Benutzerverwaltung spielt eine zentrale Rolle, wenn eine Vielzahl von Benutzern gemeinsame Ressourcen (zum Beispiel Fileserver) nutzen. Durch entsprechende Verfahren muss sichergestellt werden, daß nur autorisierte Benutzer Zugriff auf die für sie bestimmte Daten erhalten. Dazu ist es nötig eine Datenbasis aufzubauen, welche die entsprechenden Benutzer enthält. Mit einer Verwaltungssoftware ist es möglich Benutzer zu erstellen, zu entfernen oder Benutzer-Attribute zu verändern. Alle Attribute, bezogen auf einen Benutzer sowie dessen eindeutiger Name, werden im Gesamten als Benutzerkonto bezeichnet.

In Tabelle 1.5 ist ein Auszug der vorhandenen Benutzer-Attribute des Microsoft Betriebssystems Windows 2000 Proffessional zu sehen. Wurde das letzte in der Tabelle genannte Attribut (*Benutzerkonto ist deaktiviert*) aktiviert, wird der Zugriff des Benutzers auf seine Ressourcen verweigert. Weiterhin ist das Attribut Passwort aufgeführt. Mit diesem Passwort authentifiziert sich der Benutzter gegenüber der Benutzerverwaltung. Bei erfolgreicher Authentifizierung (gültiges Passwort) erhält er Zugriff auf die ihm zugeteilten Ressourcen. Nicht nur

Attribut	Kommentar
Beginn/Ablauf der Zugangsberechtigung	Festlegen einer Zeitspanne
Passwort	Setzen des gültigen Passwortes
Gültigkeitsdauer des Passworts	Passwort muss nach einer frei bestimmbaren Zeit neu festgelegt werden
Benutzerkonto läuft nie ab	Das Benutzerkonto hat eine unbegrenzte Gültigkeit
Benutzerkonto ist deaktiviert	Das Konto existiert, ist aber inaktiv

Tabelle 1.5: Benutzer-Attribute von Microsoft Windows 2000 Prof. (Auszug)

Passwörter können zur sicheren Authentifizierung eines Nutzers herangezogen werden, sondern auch Smartcards (enthält einen geheimen Schlüssel des Benutzers), der Fingerabdruck oder die Augen-Iris des Benutzers. Die Benutzerverwaltung (Benutzerdatenbank) kann sich an verschiedenen Orten im Netzwerk befinden.

Lokale Speicherung

In Abbildung 1.16 ist ein Szenario mit einer lokalen Benutzerdatenbank dargestellt. Der mit einer gestrichelten Linie umfaßte Bereich soll die lokale Speicherung verdeutlichen. Der Benutzer arbeitet am Client-Rechner und authentifiziert sich, über die auf dem gleichen Rechner (Client) abgelegte Benutzerdatenbank, gegenüber den vom Fileserver angebotenen Ressourcen (Freigaben).

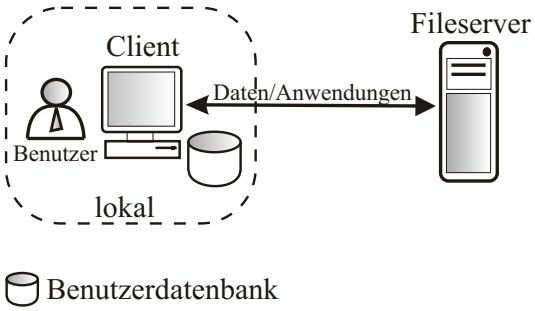

Abbildung 1.16: Lokale Speicherung der Benutzerdatenbank

Diese Betriebsart hat den gravierenden Nachteil, daß alle Client-Rechner im Netzwerk mit aktuellen Kopien der Benutzerdatenbank (welche sich jeweils auf jedem Client-Rechner befindet), im Falle einer Änderung der Benutzerdaten, versorgt werden müssen. Diese Vorgehensweise ist nötig, um die Nutzung der Ressourcen jedem Benutzer von allen Client-Rechnern aus zu

ermöglichen. Diese Methode ist nur für sehr kleine Netzwerke (1-5 Client-Rechner) praktikabel.

Speicherung auf dem Server

Die zentrale Speicherung der Benutzerdatenbank verfügt über den großen Vorteil, gegenüber der lokalen Speicherung, daß nur noch eine Benutzerdatenbank pro Fileserver verwaltet werden muss. In Abbildung 1.17 befindet sich die Benutzerdatenbank direkt (lokal) auf dem Fileserver.

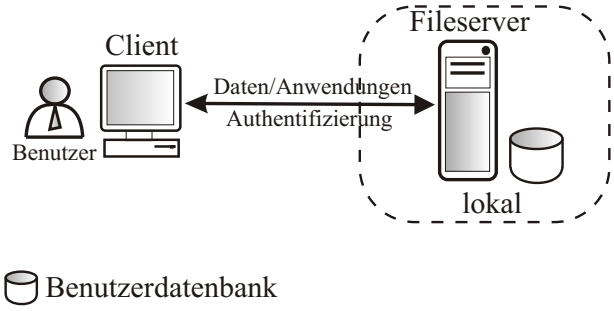

Abbildung 1.17: Speicherung der Benutzerdatenbank auf dem Fileserver

Der Benutzer authentifiziert sich (über seinen Client-Rechner) direkt am Fileserver, um an seine Daten, welche sich auf dem Fileserver befinden, zu gelangen. Somit findet die Authentifizierung über das Netzwerk statt. Dieses Verfahren wird in mittleren bis großen Netzwerken eingesetzt (5-100 Clients). Befinden sich mehrere Fileserver oder andere Dienste-Server im Netzwerk, (zum Beispiel E-Mail-Server) entsteht der Nachteil, daß jeder dieser einzelnen Server eine eigene Benutzerdatenbank enthält. Ein Ausweg wäre hier die regelmäßige Synchronisation der Benutzerdatenbanken.

Speicherung auf einem Authentifizierungsserver

In sehr großen Netzwerken (100 bis einige tausend Client-Rechner) ist es nötig die Benutzerdatenbanken auf Rechner zu verlagern, welche ausschließlich für die Authentifizierung zuständig sind. In Abbildung 1.18 ist ein derartiges Szenario dargestellt. Der Benutzer authentifiziert sich (über seinen Client-Rechner) gegenüber dem Authentifizierungsserver (enthält die zentrale Benutzerdatenbank). Dieser signalisiert dem Fileserver ob der Benutzer erfolgreich authentifiziert wurde. Ist dies der Fall, gibt der Fileserver die zugehörigen Ressourcen für den Benutzer frei.

Benutzerdatenbanken können sowohl Teil des Betriebssystems sein als auch separate Anwendungen. Reine Benutzerdatenbanken gehören aber inzwischen fast ausnahmslos der Vergangenheit an. Die neuen Verwaltungsdatenbanken (*Directories* genannt) speichern nicht nur die

nötigsten Daten der Benutzer, sondern besitzen eine beliebig erweiterbare Anzahl von Attributen (Benutzer-Attribute). Diese Attribute können zum Beispiel sein Telefonnummer, Zimmer, Adresse, Abteilung, Stelle usw. . Weiterhin dienen sie zusätzlich der Inventarisierung unterschiedlichster Betriebsmittel (zum Beispiel Rechner, Möbel, Büchern usw.). Damit steht nur noch eine einheitliche zentrale Datenbasis zur Verfügung, deren Verwaltung zentral organisiert wird.

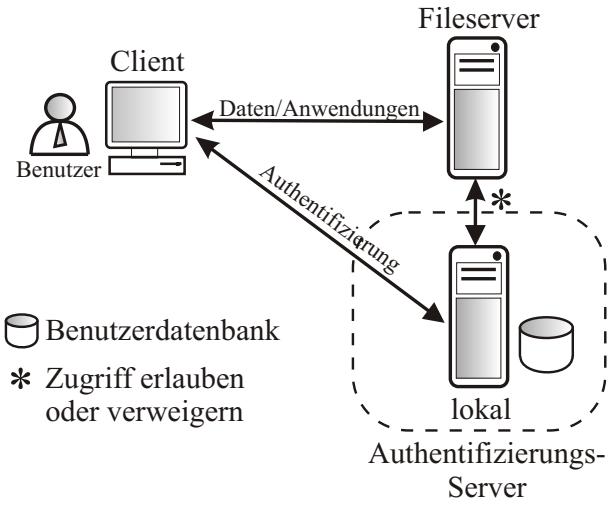

Abbildung 1.18: Benutzerdatenbank auf einem Authentifizierungsserver

Die Directories ermöglichen allen Anwendungen, welche eine Benutzerauthentifikation oder Benutzerinformationen benötigen, nur noch auf eine Datenbasis zuzugreifen. Solche Anwendungen können zum Beispiel E-Mail Server oder Telefonbücher sein. Die Folge ist die Ermöglichung des sogenannten *Single Sign On*. Der Benutzer authentifiziert sich nur einmal und bekommt Zugriff auf alle Ressourcen, welche für ihn zur Nutzung freigegeben wurden. Eine mehrmalige Authentifikation bei jeder Anwendung ist nicht mehr nötig. Im Anschluss folgt eine kurze Beschreibung der wichtigsten Directory Anwendungen.

1.5.1 NIS

NIS (Network Informaion Service)[30] der Firma SUN Microsystems ist einer der ältesten Directories. Es stellt lediglich Benutzerinformationen wie Name, Passwort, Gruppenzugehörigkeit und Gruppeninformationen zur Verfügung. Implementiert wurde es auf allen Unix/Linux-Plattformen.

[30] NIS trug den Namen Yellow Pages, welcher aber ein eingetragenen Warenzeichen der British Telecom war und somit geändert werden musste

1.5.2 Active Directory

Mit dem Windows 2000 Server Betriebssystem stellte Microsoft 1999 diesen neuen Directory Dienst vor, welcher an den offenen X.500 Directory Standard[Ban01] angelehnt ist. Er beinhaltet nicht nur die Benutzerverwaltung, mit einer großen Anzahl von Attributen, sondern auch die komplette Verwaltung aller angeschlossenen Rechner (Clients) und deren Ressourcen. Ein Beispiel dafür ist die Möglichkeit die Quotas (siehe 1.3.3) jedes einzelnen Clients einzustellen oder alle Clients mit der Zeit des Directory-Servers zu synchronisieren. Active Directory ist als Dienst (Server) nur auf Microsoft Windows 2000 Server und Windows 2003 Server verfügbar. Clients mit voller Unterstützung sind nur die Microsoft Betriebssysteme Windows 2000 Professionell und Windows XP Professionell (Windows NT 4.0 mit Einschränkungen) . Die offizielle Unterstützung, durch Microsoft, zu anderen Betriebssystemen ist nicht vorgesehen. Inzwischen existieren aber einige das Active Directory abfragende Anwendungen außerhalb der Microsoft Windows Betriebssysteme. Beispiel einer solchen Anwendung ist der freie Fileserver Samba in der Version 3 (siehe Tabelle 1.4), welcher Daten aus dem Active Directory zur Authentifikation von Benutzern verwendet. Durch die Orientierung des Active Directorys am X.500 Standard sind Verbindungen über das LDAP-Protokoll möglich.

1.5.3 OpenLDAP

Der OpenLDAP-Server[Ban01] ist eine *Open Source*[31]-Entwicklung eines Directory Servers. Er basiert auf einer abgewandelten und vereinfachten Implementierung des X.500 Directory Standards [Ban01] der OSI-Organisation[32]. OpenLDAP ist auf allen Unix/Linux-Plattformen und der Microsoft Windows Plattform verfügbar. Die Abkürzung LDAP steht für Light Weight Access Protocol und beschreibt das Kommunikationsprotokoll, welches bei der Kommunikation zwischen LDAP-Server und dem LDAP-Client[33] zum Einsatz kommt.

1.5.4 NDS

NDS steht für Novell Directory Service und wurde von der gleichnamigen Firma entwickelt, um alle Ressourcen eines Netzwerkes zu verwalten. Die Benutzer-, Gruppen- und Rechnerverwaltung ist eines der Hauptarbeitsgebiete dieses Directories. Eine beliebige Erweiterung der Attribute sowie Verbindungen über das LDAP-Protokoll sind möglich. NDS-Server sind auf

[31] Open Source: der Quellcode der Anwendung ist frei verfügbar, Funktionsweise ist einsehbar, die Anwendung kann modifiziert, erweitert, weiterentwickelt werden

[32] OSI : Open System Interconnection , Standardisierungs Gremium für Netzwerk Standards

[33] LDAP-Client ist eine Anwendung, welche Informationen vom Directory anfordert und/oder Informationen in das Directory schreibt

allen Betriebssystemplattformen verfügbar, einschließlich Novell's eigenen Netzwerkbetriebssystem *Netware*.

1.5.5 Sun ONE Directory Server

Dieser Directory Server (ehemals iPlantet Directory Server, jetzt Sun Microsystems) stellt eine weitere LDAP-konforme Implementierung dar. Das LDAP-Protokoll wird auch hier als Kommunikationsprotokoll eingesetzt. Der Funktionsumfang entspricht dem des OpenLDAP Servers. Der Sun ONE Directory Server ist nur auf dem SUN OS Betriebssystem von SUN Microsystems verfügbar.

1.6 Datensicherheit

Der Verlust von Daten kann niemals ausgeschlossen werden. Hardwaredefekte, Viren, höhere Gewalt wie Feuer, Wasser oder Naturkatastrophen und letztlich menschliches Versagen können dazu führen. Somit wird versucht mit entsprechenden Mechanismen und Technologien die Gefahr von Datenverlust zu minimieren. Das Sichern, der auf einem Fileserver vorhandenen Daten, ist eine weitere wichtige Aufgabe, die regelmäßig vorgenommen werden muss. Ebenfalls sind Mechanismen zu integrieren, die einen hohen Schutz vor dem physischen Ausfall und dem damit verbundenen Datenverlust auf dem eingesetzten Datenträger gewährleisten. Beide Arten der Sicherstellung von Daten werden in diesem Abschnitt behandelt.

1.6.1 RAID

RAID ist eine Datensicherungsmaßnahme, welche für Festplatten als Datenträger entwickelt wurde. Der Begriff RAID ist eine Abkürzung welche ursprünglich für *Redundant Array of Inexpensive Discs* stand. Er entstand in einer Zeit, in welcher große Festplatten noch sehr teuer und unzuverlässig waren[Erk03]. Mehrere kleine preiswerte Festplatten wurden zu einem großen sogenannten virtuellen Datenträger zusammengeschlossen. Heute steht RAID für *Redundant Array of Independant Discs* und bezeichnet einen Zusammenschluss einer bestimmten Anzahl unabhängiger Festplatten.

RAID verfolgt neben dem Ziel die Ausfallsicherheit (Redundanz, Schutz vor Datenverlust) zu erhöhen, die Erhöhung der I/O Performance. Wobei letzteres als Striping bezeichnet wird. Redundanz speichert zusätzlich Informationen, so daß der normale Betrieb nach Austausch eines defekten Datenträgers mit einem neuen fehlerfreien Datenträger fortgesetzt werden kann. Striping teilt die Daten auf mehrere Festplatten auf, und verteilt somit auch die I/O Last. Die Per-

formance sowie die Ausfallsicherheit einer einzelnen Festplatte kann durch RAID nicht erhöht werden, nur mit einer Kombination mehrerer Festplatten kann dies erreicht werden[Erk03].

Der sogenannte RAID-Controller[34] bündelt die einzelnen Festplatten zu einer virtuellen Festplatte zusammen. Das Dateisystem sieht nur eine gesamte Festplatte. Die rein physikalisch vorhandenen Festplatten bleiben vor ihm verborgen. Ein RAID-Controller verteilt die Daten, welche vom Rechner kommen, auf verschiedene Art und Weise auf die einzelnen physikalischen Festplatten . Diese verschiedenen Verfahren werden als RAID-Level bezeichnet. Weiter unten in diesem Abschnitt wird auf die RAID-Level näher eingegangen.

In allen RAID-Levels (außer RAID 0) wird Redundanz gewährleistet. Bei Verlust einer Festplatte kann entweder auf die verbleibende Festplatten ausgewichen werden (RAID 1) oder aus den restlichen Festplatten der Inhalt der ausgefallenen Festplatte rekonstruiert werden (RAID 4 und RAID 5). Im letzteren Fall muss die defekte Festplatte manuell ausgetauscht werden, um den weiteren Betrieb zu gewährleisten und die Rekonstruktion zu ermöglichen. Dieser notwendige sofortige Austausch kann durch den Einsatz einer sogenannten Hot Spare Disk zeitlich verschoben werden. Bei Ausfall einer Festplatte führt der RAID-Controller automatisch die Rekonstruktion auf der Hot Spare Disk durch. Damit ersetzt die Hot Spare Disk die defekte Festplatte vollständig. Die Hot Spare Disk ist eine Ersatzfestplatte, welche keine Daten enthält und immer betriebsbereit ist. Die defekte Festplatte wird ausgetauscht und durch eine neue ersetzt. Diese übernimmt dann die Rolle der Hot Spare Disk.

RAID 0:Blockweises Striping

Der RAID-Level 0 verteilt die einzelnen Daten hintereinander blockweise auf die physikalischen Festplatten. Sind zum Beispiel vier physikalische Festplatten im RAID-System vorhanden und die zu speichernden Datenblöcke tragen die Bezeichnungen A bis Z, so wird Block A auf Festplatte Eins und Block B auf Festplatte Zwei geschrieben. Steht Block E an, so wird dieser wieder auf Festplatte Eins geschrieben. Das Lesen der Blöcke erfolgt in der gleichen Weise wie das Schreiben. Durch das parallele Schreiben und Lesen kommt es zu einer Performanceverbesserung. Die Ausfallwahrscheinlichkeit steigt aber um den Faktor der Anzahl angeschlossener physikalischer Festplatten. Fällt auch nur eine der physikalischen Festplatten aus, so sind deren Blöcke verloren und somit auch die Datenintegrität der virtuellen Festplatte. Der Verlust aller Daten ist die Folge. Das **R** für Redundant trägt RAID 0 zu Unrecht. Die 0 im Namen steht daher für NULL Redundanz.

[34] Ein RAID Controller kann sowohl über eine Hardware-Logik, als auch über Software (Zentrale Recheneinheit übernimmt die Verwaltung und Organisation der RAID-Funktionen) realisiert sein.

RAID 1:Blockweises Mirroring

Bei RAID 1 spielt die Datensicherheit die Hauptrolle. Wie beim oben (RAID 0) genannten Beispiel sind wieder die vier physikalischen Festplatten und die Datenblöcke vorhanden. Der einzelne ankommende Datenblock wird aber bei RAID 1 auf jede einzelne physikalische Festplatte kopiert. Block A wird somit auf Festplatte Eins bis Festplatte Vier kopiert. Mit Block B wird gleich verfahren. Dieser Vorgang wird Mirroring (deutsch: Spiegeln) genannt, da immer die exakte Kopie jedes Blockes auf jeder Festplatte abgelegt wird. Performancesteigerungen sind nur beim Lesen zu beobachten (paralleles Lesen), wobei diese nicht die Werte von RAID 0 erreichen. Beim Schreiben ist die Performance gleich der einer einzelnen physikalischen Festplatte. Fällt eine der physikalischen Festplatten aus, so können die anderen normal weiterarbeiten. Zur Wiederherstellung der vorherigen Ausfallsicherheit (vier Festplatten) müssen lediglich die Daten der laufenden Festplatten auf die neue ausgetauschte Festplatte kopiert werden. Der Faktor der Ausfallsicherheit steigt mit der Anzahl der eingesetzten physischen Festplatten. Da die Gesamt-Speicherkapazität jeweils nur der Speicherkapazität einer einzelnen physischen Festplatte entspricht, steigen mit der Ausfallsicherheit auch die Kosten.

RAID 0+1/RAID 10: Striping und Mirroring kombiniert

RAID 0 steigert die Performance und RAID 1 steigert die Ausfallsicherheit. Aus der Kombination von beiden RAID Leveln wurde RAID 0+1 und RAID 10 entwickelt. Beide bilden jeweils eine zweistufige Virtualisierungshierarchie.

Die Funktionsweise des RAID 0+1 Standards wird folgendermaßen realisiert. Zuerst werden acht (Beispiel) physikalischen Datenträger (erste Hierarchiestufe) mittels RAID 0 zu zwei virtuellen Festplatten gestript , was den geforderten Performancevorteil schafft. Diese zwei virtuellen Festplatten werden dann mittels RAID 1 nochmals zu einem, für das Dateisystem sichtbaren, virtuellen Datenträger gespiegelt, was die benötigte Redundanz verschafft (zweite Hierarchiestufe). Der Nachteil ist, fällt ein physikalischer Datenträger aus, so ist eine der zwei virtuellen Festplatten (zweite Hierarchiestufe) komplett verloren (keine Ausfallsicherheit, gleich RAID 0).

Der RAID 10 Standard führt in der ersten Hierarchiestufe ein Mirroring der acht physikalischen Festplatten (Beispiel) durch (RAID 1). Diese vier virtuellen Festplatten werden dann mittels RAID 0 zu einem virtuellen, für das Dateisystem sichtbaren, Datenträger zusammen gestript (zweite Hierarchiestufe). Das Performanceverhalten ist mit dem von RAID 0+1 identisch. Der große Vorteil von RAID 10 liegt in der erhöhten Ausfallsicherheit gegenüber RAID 0+1. Fällt eine physikalische Festplatte aus, so kann mittels des auf der ersten Hierarchiestufe eingerichteten RAID 1 sofort die zweite gespiegelte physikalische Festplatte eingesetzt werden. Fällt eine weitere physikalische Festplatte aus, so verkraftet RAID 10 auch diesen Ausfall (Ausnahme:

es fällt die gespiegelte physikalische Festplatte der vorher defekt gegangenen physikalischen Festplatte aus).

RAID 4 und RAID 5: Parity

RAID 10 und RAID 1 besitzen einen gemeinsamen Nachteil. Durch das Mirroring werden alle Daten mindestens doppelt[35] auf die einzelnen physischen Datenträger geschrieben. Der Bedarf an Speicherkapazität ist dabei auch mindestens doppelt so hoch, wie die wirklich nutzbare Speicherkapazität.

RAID 4 besitzt zusätzlich eine sogenannte Paritätsfestplatte zu den vorhandenen physikalischen Festplatten. Ist zum Beispiel die fünfte Festplatte die Paritätsfestplatte, so werden auf den restlichen vier Festplatten die Daten (siehe RAID 0) mittels Striping parallel verteilt. Wie in dem schon vorher genannten Beispiel, werden die Datenblöcke A bis D wieder auf die vier Festplatten hintereinander verteilt. Ist der Datenblock D auf die vierten Festplatte geschrieben, errechnet der RAID-Controller aus den jeweils zuletzt geschriebenen Datenblöcken (im Beispiel: A (Festplatte 1), B (Festplatte 2), C (Festplatte 3), D (Festplatte 4)) mittels XOR-Verknüpfung einen Paritätsblock. Dieser Paritätsblock wird auf der Paritätsfestplatte (Festplatte Fünf) gespeichert.

Fällt eine der vier Festplatten aus, so läßt sich mittels der drei anderen Festplatten und dem auf der Paritätsfestplatte gespeicherten Paritätsblock der jeweilige verlorene Block mittels einer XOR Verknüpfung wieder rekonstruieren. Nachteilig wirkt sich die sogenannte Write Penalty auf die Performance von RAID 4 aus. Ändert sich nur ein Block auf einem der vier physikalischen Datenträger, so muß immer der jeweilige Paritätsblock neu berechnet und geschrieben werden. Dieser Vorgang führt zu Performanceeinbußen.

Nachteilig für RAID 4 ist die Paritätsfestplatte. Werden alle Zugriffe auf die vier physikalischen Festplatten verteilt, so bedeutet jeder Schreib-Zugriff auf eine der vier physikalischen Festplatten einen Zugriff auf die Paritätsfestplatte (Neuberechnung des Paritätsblockes). Diese wird somit stärker mechanisch beansprucht und besitzt dadurch eine höhere Ausfallwahrscheinlichkeit. Fällt diese aus, so müßen alle Paritätsblöcke auf der ausgetauschten Paritätsfestplatte zeitaufwendig neu berechnetet werden. Diesen Nachteil beseitigt RAID 5. Es verteilt die berechneten Paritätsblöcke über alle fünf physikalischen Datenträger gleichmäßig. Somit werden alle Festplatten gleichmäßig mit I/O Operationen belastet. Da RAID 5 auch Paritätsblöcke berechnet, tritt auch dort Write Penalty auf.

RAID 4 und RAID 5 benötigen insgesamt fünf Datenträger. Die nutzbare Speicherkapazität ist nur um ein Fünftel geringer wie die Gesamtspeicherkapazität aller beteiligten physikalischen Festplatten. Beide RAID-Level (RAID 4 und 5) können nur den Ausfall einer physikalischen

[35] oder je nach Anzahl der Festplatten, welche an dem Mirroring beteiligt sind

Festplatte kompensieren. Kommt es zum Ausfall einer zweiten Festplatte gehen Daten verloren. Der Ausfall einer einzigen physikalischen Festplatte bedingt die Berechnung der wiederherzustellenden Daten aus den Paritätsdaten und den Daten der verbleibenden Festplatten. Dieser Vorgang ist erheblich komplexer als das einfache Kopieren der Daten bei RAID 1 und RAID 10/0+1.

RAID 2 und RAID 3

RAID 2 und RAID 3 werden in neuen Systemen nicht mehr eingesetzt. Beide Standards stammen aus der Entwicklungszeit der RAID-Standards[Erk03].

RAID 2 nutzte den Hamming-Code, um Bitfehler zu erkennen und zu korrigieren. Mangelnde mechanische Zuverlässigkeit der Festplatten führte zu diesem Standard. Durch die fortgeschrittene Festplattentechnologie tritt der Mangel inzwischen nicht mehr auf. Somit verlor RAID 2 seine Existenzberechtigung.

RAID 3 arbeitete ähnlich wie RAID 4 und RAID 5 mit Paritätsdaten. Es verteilte die Daten eines Blockes auf alle Festplatten gleichzeitig. Zusätzlich wurde die Rotation der einzelnen Festplatten synchronisiert, so daß die Daten eines Blockes gleichzeitig geschrieben und gelesen wurden. Durch diese Arbeitsweise wurde es für Data Mining und in der Videobearbeitung angewendet. Die aufwendige Rotations-Synchronisation der Festplatten und die neuen RAID Level 4 und 5 ließen RAID 3 an Bedeutung verlieren.

Vergleich der RAID Level

In Tabelle 1.6 sind die Hauptmerkmale der verschiedenen RAID-Level für einen direkten Vergleich aufgeführt. Die genannten Größen stellen eine relative Gewichtung der RAID-Level untereinander dar. Da RAID 2 und RAID 3 in der Praxis faktisch nicht mehr eingesetzt werden, sind sie nicht in der Tabelle aufgeführt.

RAID-Level	Ausfall-sicherheit	Lese-Performance	Schreib-Performance	Platzverbrauch
RAID 0	keine	gut	sehr gut	minimal
RAID 1	hoch	schlecht	schlecht	hoch
RAID 10	sehr hoch	sehr gut	gut	hoch
RAID 0+1	hoch	sehr gut	gut	hoch
RAID 4	hoch	gut	schlecht	gering
RAID 5	hoch	gut	schlecht	gering

Tabelle 1.6: RAID Level im Vergleich[Erk03]

RAID 0 bietet eine Verbesserung der Performance verbunden mit einer nicht vorhandenen Ausfallsicherheit. RAID 1 besitzt eine hohe Ausfallsicherheit, aber der Platzverbrauch ist sehr hoch. Bei RAID 10 ist die Ausfallsicherheit und die Performance sehr gut, wobei der Platzverbrauch hoch ist. Die beste Lösung für einen Fileserver stellen die RAID Level 4 und 5 in Zusammenarbeit mit Hot Spare Disks dar. Die effektive Speicherkapazität wird durch die Paritätsverfahren nicht zu stark reduziert. Die Ausfallsicherheit ist bei RAID 4 und besonders bei RAID 5 sehr hoch. Die Leseperformance ist gut, wobei der einzige Nachteil die schlechte Schreibperformance ist (Write Penalty).

1.6.2 Backup/Archivierung

Ein Backup oder eine Archivierung stellen eine Sicherheitskopie von Datensätzen für den Fall eines unerwarteten Datenverlustes dar. Tritt dieser Fall ein, wird die Sicherheitskopie genutzt, um die verlorenen Daten wieder zu ersetzen[Ste03]. Der deutsche Gesetzgeber schreibt eine Frist von 10 Jahren für die Datenaufbewahrung in Firmen und im öffentlichen Dienst vor. Bevor Daten gesichert/archiviert werden, müssen einige Überlegungen bezüglich der Organisation angestellt werden. Die wichtigsten Punkte sind:

- Art der Sicherungsmedien
- Größe der Medien
- Backupstrategie
- Ursprung Daten (Lokal,Netzwerk)

Die zur Sicherung bestimmten Medien müssen für die dauerhafte Lagerung geeignet sein. Sie müssen in Größe und Datentransferraten dem eingesetzten Umfeld entsprechen. Ihre Technologie muß über einen möglichst langen Zeitraum verfügbar sein und unterstützt (Service,Wartung) werden[36], eine hohe Fehlertoleranz besitzen sowie Fehlerkorrektur-Maßnahmen anbieten. Bei den Sicherungsmedien wird oft von Sekundärspeichern gesprochen. Primärspeicher[37] sind somit alle Datenträger mit welchen unmittelbar direkt gearbeitet wird (zum Beispiel Festplatten).

Drei sogenannte Backupstrategien stehen zur Auswahl. Das Voll-, das inkrementelle- sowie das differentielle-Backup. Das erstere speichert bei jedem Backupvorgang die gesamten Daten. Tritt der Fall eines Datenverlustes ein, werden einfach die Daten des benötigten Voll-Backups wieder

[36] Dies ist notwendig um auch später die Medien noch lesen zu können (Zugriff auf die archivierten Daten)

[37] Primärspeicher sind (technologisch bedingt) Datenspeicher mit wesentlich geringeren Zugriffszeiten (unter 10 ms) und hohen Transferraten (bis 180 MByte/s) im Vergleich zu den langsamen (technologisch bedingt) Sekundärspeichern mit niedrigen Transferraten (bis 11 MByte/s) und hohen Zugriffszeiten (im Minutenbereich)

zurück kopiert. Der Nachteil ist die benötigte Speicherkapazität. Bei jedem Voll-Backup wird ein Anteil an Daten gesichert, welcher schon mit dem vorhergehenden Backup gesichert wurde und sich in der Zeitspanne seit diesem nicht verändert hat.

Das inkrementelle Backup benötigt weniger Kapazität auf den einzelnen Sicherungsmedien, im Vergleich zum Voll-Backup. In Abbildung 1.19 ist die Arbeitsweise zu sehen. Es werden nur jeweils die Daten auf ein Medium gesichert, welche seit dem letzten Backup neu hinzugekommen sind oder sich verändert haben (Teil-Backup). In frei wählbaren Abständen[38] wird aus dem vorhergehenden Voll-Backup[39] und den einzelnen Teil-Backups ein neues Voll-Backup erstellt. Die Medien der Teil-Backups (das gerade erstellte Voll-Backup Medium ausgenommen) können gelöscht werden und stehen dem Backup-Kreislauf wieder vollständig zur Verfügung. Anschließend wird in der gleiche Weise fortgefahren. Ist die Speicherkapazität eines Mediums ausgeschöpft, wird es zur Einlagerung aufbewahrt und durch ein leeres Medium ersetzt. Werden die Daten benötigt, stellt das letzte Vollbackup und alle bis zum gewählten Zeitpunkt gespeicherten Teilbackups zusammen den benötigten Datenzustand wieder her.

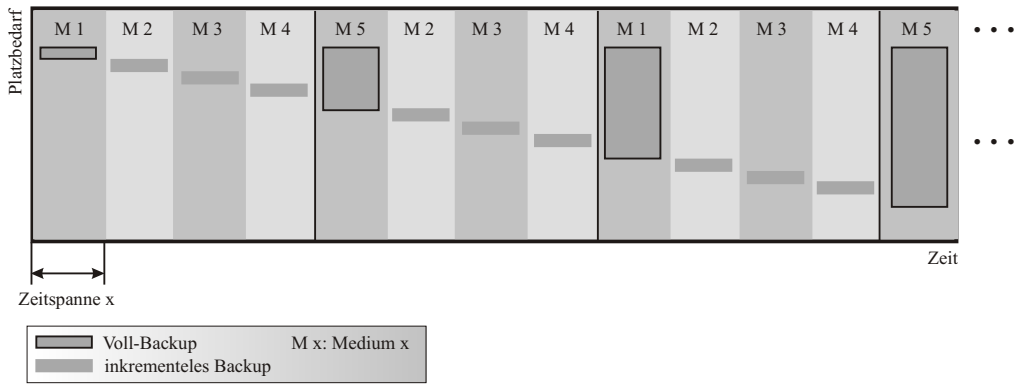

Abbildung 1.19: Inkrementelles Backup

Das differentielle Backup sichert bei jedem Teil-Backup, im Unterschied zum inkrementellen Backup, alle Daten die sich seit dem letzten Voll-Backup verändert haben oder hinzugekommen sind. In Abbildung 1.20 ist die Arbeitsweise dargestellt. Dabei werden bei jedem differenziellen Teil-Backup die gleichen Daten des vorhergehenden Teil-Backups nochmals gesichert. Dies bedeutet zusätzlichen Platzverbrauch auf den Speichermedien. In gleichen Abständen wird eines der Teilbackups zum Vollbackup erklärt und die anderen Teilbackups werden komplett gelöscht. Mit den leeren Medien werden dann wieder, auf das letzte Voll-Backup aufbauende, inkrementelle Teil-Backups erstellt. Der Vorteil des inkrementellen Backups liegt in der schnellen

[38] Die Wahl des Abstands bestimmt über die Anzahl der benötigten Medien.
[39] existiert dieses noch nicht, wird durch Zusammenfügen der Teil-Backups ein erstes Vollbackup erstellt

Verfügbarkeit des Backups im Fall eines Datenverlustes. Um bei Bedarf an die Daten zu gelangen, ist nur die Zusammenführung des letzten Voll-Backups und des letzten differentiellen Teil-Backups nötig.

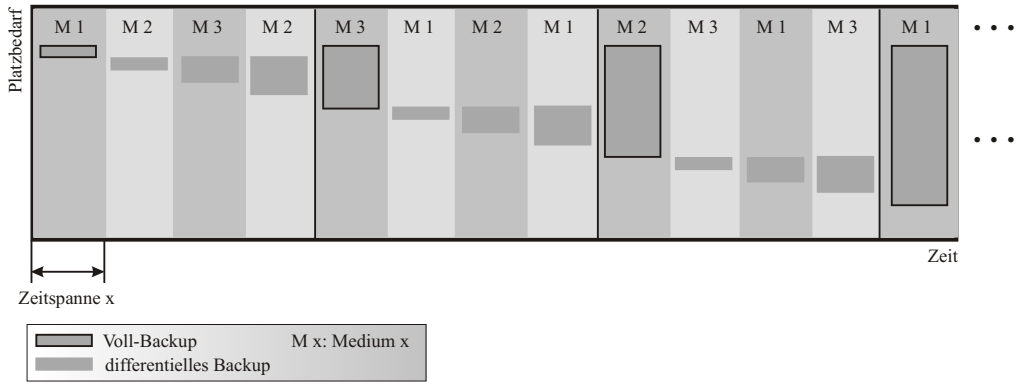

Abbildung 1.20: Differentielles Backup

Woher kommen die Daten für das Backup? Daten können sowohl lokal (auf dem jeweiligen Rechner selbst) oder auch über das Netzwerk[40] gesichert werden. Bei einem lokalen Backup wählt der Benutzer die zu sichernden Daten aus und teilt die Auswahl dem Backup-Programm mit. Dieses sichert dann in bestimmbaren Zeitabständen und unter Verwendung einer der Backupstrategien die Daten.

Das Backup über das Netzwerk arbeitet mit einem Netzwerk-Backup-Server[41] und mit Backup-Clients. Diese Backup-Clients sind auf den Rechnern installiert, auf welchen sich die zu sichernden Daten befinden. Der Backup-Server besteht aus mehreren Komponenten dem Job Scheduler, dem Error-Handler, der Metadaten-Datenbank und dem Media-Manager. Der Job Scheduler führt die Datensicherung zu einem vorgegebenen Zeitpunkt automatisch aus. Diese Einstellung übernimmt der Verwalter des Backups.

Der Error-Handler sorgt für die fehlerlose Sicherung und für das fehlerlose Rückspeichern der Daten. Eine weitere Aufgabe ist das Melden von Fehlern sowie deren Priorisierung. Die Metadaten-Datenbank enthält für jedes gesicherte Daten-Objekt (Dateien,Verzeichnisse) einen Eintrag. Dieser Eintrag enthält mindestens: Name, Ursprungsrechner, Datum der letzten Änderung, Datum des letzten Backups und Name des Backup-Mediums. Mit Hilfe der Angaben aus der Metadaten-Datenbank werden die Daten aus den inkrementellen oder differentiellen Backups gefunden und wiederhergestellt. Der Media-Manager ist für die Organisation der eingesetzten Sicherungsmedien verantwortlich. Er vergibt eindeutige Identifikationen für jedes

[40] zu sichernde Daten befinden sich auf einem über das Netzwerk erreichbaren Rechner
[41] Anwendung welche für das Backup über das Netzwerk programmiert wurde

Medium, sorgt für die richtige Auswahl des jeweils benötigten Mediums und überwacht deren Alterung. Nach einer von der eingesetzten Technologie abhängigen Anzahl von Lese- und Schreibvorgängen weisen die Medien immer höhere Fehlerraten auf und müssen ausgetauscht werden.

Datenbänder stellen den größten Anteil der benutzten Sicherungsmedien dar. Sie besitzen hohe Speicherkapazitäten (siehe Tabelle 1.7). Nachteil ist der nur sequentiell mögliche Zugriff (technologisch bedingt). MO-Medien[42] stellen ein weiteres Sicherungsmedium dar. Die Speicherkapazitäten wachsen technologiebedingt nicht weiter an (von 128 MByte bis 5,2 GByte). Da der Speicherbedarf und somit auch der Bedarf an großen Sekundärmedien stetig steigt [Ric03], werden diese wegen ihrer geringen Kapazität nur noch selten eingesetzt. Ein Vorteil ist aber der wahlfreie Zugriff auf die abgelegten Daten (technologisch bedingt).

Name	Entwickler	Kapazität in GByte	Transfergeschwindigkeiten
DDS (DAT)	Phillips/Sony	2(4),4(8),12(24)	183k,510k,1M-Byte/s
Exabyte	Exabyte	3.5(7),7(14),20(40)	0.27M,0.5M,3M-Byte/s
AIT	Sony	25(65)	3MByte/s
Magstar MP	IBM	5(10)	6-11 MByte/s
MLR 1 QIC	Tandberg	13(26)	1.5 MByte/s
DLT	Digital	10(20),15(30),20(40),35(70)	1M,1.25,1.5,5M-Byte/s

Tabelle 1.7: Sekundärmedien: Datenbänder

In Tabelle 1.7 sind alle Arten von Datenbänder aufgeführt, sowie deren Entwickler, Speicherkapazitäten und Transfergeschwindigkeiten. Es ist generell die Möglichkeit vorhanden die Daten auf den Bändern zu komprimieren. Die Zahlen in den Klammern geben die Speicherkapazitäten bei Komprimierung wieder. In Tabelle 1.8 ist ein Auszug verfügbarer Backup Programme mit Netzwerk-Sicherung aufgeführt.

Name	Hersteller
NetBackup	Veritas
Arcserver	Computer Associates
Networker	Legato
Tivoli Storage Manager	IBM

Tabelle 1.8: Backup Programme mit Netzwerk-Sicherung (Auszug)

Die in der Tabelle 1.8 aufgeführten Backup Programme stellen eine sogenannte Hierarchische Speicherverwaltung (HSM) zur Verfügung. Diese täuscht dem Benutzer unendlich große physikalische Datenträger vor. HSM lagert Daten, auf welche längere Zeit (Zeitspanne ist konfigurierbar) nicht zugegriffen wurde, von den lokalen Datenträgern (zum Beispiel vom Fileserver)

[42] MO steht für Magnetic Optical, meist CD artige Medien

auf den Backup-Server aus. Lediglich die Metadaten der Dateien (siehe 1.3.2) bleiben auf dem lokalen Dateisystem erhalten. Diese benötigen im Vergleich zu den vollständigen Daten sehr wenig Speicherplatz.

Wird auf den Inhalt einer mit HSM ausgelagerten Datei zugegriffen, blockiert HSM den zugreifenden Prozess und überträgt den Dateiinhalt vom Backup-Server zurück in das lokale Dateisystem (zum Beispiel des Fileservers). Danach wird die Blockierung des Prozesses wieder aufgehoben. Anwendungen oder dem Benutzer bleibt dieser Ablauf, bis auf die längere Zugriffszeit, vollkommen verborgen. Ältere Daten werden automatisch auf billigere Medien (Sekundärspeicher, zum Beispiel Datenbänder) ausgelagert und werden bei Bedarf zurückgeholt[Erk03].

HSM und Datensicherung sind voneinander unabhängige Konzepte. Sie können kombiniert oder einzeln eingesetzt werden.

Kapitel 2

Erstellung des neuen Fileserver-Betriebskonzeptes

Thema dieses Kapitels ist die Erstellung des neuen Fileserver-Betriebskonzeptes am Umweltforschungszentrum (UFZ) Leipzig. Zuvor wird jedoch in Abschnitt 2.1 das alte vorhandene Konzept betrachtet und die Gründe für dessen nötige Ablösung aufgezeigt. Im Anschluss (Abschnitt 2.2) wird das neue Fileserver-Betriebskonzept erläutert und auf dessen Entstehungsprozess eingegangen. In Unterabschnitten werden die wichtigsten Entscheidungen erläutert und begründet. Dabei wird die Wahl des Server-Betriebssystems ausführlich besprochen. Mittels umfangreicher Tests werden verschiedene Dateisysteme auf ihre Eignung im praktischen Betrieb getestet und anschließend ist die Fileserver-Software Gegenstand der Diskussion. Abgeschlossen wird Kapitel 2 mit Abschnitt 2.2 und einem Überblick der getroffenen Entscheidungen bezüglich der neuen Fileserver.

2.1 Die vorhandene Fileserverstruktur am UFZ

Einen Gesamt-Überblick über die vorhandene Fileserverinfrastruktur (altes Betiebskonzept) gibt die Abbildung 2.1. Als zentraler Punkt ist der Fileserver, welcher unter dem Namen VENUS betrieben wird, zu sehen. Die zugrundeliegende Hardware ist eine SUN Microsystems Enterprise 4500, deren wichtigsten technischen Spezifikationen in Tabelle 2.1 aufgeführt sind. An dieser sind mittels Fibre Channel (siehe 1.2.1) zwei Festplatten-Arrays angeschlossen. Das erste besitzt eine Speicherkapazität von 200 GByte und des zweite eine Speicherkapazität 500 GByte. Diese Festplatten-Arrays verfügen über keine Hardware-RAID-Funktionalität (siehe 1.6.1). Somit sind die zentralen Recheneinheiten des Servers (VENUS) für die RAID-Funktionalität zuständig. Um die Datensicherheit zu gewährleisten wurden aus jedem Array, mittels des in SUN-OS 8 enthaltenen Software-RAID's, ein separater RAID-5 Datenträger gebildet.

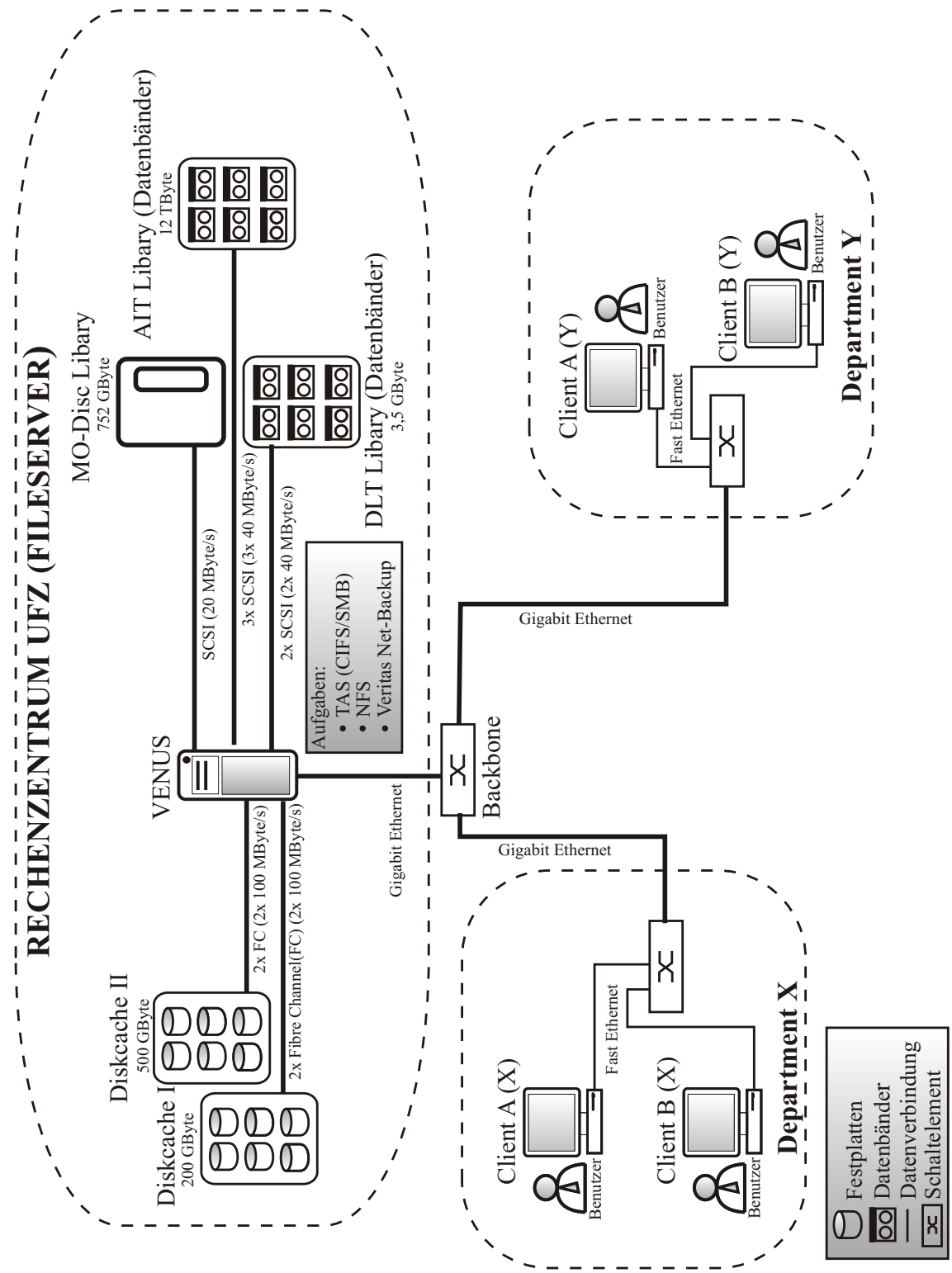

Abbildung 2.1: Die vorhandene Fileserverstruktur am UFZ

Über einen SCSI-BUS [Fie01] sind jeweils zwei Band-Roboter und ein MO-Roboter mit dem Fileserver (VENUS) verbunden. Diese enthalten, organisieren und verwalten die Sekundärmedien für die Hierarchische Speicherverwaltung (HSM) und das Backup (siehe 1.6.2). Über eine Gigabit Ethernet Schnittstelle ist der Fileserver mit dem Backbone[1]-Knoten des UFZ verbunden. Über diese Verbindung und das sich daran anschließende Netzwerk werden die Daten mit den Benutzern, welche an den jeweiligen Client-Rechnern arbeiten, ausgetauscht.

Prozessoren	Vier CPU's mit 480 MHz Taktfrequenz
Arbeitsspeicher	4 GByte
lokale Festplatten	18 GByte RAID 1
Netzwerk-Interfaces	2x Gigabit Ethernet Ports 4x Fast Ethernet Ports
Speicher-Interfaces	6 SCSI Ports 4 Fibre Channel (FC) Ports
Betriebssystem	SUN Microsystems SUN OS 8

Tabelle 2.1: Hardware Spezifikation der SUN Enterprise 4500

Der Server VENUS hat drei Hauptaufgaben zu erfüllen:

- NFS-Server-Dienst (SUN-Microsystems), zum Austausch von Dateien und Anwendungen über das NFS-Protokoll (siehe 1.4.1)
- CIFS/SMB-Server-Dienst, TAS (Total Advanced Server), zum Austausch von Dateien und Anwendungen über das CIFS/SMB-Protokoll (siehe 1.4.2)
- Backup, Veritas NET-BACKUP, Steuerung des zentralen Backups (siehe 1.6.2)

Der Backup-Server sammelt die zu sichernden Daten der einzelnen Backup-Clients über die Nachtstunden ein und speichert sie auf den über Fibre-Channel angeschlossenen Festplatten-Arrays. Auf diesen befindet sich ein SAM Dateisystem der Firma SUN-Microsystems. Dieses verwirklicht die in Abschnitt 1.6.2 genannte Hierarchische Speicherverwaltung (HSM). Von den beiden genannten Fileserver-Diensten bedient der NFS-Server-Dienst bis zu 30 Benutzer (an SUN-Workstations mit SUN-OS Unix), der CIFS-Server-Dienst bis zu 550 Benutzer. Der CIFS-Server-Dienst bedient damit hauptsächlich Clients mit den Microsoft Betriebssystemen Windows NT 4, Windows 2000 sowie Windows XP, aber auch Clients mit dem Apple-Macintosh-Betriebssystem. Der CIFS-Server-Dienst stellt die in Tabelle 2.2 aufgeführten Freigaben den Benutzern zur Verfügung.

[1] Backbone, deutsch: Rückgrat, ist die Bezeichnung für den zentralen Switch (Netzwerkschaltelement, siehe [Kau97]), dieser verbindet alle Netzwerke der Departments und des Rechenzentrums miteinander, und verfügt über eine entsprechende Bandbreite (1 Gbit/s)

Freigabe	Beschreibung
Gruppe	alle Benutzer die der Gruppe X angehören können auf dieser Freigabe X lesen und schreiben
Home	jeder Benutzer besitzt eine Home-Freigabe auf die nur er lesend und schreibend zugreifen darf
Programme	hier befinden sich verschiedene Anwendungsprogramme, welche den Mitarbeitern des UFZ für ihre Arbeit zur Verfügung stehen

Tabelle 2.2: Datei-Freigaben des CIFS-Server-Dienstes (VENUS-Server)

Beide Fileserver-Dienste arbeiten mit den Festplatten-Arrays und deren SAM-FS Dateisystem, wodurch beide von der Hierarchischen Speicherverwaltung profitieren. Wie in Abschnitt 1.6.2 beschrieben, findet hier ein Gesamt-Backup statt, welches auf dem aktuellen Stand gehalten wird. Dabei werden alle Änderungen an den Daten täglich und inkrementell über den Backup-Dienst gesichert. Um die Datensicherheit zu erhöhen wird ein erstes Backup auf den, im Gegensatz zu den Sekundärmedien (Datenbänder), schnelleren Festplatten-Arrays gesichert (daher die Namen Diskcache I und Diskcache II, siehe Abbildung 2.1). Parallel dazu wird die exakte Kopie des Backups auf die langsamen Sekundärmedien kopiert. Dieses Verfahren stellt nach abgeschlossenem Kopiervorgang sicher, daß immer zwei Backups vorhanden sind.

Gründe für die nötige Ablösung

Der zentrale Server VENUS führt gleichzeitig drei verschiedene Dienste aus. Den stetig anwachsenden Anforderungen bezüglich der zu bewältigenden Datenmengen (interne I/O-Last), des Backups, des NFS-Server-Dienstes und vor allen die steigenden CIFS-Server-Dienst-Benutzerzahlen, war der Server immer weniger gewachsen. In Abbildung 2.2 ist die Entwicklung der UFZ-Benutzerzahlen (Ordinatenachse) der verschiedenen Server-Dienste im Laufe der Jahre 1997 bis 2003 (Abszissenachse) zu sehen. Deutlich wird der starke Anstieg der Anzahl der CIFS-Benutzer[2] über die Jahre 1997 bis 2002 gegenüber den anderen Fileserver-Diensten.

In Abbildung 2.3 ist der Anstieg der gespeicherten Datenmengen (Ordinatenachse) der Jahre 1997 bis 2003 (Abszissenachse) abgebildet. Die Benutzer des CIFS-Servers benötigen, verglichen mit den Benutzern der anderen Server-Dienste, jedes Jahr über die doppelte Menge mehr an Speicherplatz. Die Gründe für diesen Anstieg der Datenmengen liegen neben der Zunahme der Benutzerzahlen, auch in der Zweckentfremdung der Freigaben. Daten aller Art (zum Beispiel komplette Programm-CD's, Musik, usw.) wurden auf den Freigaben abgelegt. Dies entsprach nicht mehr dem ursprünglichen Zweck. Der Speicherplatz auf den Fileservern wurde zur

[2] In Abbildung 2.2 sind die CIFS-Benutzerzahlen mit TAS-Nutzer (Total Advanced Server, siehe Tabelle 1.4) bezeichnet

Verfügung gestellt, um wissenschaftliche Daten abzulegen, zu sichern (Home-Freigabe) oder mit anderen diese über das Netzwerk (Gruppen-Freigaben) auszutauschen. Durch den Verzicht auf eine Beschränkung des Speicherplatzes (Quota, siehe 1.3.3) auf den Home-Freigaben wie auch auf der Gruppen-Freigabe waren keinerlei Einschränkungen des Speicherplatz-Verbrauchs möglich.

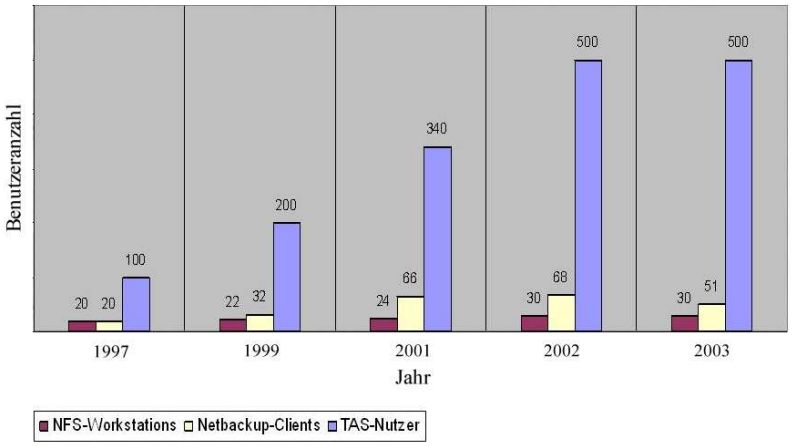

Abbildung 2.2: Die Entwicklung der Benutzerzahlen am UFZ

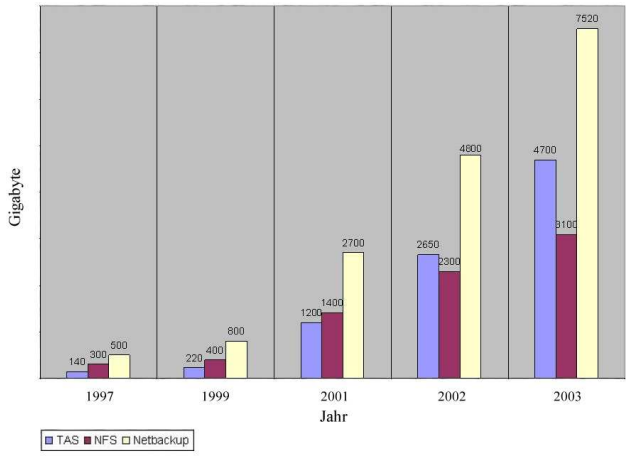

Abbildung 2.3: Die Entwicklung des Datenbestandes am UFZ (in GByte)

Zu den ansteigenden Benutzerzahlen und zunehmenden Datenmengen kam der immer höhere Verwaltungsaufwand (durch das nötige Aufstocken der Speicherkapazität) der über Software-RAID konfigurierten Festplatten-Arrays (Diskcache I und II). Mit Verwaltungsaufwand ist hier die benötigte Rechenleistung für die Realisierung von Software-RAID gemeint.

Trotz seiner vier zentralen Recheneinheiten (CPU's) wird eine durchschnittliche interne I/O (Eingabe/Ausgabe)-Auslastung der vier CPU's von 80 Prozent angezeigt. In den letzten Wochen, vor der Umstellung auf das neuen Betriebskonzept (siehe 2.2), kam es circa zweimal pro Monat zu einem totalen Systemausfall des gesamten Servers (VENUS). Der bei Systemausfällen mögliche Verlust von Daten wurde durch die Journaling Funktion des SUN OS Dateisystems UFS verhindert.

Die Suche nach den genauen Ursachen der Systemabstürze gestaltete sich als schwierig. Durch die ständig notwendige Verfügbarkeit des Servers (24 Stunden Dauerbetrieb, am Tag Fileserver-Dienste, in der Nacht Backup-Server), war eine genaue Untersuchung nur eingeschränkt möglich. Sowohl der Austausch von Hardware (Austausch der zentralen Recheneinheiten) und die Benutzung der aktuellsten verfügbaren Programmversionen der eingesetzten Software brachten keine Besserung. Anhand von Log-Dateien[3] lässt sich nur folgende Ursache für die Ausfälle ermitteln. Laut der Firma, welche für die TAS-Fileserver-Software (Produktunterstützung) verantwortlich ist, häufen sich in Extremsituationen Fehler in der Abarbeitung der Verbindungsanfragen an den TAS-Fileserver-Dienst. Diese Fehler können zum Abstürzen des gesamten Systems führen. Eine Extremsituation stellt zum Beispiel die permanente hohe Überlastung des Fileservers dar. Dieser untragbare Zustand muß mit dem neuen Betriebskonzept beseitigt werden.

Der gerade genannte Fakt, wodurch einer der Server-Dienste (Fileserver-Dienste oder Backupserver-Dienst) die Stabilität des gesamten Systems bedroht, birgt einen weiteren Nachteil des alten Betriebskonzeptes[4].

2.2 Das neue Betriebskonzept

Mit dem neuen Betriebskonzept sollen alle in Abschnitt 2.1 genannten Probleme und Nachteile beseitigt werden, sowie eine Grundlage für den weiteren Ausbau der zur Verfügung stehenden Speicherkapazität geschaffen werden. Es wurden die folgenden Hauptpunkte festgelegt:

1. Anschaffung neuer leistungsfähiger Hardware
2. Anschaffung von Hardware-RAID-Festplatten-Arrays mit entsprechender Speicherkapazität
3. Benutzung eines modernen Dateisystems mit Journaling-Funktion und der Eignung für den Fileserverbetrieb

[3] In den sogenannten Log-Dateien werden Statusmeldungen des Systems oder einzelner Anwendungen gespeichert

[4] zentralisiertes Betriebskonzept, mehrere Server-Dienste befinden sich auf einer Server-Hardware

4. Verzicht auf kommerzielle Software und Einsatz von Open-Source-Software zur Kosteneinsparung
5. Ausbaufähigkeit der Gesamtspeicherkapazität, Einführung einer Speicherplatzbeschränkung durch Quota-Software
6. Aufteilung der verschiedenen Dienste, wie Backup-Server und Fileserver, auf separate Hardware
7. Erhöhung der Ausfallsicherheit und somit Erhöhung der Verfügbarkeit
8. Komplette Ablösung des TAS/SMB-CIFS-Server-Dienstes (Programmfehler)

Durch die Anschaffung von zusätzlicher Server-Hardware soll die interne I/O-Last des alten Servers (VENUS) auf mehrere neue Systeme aufgeteilt werden, um größere Spielräume bei der zukünftigen Ausbaufähigkeit zu schaffen. Der Einsatz externer RAID-Arrays, welche über einen eigenen Hardware-RAID-Logik verfügen (verbunden mit den neuen Servern über SCSI-Bus), verringern die I/O-Last (Entlastung der zentralen Recheneinheiten) zusätzlich. Bei beiden neuen Servern wurde die lokale Speicherung der Daten, als Speicherstrategie (siehe 1.2) mit den geringsten Kosten, gewählt. Die neuen Hardware-RAID-Arrays sind jeweils direkt mit den Servern verbunden. Der Aufbau eines Speichernetzes am UFZ ist für die Zukunft vorgesehen. Ein modernes Dateisystem mit Journaling sorgt im Falle eines Systemabsturzes für die schnelle Herstellung der Verfügbarkeit der neuen Fileserver. Die einzelnen Server-Dienste (Backup-Server-Dienst und Fileserver-Dienst) werden auf separate Server-Hardware verteilt. Ergebnis ist die Verringerung potentieller Fehlerquellen und somit die Erhöhung der Verfügbarkeit und Verringerung der Ausfallrate.

Durch den Einsatz von Open-Source-Software und deren freier Entwicklergemeinde wird eine Unabhängigkeit gegenüber kommerziellen Anbietern und dessen zukünftigen Lizenzmodellen sowie eine Einsparung von anfallenden Lizenzkosten erreicht. Für den kommerziellen TAS-CIFS/SMB-Server belaufen sich die Lizenzkosten auf über 800 Euro pro Monat[5]!

Die Server verfügen über die technischen Voraussetzungen den Speicherbedarf, durch Anschließen zusätzlicher Festplatten-Arrays, für mindestens zwei Jahre zu decken. Durch die Positionierung der neuen Fileserver an verschiedenen Standorten wird die Gesamt-Verfügbarkeit erhöht.

[5] Lizenzkosten von 800 Euro pro Monat für 650 Benutzer

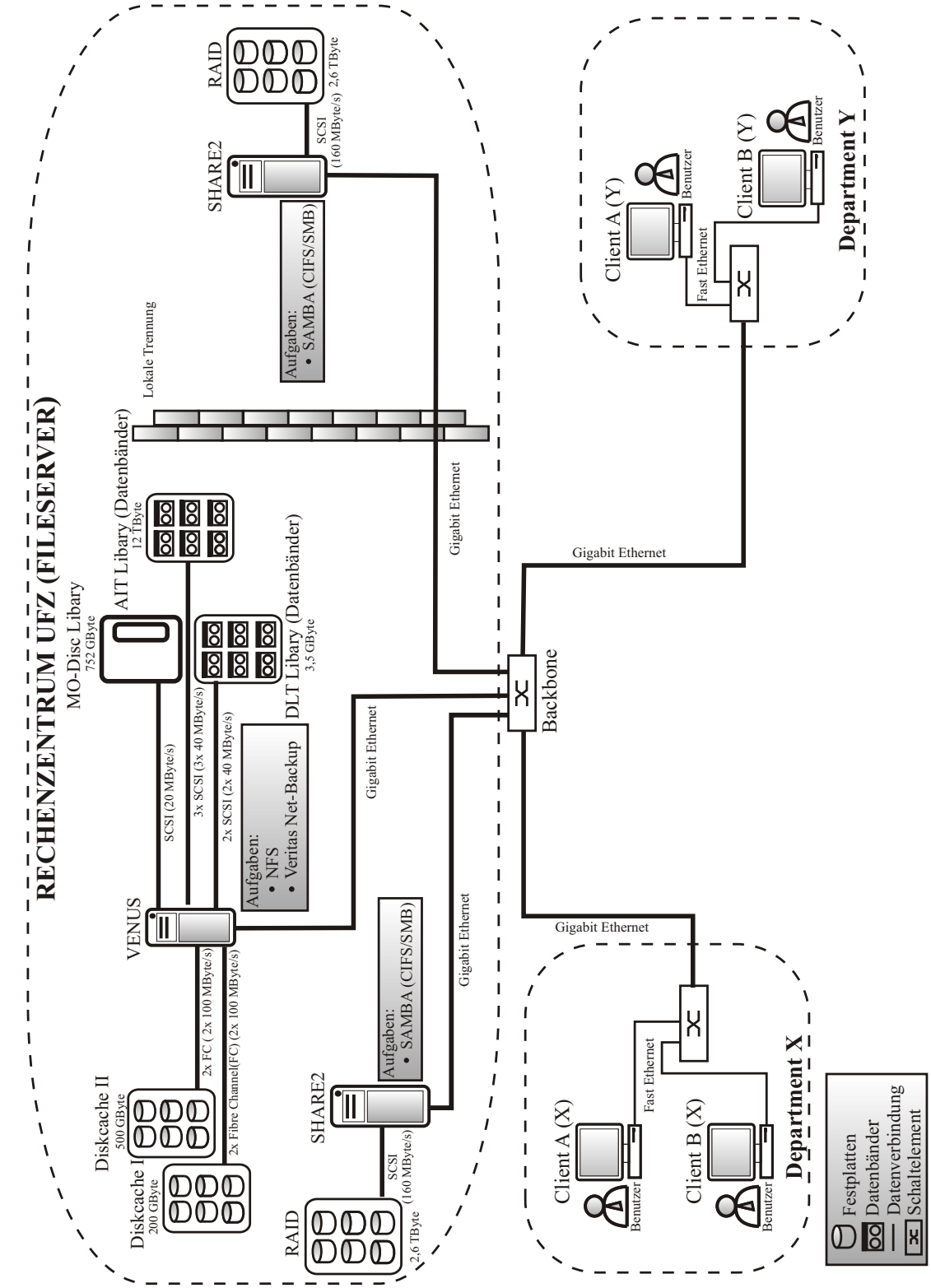

Abbildung 2.4: Die neue Fileserverstruktur am UFZ

Diese wird durch das Abgleichen der Datenbestände beider neuer Fileserver erreicht. Fällt einer der beiden Server aus, kann der andere dessen Aufgaben übernehmen. In diesem Fall würde die I/O-Last zweier Fileserver von einem einzelnen Fileserver bewältigt. Daher sind entsprechende Speicher- und Leistungreservern bei der Hardware-Auswahl der neuen Fileserver zu berücksichtigen.

Aus den genannten Hauptpunkten entstand das neue Betriebskonzept, welches in Abbildung 2.4 dargestellt ist. Hauptaugenmerk ist die Aufteilung der verschiedenen Aufgaben, wie Backup-Server-Dienst und Fileserver-Dienst, auf drei separate Server. Durch diese Aufteilung und die zusätzlichen Server werden neue Ressourcen für einen zukünftigen Ausbau der Speicherkapazität geschaffen. Die I/O-Last des Servers VENUS wird verringert. Die Aufteilung führt zu einer höheren Verfügbarkeit und Ausfallsicherheit. Kommt es zu einem Systemausfall eines einzelnen Servers, arbeiten die anderen Server unabhängig weiter. In Tabelle 2.3 sind die wichtigsten Hardware Spezifikationen der beiden neuen Server aufgeführt.

Prozessoren	Vier Intel Xeon CPU's mit 2,7 GHz Taktfrequenz
Arbeitsspeicher	4 Gbyte
lokale Festplatten	36 GByte RAID 1
Netzwerk-Interfaces	2x Gigabit Ethernet Ports
Speicher-Interfaces	4x SCSI Ports
Festplatten-Array	Dell PowerVault 3,7 TByte Speicherkapazität brutto RAID-Level 0, 1, 10 oder 5 frei wählbar 2,8 TByte netto mit RAID-Level 5

Tabelle 2.3: Hardware Spezifikation Dell 6500

Die beiden neuen Server SHARE1 und SHARE2 werden ausschließlich für den CIFS/SMB-Fileserver-Dienst eingesetzt. Die Aufteilung der Server-Dienste auf die drei Server und deren jeweilige Fileserver-Freigaben sind in Tabelle 2.4 zu sehen. Die neue Freigabe *UFZ-All* stellt eine UFZ-weite Freigabe dar, auf welche jeder Benutzer seine Daten zum Austausch mit anderen Benutzern ablegen kann. Jeder Benutzer besitzt Lese und Schreibrechte auf diese Freigabe.

VENUS	Backup-Server NFS-Server
SHARE1	CIFS-Server Freigaben: UFZ-All und Home
SHARE2	CIFS-Server Freigaben: Gruppe und Programme

Tabelle 2.4: neue Aufgabenverteilung der Server (Server-Dienste)

Der schon vorhandene Server VENUS ist nur noch für den NFS-Fileserver-Dienst, HSM und das Backup zuständig. Auch die Bandroboter und MO-Roboter werden von ihm weiterhin gesteuert und verwaltet. Der Betrieb des TAS-CIFS/SMB-Dienstes wird im Zuge der Umstellung (Migration, siehe Kapitel 3) auf dem Server VENUS vollständig eingestellt.

2.2.1 Wahl des Betriebssystems

Mit der Festlegung des Einsatzes von Open-Source Software standen mit Linux, OpenBSD und FreeBSD drei freie Betriebssysteme auf Unix-Basis zur Auswahl.

Der Hersteller der neuen Server (Dell) unterstützt das vom Distributor Red Hat zusammengestellte Linux als einziges dieser Betriebssysteme offiziell. Diese Unterstützung bezieht sich auf die Gerätetreiber der in den neuen Servern vorhanden Hardware. Das sind die RAID-Controller, welche die Festplatten-Arrays steuern und verwalten, sowie die Netzwerkschnittstellen. Produktunterstützung wird von Dell nur im Zusammenhang mit Red Hat Linux gewährleistet.

Daher musste Red Hat Linux als Betriebssystem für die zwei neuen Fileserver gewählt werden. Red Hat Linux ist laut [Sch03] der größte Linux-Distributor in Nordamerika und verfügt über große Erfahrung im professionellen Server-Umfeld. Die aktuelle Version, welche von Dell unterstützt wird, ist die Version 8.0 .

Red Hat Linux ist aber nur eine Distribution des freien Linux-Betriebssystems. Linux selbst besteht nur aus dem Kernel[6] des Betriebssystems. Es sind keine Verwaltungstools oder Systemprogramme in diesem enthalten. Um ein komplettes Betriebssystem zu erhalten, ist eine Zusammenstellung aller dieser Teile nötig. Diese Zusammenstellung wird von kommerziellen und nicht kommerziellen Vereinen, Firmen und Privatpersonen zu sogenannten Distributionen zusammengepackt.

Typische Vertreter solcher Distributionen sind: Debian Linux, Red Hat Linux, Suse Linux und Mandrake Linux. Diese bieten komplette Softwarepakete an, mit allen nötigen Anwendungen (Systemprogramme, Verwaltungstools, grafische Oberflächen und Anwendungsprogramme) um Linux im professionellen Umfeld einzusetzen.

2.2.2 Speicherplatzbeschränkung (Quota)

Mit einer Beschränkung des Speicherplatzes soll ein Missbrauch, wie im alten Betriebskonzept, vermieden werden. Alle Benutzer wurden aufgefordert nicht mehr benötigte Daten sowie Daten welche nicht mit ihrer Tätigkeit am UFZ in Zusammenhang stehen, je nach Bedarf zu sichern

[6] Kernel: Betriebssystemkern, Grundlegende Routinen eines Betriebbsystems welche für dessen Betrieb unbedingt nötig sind

und anschließend zu löschen. Dieses Aufforderung wurde auch für die Gruppen-Freigaben der einzelnen Departments den jeweiligen Verantwortlichen mitgeteilt.

Mit der auf dem neuen Fileserver (SHARE1) zur Verfügung stehenden Speicherkapazität von 2,8 TByte erhält jeder Benutzer 5 Gigabyte persönlichen Speicherplatz (Home-Freigabe). Diese Größe (5 GByte) ergibt sich aus den circa 500 Benutzern des CIFS-Server-Dienstes bezogen auf 2,8 TByte. Dabei wird angenommen, daß nicht alle Benutzer ihre maximale Kapazität nutzen werden. Sollte die mögliche Gesamtspeicherkapazität erreicht werden, ist die Anschaffung neuer Festplatten-RAID-Arrays geplant. Eine ständige Überwachung der Speicherplatzausnutzung ist dazu notwendig, um rechtzeitig reagieren zu können.

Eine ähnliche Regelung wurde auch für die Gruppen-Freigaben (Fileserver SHARE2) festgelegt. Auf Basis der circa fünfzehn vorhandenen Gruppen am UFZ wurden zehn Gigabyte plus ein Gigabyte pro Gruppenmitglied als Beschränkung des Speicherplatzes festgelegt. Dabei wurde aber eine Maximum von 40 Gigabyte pro Gruppen-Freigabe veranschlagt. Einige der Gruppen benötigen für ihre wissenschaftlichen Daten wesentlich mehr Speicherplatz (größer 200 Gigabyte). Um diesen Gruppen den benötigten Speicherplatz zur Verfügung zu stellen, wurde diese Begrenzung (40 Gigabyte) eingeführt. Benötigt ein Benutzer oder eine Gruppe mehr Speicherplatz, so kann dies beantragt werden. Für die Gewährung werden nur wissenschaftliche oder andere für das UFZ wichtige Gründe akzeptiert. Red Hat Linux ist mit einer vom Dateisystem unabhängigen Quota-Software ausgestattet und erlaubt somit die Realisierung der getroffenen Speicherplatz-Festlegungen.

2.2.3 Auswahl des Dateisystems

Die Festlegung des Betriebssystems auf Linux beschränkt die Wahl auf Dateisysteme, welche für Linux entwickelt oder portiert worden. Trotz dieser scheinbaren Einschränkung ergibt sich eine breite Auswahl. Über 80 verschiedene Dateisysteme sind unter Linux einsetzbar[7].

Vorauswahl geeigneter Dateisysteme

Nicht alle dieser 80 Dateisysteme sind für den Einsatz auf einem Fileserver bestimmt. Den größte Teil stellen veraltete Dateisysteme dar, welche nur noch aus Kompatibilitätsgründen zur Verfügung gestellt werden. Im Abschnitt 1.3.2 wurden die Kriterien für ein Dateisystem, welches auf einem Fileserver zum Einsatz kommen soll, mit folgenden Punkten festgelegt.

- Dateisystemgröße von mindestens 16 TeraByte

[7] Diese hohe Anzahl stammt von der Linux Anwendung **fdisk**, welche das Einrichten und Partitionieren von Datenträgern ermöglicht (Partitionieren, siehe 1.3.4)

- Dateigrößen von mindestens 10 TeraByte
- Unterstützung von Dateiberechtigungen (Rechtevergabe, ACL's)
- Unterstützung von Journaling

Unter Berücksichtigung der genannten Kriterien und des gewählten Betriebssystems wurden drei Dateisysteme ausgewählt. Das sind ext3 von Stephen Tweedies, ReiserFS von Hans Reiser und XFS von Silicon Graphics. Das in 1.3.2 vorgestellte NTFS von Microsoft scheidet aus, da es nur für Microsoft Windows Betriebssysteme verfügbar ist[8].

Geeignete Testumgebungen und Benchmarks

Da es von enormer Wichtigkeit für einen Fileserver ist in welcher Geschwindigkeit dieser Daten liefern oder/und entgegennehmen kann, wurden die drei Dateisysteme auf ihre Performance getestet. Für die drei Dateisysteme der Vorauswahl mussten geeignete Messmethoden für deren Eignung im Bezug auf die Performance gefunden werden. Mit Performance ist hier die Anzahl von I/O-Operationen (Lesen und Schreiben), welche in einer festgelegte Zeitspanne ausgeführt werden können, gemeint.

Eine Orientierung für umfangreiche Dateisystem-Tests (Benchmarks) geben die Dokumente *Filesystem Performance and Scalability in Linux* [u.a02], *PenguinoMeter: A New File-I/O Benchmark for Linux* [u.a01] und *Fürs Protokoll* [Die02]. Getestet wurden Linux-Dateisysteme auf drei verschieden ausgebauten Hardware-Konfigurationen. Dabei wird neben der allgemeinen Performance auch die Skalierbarkeit[9] getestet. Da hier aber nur eine Computer-Hardware (neue Fileserver) zur Verfügung steht, wird die Skalierbarkeit nicht ermittelt. So müssen die Ergebnisse der in [u.a02] durchgeführten Tests herangezogen werden. In Bezug auf die Hardware-Skalierbarkeit wird das Dateisystem XFS als Testsieger genannt.

In [u.a02] und [u.a01] werden drei Tests benutzt Tiobench, Postmark (Filemark) und AIM7. Tiobench und Postmark wurden ausgewählt, da beide frei verfügbar sind (Open-Source-Software) und hauptsächlich auf die Performance des Dateisystems und des I/O Subsystems des Rechners ausgerichtet sind. Im Gegenteil dazu besteht AIM7 aus 53 unterschiedlichen Tests, welche sich sowohl auf die Leistung der CPU, auf die I/O-Leistung des Arbeitsspeichers und auf die I/O-Leistung des Dateisystems konzentrieren. AIM7 ist somit ein Misch-Test. Die technischen Spezifikationen (Recheneinheiten, Arbeitsspeicher, I/O-Subsystem) der neuen Server ändern sich aber nicht! Es sollen nur unterschiedliche Dateisysteme getestet werden. Somit würden

[8] Linux unterstüzt nur das Lesen von NTFS-Datenträgern

[9] Skalierbarkeit eines Dateisystems bedeutet hier: die Fähigkeit eines Dateisystems bei schnellerer Hardware (in Bezug auf die Rechenleistung des Gesamtsystems) im gleichen Maße schneller (im Bezug auf das Schreiben und Lesen von Daten pro Zeiteinheit) zu werden.

die Tests, welche nicht auf das Dateisystem bezogen sind, gleiche Resultate liefern und damit unnötig sein!

In [u.a02] sind drei unterschiedliche Systeme (in Bezug auf die Hardware) beteiligt. Dort ist ein Test mit AIM7 als Mischtest wieder sinnvoll. Im folgenden wird auf die beiden Tests Tiobench und Postmark näher eingegangen.

Tiobench Tiobench ist ein Datei I/O-Test, welcher die Transferrate beim Schreiben in eine Datei und beim Lesen aus einer Datei misst. Dazu wird zuerst eine Datei mit zufälligen Byte-Folgen erstellt. Dabei ist es wichtig die Größe der Datei so zu wählen, daß diese nicht mehr in den Arbeitsspeicher des Servers passt. Wird eine Dateigröße gewählt, welche kleiner als der Arbeitsspeicher ist, so lädt das Betriebssystem die gesamte Datei in den Arbeitsspeicher (Puffer, siehe 1.3.1). Damit erfolgt der Zugriff nicht mehr über das Dateisystem auf dem Datenträger. In diesem Fall wird die Transferrate zwischen Arbeitsspeicher und CPU gemessen. Diese Arbeitsweise ist aber nicht im Sinn des Dateisystem-Tests.

Das **T** im Namen steht für Threaded[10]. Tiobench ermöglicht das Testen des Dateisystems mit einer beliebig wählbaren Anzahl gleichzeitig laufender Einzeltests (Threads). Diese Eigenschaft simuliert sehr genau die Arbeit der eingesetzten Fileserversoftware (siehe 2.2.4). Bei jeder neuen Anfrage zum Ausliefern oder Entgegennehmen von Daten wird ein neuer Prozess gestartet, welcher unabhängig vom Hauptprozess bis zum Abschluss der Anfrage arbeitet. Tiobench wird über die Linux-Kommandozeile gesteuert und nimmt auch seine Parameter, wie zum Beispiel Test-Dateigröße, Anzahl der Threads usw., von dieser entgegen. Diese Betriebsart erlaubt das sequentielle Ausführen von beliebig vielen Test mit unterschiedlichen Parametern.

Tiobench testet vier Eigenschaften des Dateisystems: sequentielles Lesen, zufälliges Lesen, sequentielles Schreiben und zufälliges Schreiben. Alle diese Tests werden mit der zuvor erzeugten Datei durchgeführt. Nach Abschluss des Tests werden die Ergebnisse in einer Resultat-Datei gespeichert. Zusätzlich zu den Transferraten speichert Tiobench die Auslastung der zentralen Recheneinheiten für jeden der vier Tests.

Postmark Im Gegensatz zu Tiobench, welcher nur mit einer Datei arbeitet, simuliert der Postmark-Test das Erstellen, Löschen, Lesen und Schreiben von einer festlegbaren Anzahl von Dateien. Die Testparameter sind wie beschrieben die Anzahl der Dateien, Anzahl der Transaktionen und ein Bereich[11] für die Größe der einzelnen Dateien. Nach dem Starten arbeitet Postmark die vorgegebene Anzahl von Transaktionen ab. Als Ergebnis werden die Lese- und

[10] sinngemäß im deutschen: verteilt

[11] Es wird eine untere und eine obere Grenze für die Dateigrößen festgelegt. Während der Laufzeit des Tests werden Dateien innerhalb dieser festgelegten Grenzen erstellt.

Schreibtransferrate in KByte/s und die Anzahl der Transaktionen pro Sekunde ausgegeben. Auch hier gilt es die Gesamtgröße aller Dateien über der des Arbeitsspeichers festzulegen.

Die Arbeitsweise von Postmark (Erstellen, Löschen, Lesen und Schreiben von vielen Dateien) fordert die internen Organisationsstrukturen der getesteten Dateisysteme (siehe 1.3.2). Aus den Ergebnissen lassen sich Schlüsse auf die Implementierung, bezüglich der Effektivität der Organisation und Verwaltung der Daten, ziehen.

Umfang und Ergebnis der Tests

Durch die Festlegung des Betriebssystems, der zu testenden Dateisysteme und der Test-Programme sind alle Voraussetzungen für einen ausführlichen Test der Dateisysteme gegeben. Dessen Ergebnis soll das auf den externen RAID-Arrays eingesetzte Dateisystem bestimmen. In Tabelle 2.5 sind alle Testparameter nochmals aufgeführt.

Dateisysteme	ext3, ReiserFS, XFS
Betriebssystem	Red Hat Linux 8.0
Test-Programme	Tiobench, Postmark

Tabelle 2.5: Test-Daten

Da beide neuen Server in der Hard- und Software-Ausstattung gleichwertig sind, werden die Tests nur auf einem Server durchgeführt. Betriebssystem und Testprogramme wurden installiert und konfiguriert. Beide RAID-Arrays wurden mit dem in Abschnitt 1.6.1 beschriebenen RAID-Level 5 (bester Kompromiss zwischen Datensicherheit und Speicherplatzverbrauch) eingerichtet. Zusätzlich wurde eine Hot-Spare-Festplatte (siehe 1.6.1) eingerichtet. Die Speicherkapazität der RAID-Arrays beträgt mit RAID-Level 5 insgesamt 2,8 TByte[12]

Während der Tests werden alle nicht testrelevanten Anwendungen beendet. Die Einhaltung dieser Vorgabe ist aber nur begrenzt möglich. Systemanwendungen[13], welche für die Arbeit des Betriebssystem notwendig sind, können nicht beendet werden. Diese arbeiten entsprechend parallel zu den Tests und beeinflussen somit deren Ergebnisse. Die Tests finden auf einem inhomogenen System statt! Um dennoch aussagekräftige Ergebnisse zu erhalten, werden die Tests mehrfach ausgeführt und anschließend der Mittelwert gebildet.

[12] Diese Zahl ergibt sich aus den 3,7 TByte physikalisch vorhandenen Speicherplatz minus 25 Prozent, welche für die Paritätsinformationen des RAID-Levels 5 (siehe 1.6.1) auf jeder Festplatte benötigt werden

[13] zum Beispiel: Hardware-Treiber, Dateisystem-Treiber, Prozessverwaltung usw.

Tiobench-Testreihe

Die erste Testreihe wurde mit Tiobench durchgeführt. Es wurden die folgenden Test-Parameter festgelegt: jeweils zehn Tests mit 4, 8, 16, 32, 64 und 128 Threads (Summe: 60 Tests), Größe der Test-Datei: 5 GByte. Eine Anzahl von zehn Tests wurde gewählt, um einen aussagekräftigen Mittelwert zu erhalten. Zusätzlich wurden für jeden Test der Median ermittelt und die mittlere quadratische Abweichung berechnet. Weichen Mittelwert und Median stark voneinander ab, sind starke Abweichungen unter den Testwerten aufgetreten (inhomogenes Sytem). Sind die Werte von Median und Mittelwert nahezu identisch, sind kaum signifikante Abweichungen unter den Testwerten aufgetreten und der Mittelwert besitzt eine hohe Aussagekraft. Mit dem Wert der mittleren quadratischen Abweichung wurde die prozentuale Abweichung vom Mittelwert berechnet. Dies war nötig um eine testwertunabhängige Größe (Prozent-Wert) zu erhalten. Alle prozentualen Abweichungen eines Gesamt-Tests (sequentieller Lesetest, zufälliger Lesetest, sequentieller Schreibtest, zufälliger Schreibtest) wurden anschließend gemittelt. Mit diesem Wert ist es möglich eine Aussage über alle mittleren quadratischen Abweichungen eines Gesamt-Tests zu treffen. Je näher dieser bei Null liegt, desto weniger Abweichungen gab es unter den einzelnen Testwerten eines Gesamtests.

Die sechs verschiedenen Thread Vorgaben ermöglichen Aussagen bezüglich der Skalierbarkeit der Dateisysteme bei paralleler Beanspruchung. Die Dateigröße von 5 Gigabyte wurde gewählt, um eine Verlagerung der Tests in den System-Arbeitsspeicher zu verhindern (siehe 2.2.3). Die Resultate (berechnete Mittelwerte) der Tests sind in den folgenden Diagrammen dargestellt und werden jeweils diskutiert. Die Abszissenachse stellt die Anzahl der Threads und die Ordinatenachse die Transferrate in MByte pro Sekunde sowie die Auslastung der Recheneinheiten (CPU's) in Prozent dar. Die aufgeführte Prozentzahl ist durch vier (Server besitzen vier zentrale Recheneinheiten) zu teilen, um die wirkliche CPU-Auslastung des Gesamtsystems zu erhalten. Die genauen Testergebnisse in Form von Tabellen sind im Anhang 6.1 zu finden. Dort sind Mittelwert, Median und die mittlere quadratische Abweichung für jedes einzelne Ergebnis aufgeführt.

Sequentieller Lesetest: In Abbildung 2.5 sind die Ergebnisse des Tiobench-Test für das sequentielle Lesen von Daten zu sehen. XFS bietet die höchste Performance in diesem Test. Im Vergleich weist XFS eine um etwa fünffach höhere Performance gegenüber ext3 und eine um ein drittel höhere Performance gegenüber ReiserFS im gesamten sequentiellen Lesetest auf. XFS belastet aber die Recheneinheiten um das doppelte höher als ReiserFS und sogar um das vierfache höher als ext3. In Abhängigkeit der Threads steigen die Transferraten und die Belastung der zentralen Recheneinheiten aller Dateisysteme, mit Ausnahme der Transferrate von ext3 und ReiserFS, stetig an. Bei allen Dateisystemen ist bei 16 und 32 Threads ein Abflachen des Anstieges beider Werte zu beobachten (Ausnahme ext3). Zwei Ausnahmen stellen

die Transferraten von ext3 und RaiserFS dar. Während sich die Transferrate von ext3 über den gesamten Testverlauf auf gleichem niedrigen Niveau bewegt, nimmt die Transferrate von RaiserFS ab 32 Threads leicht ab. Die Vergleiche der Mittelwerte mit den Medianen ergeben nur sehr geringe Abweichungen (siehe 6.1). Für die mittlere quadratische Abweichung (Mittelwert aller mittleren quadratischen Abweichungen im sequentiellen Lesetest) wurde ein Wert von 3,31 Prozent errechnet. Somit besitzen die ermittelten Testergebnisse eine hohe Aussagekraft.

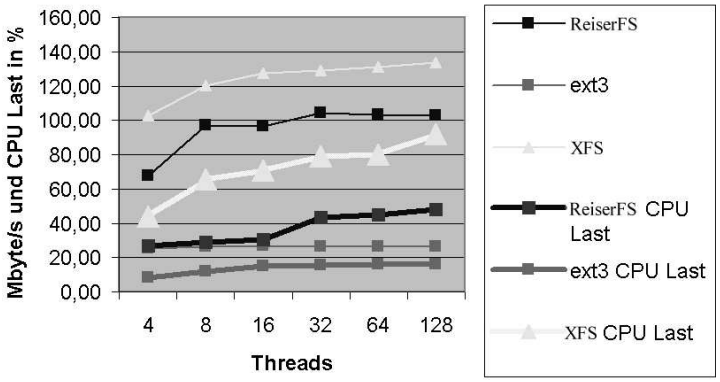

Abbildung 2.5: Die Tiobench-Ergebnisse für sequentielle Lesezugriffe

Zufälliger Lesetest: Das in Abbildung 2.6 dargestellte Diagramm des zufälligen Lesetests zeigt einen durchwachsenen Testverlauf. XFS und ReiserFS liegen in der Transferraten-Performance über den gesamten Testverlauf in etwa gleichauf. Gegenüber ext3 erreichen sie die doppelte Transferrate (128 Threads). Die Transferraten und die Belastung der Recheneinheiten sind in diesem Test wesentlich geringer, verglichen mit denen des sequentiellen Lesetest.

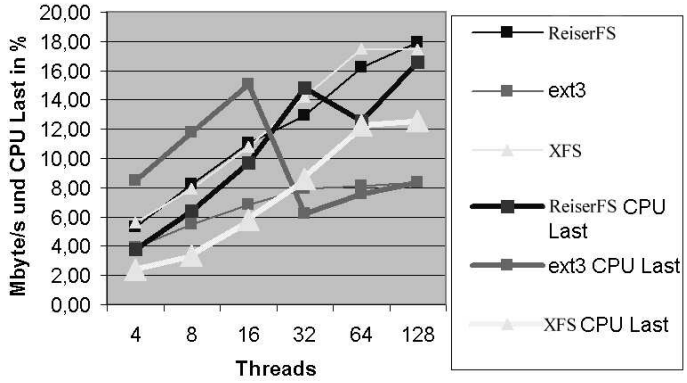

Abbildung 2.6: Die Tiobench-Ergebnisse für zufällige Lesezugriffe

Die Ursache sind die mechanischen/physikalische Eigenschaften der Datenträger (Festplatten). Durch das zufällige Lesen an verschiedenen Stellen des Datenträgers müssen die Leseköpfe der Festplatten immer wieder neu positioniert werden. Das kostet Zeit, in der sich die Recheneinheiten im Ruhezustand (Wartend) befinden und kein Daten-Transfer stattfindet. Somit ergibt sich im Mittel eine geringere Belastung und eine geringere Transferrate. Die Transferraten und die Belastung der Recheneinheiten steigen während der zunehmenden Thread-Anzahl bei allen Dateisystemen in etwa linear an (Ausnahme: CPU Belastung von ext3 und ReiserFS). Auffällig ist der starke Einbruch der CPU-Belastung von ext3 zwischen 16 und 32 Threads und ReiserFS zwischen 32 und 64 Threads. Dieses Verhalten deutet auf interne Probleme bei der Implementierung oder der Organisation/Verwaltung des Dateisystems hin. Die Mittelwerte und Mediane zeigen miteinander verglichen nur geringe Unterschiede (siehe 6.1). Für die mittlere quadratische Abweichung (Mittelwert aller mittleren quadratischen Abweichungen im zufälligen Lesetest) wurde ein erhöhter Wert von 5,59 Prozent errechnet. Dieser Wert zeigt, im Gegensatz zum Mittelwert/Median-Vergleich, hohe Abweichungen unter den einzelnen Testwerten an. Die Ursachen sind die bereits genannten Probleme (Testdurchführung auf inhomogenen Systemen).

Sequentieller Schreibtest: Es folgt das Diagramm mit den Testergebnissen des sequentiellen Schreibtests in Abbildung 2.7. Alle Werte bewegen sich über den gesamten Testverlauf (Threadabhängigkeit) auf etwa gleichem Niveau. Die Transferraten aller Dateisysteme sind mit geringen Abweichungen als gleichwertig anzusehen.

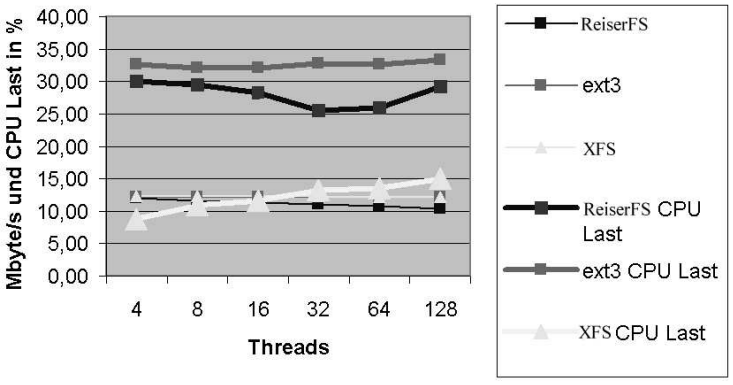

Abbildung 2.7: Die Tiobench-Ergebnisse für sequentielle Schreibzugriffe

Die jeweilige Belastung der Recheneinheiten durch die Dateisysteme sind aber verschieden. Während sich XFS zwischen acht (4 Threads) und fünfzehn (128 Threads) Prozent bewegt, ist die Belastung der Recheneinheiten bei ext3 und ReiserFS um das dreifache höher. XFS benötigt bei gleicher Transferrate weniger Rechenleistung im Vergleich zu ext3 und ReiserFS

und ist somit das leistungsfähigste Dateisystem im sequentiellen Schreibtest. Die Vergleiche der Mittelwerte mit den Medianen ergeben nur sehr geringe Abweichungen (siehe 6.1). Für die mittlere quadratische Abweichung (Mittelwert aller mittleren quadratischen Abweichungen im sequentiellen Schreibtest) wurde ein sehr guter Wert von 0,42 Prozent errechnet. Somit besitzen die ermittelten Testergebnisse eine hohe Aussagekraft.

Der Vergleich zwischen den hohen Transferraten der sequentiellen Lesetests und den niedrigen Transferraten der sequentiellen Schreibttests zeigt den Unterschied zwischen Lese- und Schreiboperationen. Das Schreiben von Daten stellt im Vergleich zum Lesen eine wesentlich höhere Anforderung an das Dateisystem und an die Datenträger dar. Diese Anforderungen sind zum Beispiel auf der Seite des Dateisystems das Schreiben des Journals (siehe 1.3.2), Anlegen und Verändern der Inodes, das Anpassen der Datenträger-Bitmaps und auf der Seite des RAID-Controllers das Berechnen der Paritätsinformationen (siehe 1.6.1).

Zufälliger Schreibtest: Im Diagramm des zufälligen Schreibtests (siehe Abbildung 2.8) zeigen sich ähnliche Ergebnisse, verglichen mit dem sequentiellen Schreibtest in Abbildung 2.7. Die Transferraten-Werte liegen bei allen getesteten Dateisystemen über dem gesamten Testverlauf (Threadabhängigkeit) auf gleichen Niveau.

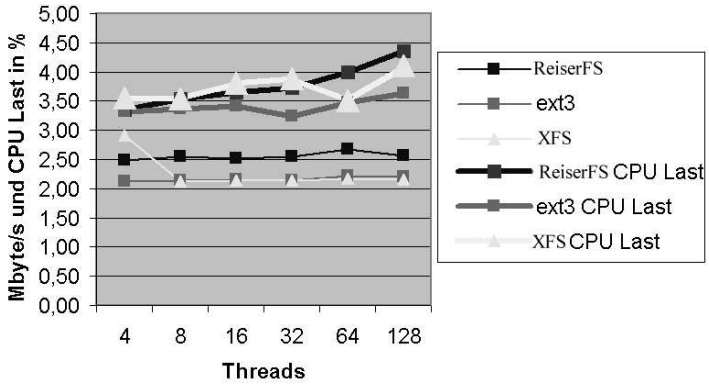

Abbildung 2.8: Die Tiobench-Ergebnisse für zufällige Schreibzugriffe

Im Gegensatz zum sequentiellen Schreibtest liegen hier auch alle Werte für die Belastung der Recheneinheiten über den gesamten Testverlauf auf gleicher Höhe (mit einer leichten Zunahme bei 64 und 128 Threads). Verglichen mit den Transferraten und den Belastungen der Recheneinheiten des sequentiellen Schreibtests sind diese im zufälligen Schreibtests um etwa ein viertel (Transferraten) und ein zehntel (Belastungen der Recheneinheiten) niedriger. Die Gründe für diese niedrigen Ergebnisse sind unter anderen die bereits im zufälligen Lesetest erläuterten mechanischen/physikalische Eigenschaften der Datenträger (Festplatten).

Hinzu kommen noch die nötigen Operationen zum Schreiben der Daten auf die Datenträger (siehe sequentieller Schreibtest). Diese Faktoren begrenzen die Leistung der Dateisysteme. Diese können somit ihre jeweiligen technologischen Vorteile nicht ausnutzen und zeigen ähnliche Ergebnisse! Die Mittelwerte und Mediane zeigen miteinander verglichen nur geringe Unterschiede (siehe 6.1). Für die mittlere quadratische Abweichung (Mittelwert aller mittleren quadratischen Abweichungen im zufälligen Schreibtest) wurde ein guter Wert von 1,82 Prozent errechnet. Somit besitzen die ermittelten Testergebnisse eine hohe Aussagekraft.

Postmark-Testreihe

Es wurden die folgenden Test-Parameter für einen Testverlauf festgelegt: drei Einzeltests mit jeweils 1000 Dateien und 50.000 Transaktionen, 20 000 Dateien und 50 000 Transaktionen, 20 000 Dateien und 100 000 Transaktionen; Dateigrößen jeweils von 0,05 MByte bis 5 MByte. Die Ergebnisse waren jeweils: Transferrate für Lesezugriffe und Transferrate für Schreibzugriffe in kByte pro Sekunde sowie durchgeführte Transaktionen pro Sekunde. Dieser Testverlauf wurde zehnmal durchgeführt (ergibt 30 Einzeltests), um ein Gesamtergebnis unabhängig von eventuell auftretende Abweichungen zu erhalten. Um Abweichungen in den Einzeltests zu erkennen, wurde auch in der Postmark-Testreihe zusätzlich zum Mittelwert der Median sowie die mittlere quadratische Abweichung jedes Einzeltests berechnet.

Die Resultate (berechnete Mittelwerte) der Tests sind in den folgenden Diagrammen dargestellt und werden jeweils diskutiert. An der Abszissenachse sind die drei vorgegebenen Testparameter angegeben und an der Ordinatenachse sind die Transferraten in kByte/s für die Lesezugriffe/Schreibzugriffe oder die Transaktionen pro Sekunde dargestellt. Die genauen Testergebnisse sind im Anhang (siehe 6.1) aufgeführt. Dort sind Mittelwert, Median und die mittlere quadratische Abweichung für jedes einzelne Ergebnis aufgeführt.

Lesetest: In Abbildung 2.9 ist das Diagramm mit den Postmark-Ergebnissen für die Lesetests zu sehen. Deutlich wird der starke Abfall der Lese-Transferrate beim Übergang zwischen 1000 Dateien mit 20 000 Transaktionen und 20 000 Dateien mit 50 000 Transaktionen. Dieses Verhalten ist bei allen ermittelten Postmark-Test-Ergebnissen zu beobachten. Nach dem starken Abfall besitzen alle Dateisysteme die gleichen Lese-Transferraten. Ein Blick auf die genauen Ergebnisse (siehe Anhang 6.1) erlaubt die folgenden Schlüsse. Bei einer geringen Anzahl von Dateien und Transaktionen (1000 Dateien, 20 000 Transaktionen) erreicht das Dateisystem ext3 die höchste Leserate. Dagegen erreicht ReiserFS nur etwa die Hälfte und XFS nur etwa ein Fünftel der Leserate von ext3. Anschließend erfolgt der bereits beschriebene Einbruch der Leserate aller Dateisysteme auf etwa 750 kByte/s. Die Leserate verbleibt anschließend auf die-

sem Niveau (750 kByte/s). Die mittlere quadratische Abweichung beträgt im Lese-Test 4,58 Prozent.

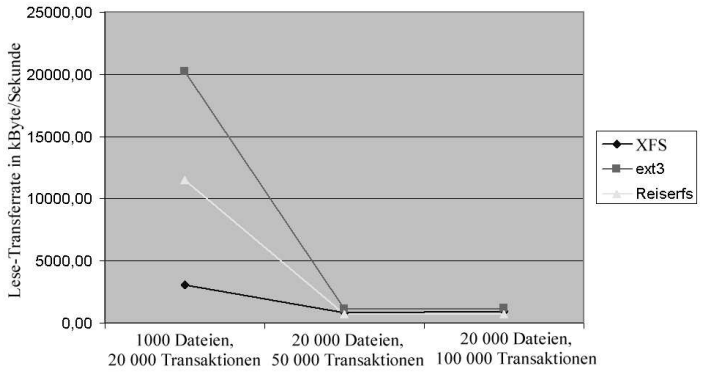

Abbildung 2.9: Die Postmark Ergebnisse für Lesezugriffe

Schreibtest: Es folgen die im Diagramm (siehe Abbildung 2.10) dargestellten Ergebnisse des Schreibtests. Der Testverlauf entspricht dem des Lesetests. Es werden bei 1000 Dateien und 20 000 Transaktionen um 15 Prozent höhere Ergebnisse (alle Dateisysteme) im Vergleich zum Lesetest erreicht. Ab 20 000 Dateien und 50 000 Transaktionen brechen alle Dateisysteme auf etwa 2000 kByte/s ein. Anschließend verbleibt die Schreibrate aller Dateisysteme auf etwa gleichem Niveau (2000 kByte/s). Die mittlere quadratische Abweichung betrug in diesem Test 5,20 Prozent.

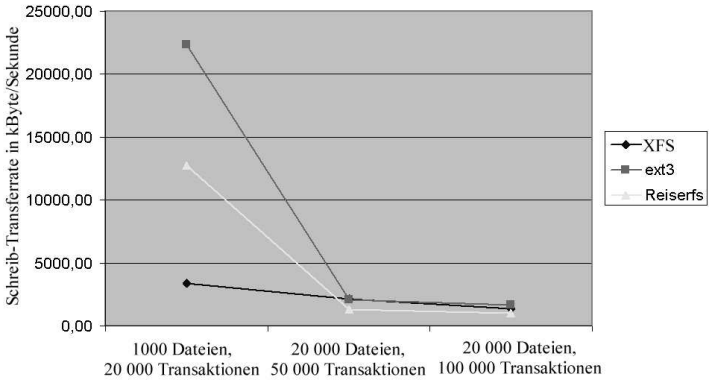

Abbildung 2.10: Die Postmark Ergebnisse für Schreibzugriffe

Transaktionstest: In Abbildung 2.11 ist das Diagramm des Transaktionstests zu sehen. Die Transaktionen pro Sekunde werden aus der Summe der gesamten Schreib- und Lesezugriffe pro

Sekunde während eines Einzeltests ermittelt. Der Testverlauf im Transaktionstest ist identisch zum Lese- und Schreibtest. Wie in diesen liegt ext3 vor RaiserFS und XFS. Dabei ereicht ReiserFS in etwa die Hälfte und XFS ein Sechstel der Ergebnisse von ext3 bei 1000 Dateien und 20 000 Transaktionen. Ab 20 000 Dateien und 50 000 Transaktionen brechen alle Dateisysteme auf etwa 400 Transaktionen pro Sekunde ein. Die Transaktionsrate aller Dateisysteme verbleibt anschließend auf diesen Niveau (400 Transaktionen pro Sekunde). Die mittlere quadratische Abweichung beträgt im Transaktions-Test 5,80 Prozent.

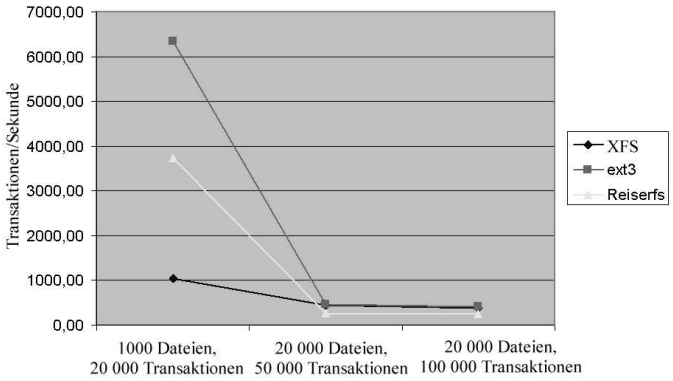

Abbildung 2.11: Die Postmark Ergebnisse für die Transaktionen/s

Während der Mittelwert/Median-Vergleich eine hohe Aussagekraft aller Messwerte bescheinigt, sprechen die erechneten hohen Werte der mittleren quadratischen Abweichung dagegen. Auch hier sind die bereits genannten Probleme (Testdurchführung auf einem inhomogenen System (Fileserver)) Ursache für die erhöhten Werte (mittlere quadratische Abweichung).

Auswertung der Tests und Wahl des Dateisystems

In den Tiobench-Tests zeigte sich das XFS Dateisystem von SGI, hauptsächlich im sequentiellen Lesetest, überlegen. Dort kam es mit fast 134 MByte/s (128 Threads) der theoretisch möglichen maximalen Transferrate der SCSI-Verbindung (160 MByte/s) zwischen Server und den RAID-Arrays sehr nah. Der Einbruch der CPU-Belastungs-Kennlinie von ext3 und ReiserFS im zufälligen Lesezugriffstest zeugt von Problemen bei der Programmierung oder der Verwaltung/Organisation des Dateisystems. Parallel zu diesem Phänomen stagniert auch die Transferrate von ext3 in diesem Test.

Weiterhin spricht für XFS die im sequentiellen Schreibtest niedrige Belastung der Recheneinheiten. Im zufälligen Schreibzugriffstest liegen alle Dateisysteme auf gleichem Niveau. Hier begrenzen die bereits genannten physikalischen/mechanischen Trägheiten der Datenträger die Leistungsentfaltung der Dateisysteme.

Alle durchgeführten Postmark-Tests zeigen einen einheitlichen Diagrammverlauf. Das Dateisystem ext3 führt zu Beginn des Tests (5000 Dateien mit 20 000 Transaktionen) die Bewertung an. Es folgt ReiserFS mit etwa der Hälfte und XFS mit einem Viertel des ext3 Ergebnisses. Ab 20 000 Dateien mit 50 000 Transaktionen pendeln sich alle Dateisysteme in ihrem Ergebnis auf das gleiche niedrige Niveau ein. Dieser Testverlauf gilt sowohl für die Lese/Schreibzugriffstests als auch für den Transaktionstest.

Im realen Betrieb der Fileserver sind Transaktionen mit mehr als 5000 Dateien die Regel. Somit sind für die Bewertung der Dateisysteme die letzten beiden Postmark-Test relevant (20 000 Dateien mit 50 000 Transaktionen, 20 000 Dateien mit 100 000 Transaktionen). Mit deren Ergebnissen konnten abschließend keine relevanten Aussagen über die Leistungsfähigkeit der einzelnen Dateisysteme getroffen werden!

Ausgehend von allen ermittelten Testergebnissen (vorrangig Ergebnisse des Tiobench-Tests) wurde XFS als Dateisystem für die neuen Fileserver gewählt. Die hohen Transferraten bei den Lesezugriffen und die moderaten Belastungen der Recheneinheiten sind die Vorteile auf der Performance-Seite von XFS. Auf der Seite der Dateisystem-Merkmale sprechen für den Einsatz von XFS die moderne Organisations/Verwaltungsstruktur (siehe 1.3.2), die Nutzung von ACL's, erweiterte Attribute und der integrierte Journalingmechanismus. Die Ergebnisse der Tests und die daraus abgeleiteten Schlussfolgerungen, für eine Wahl von XFS als leistungsfähigstes Dateisystem bei großen Speichermengen (größer 2TB), entsprechen denen in [u.a02], [u.a01] und [Die02].

2.2.4 Fileserver-Software

Durch die Festlegungen (siehe 2.2) auf den Einsatz von Open-Source-Software, sowie die Wahl des Linux-Betriebssystems (siehe 2.2.1), stand nur noch die von Andrew Tridgell entwickelte Samba-Fileserver-Software zur Auswahl. Der Samba SMB/CIFS-Server ist Open-Source Software und ist auf allen Linux/Unix-Systemen lauffähig. Er wurde entwickelt, um Microsoft-Windows Fileserver zu ersetzen und damit eine unabhängige freie Implementierung und Nutzung des SMB/CIFS-Protokolls zu ermöglichen. Durch diesen Ersatz entfallen auch die Lizenzgebühren, welche Microsoft für seine Windows-Server-Betriebssysteme erhebt. Der Einsatz von Microsoft-Windows als Fileserver-Betriebssystem und Client-Betriebssystem würde in einem Netzwerk mit 50 Client-Rechnern ca. 10 340 Euro kosten[14].

Mit einem generellen Einsatz von Open-Source-Software (Server- und Client-Seite) würde eine

[14] Kosten der Fileserver-Software (Windows 2003 Server) plus Lizenzkosten für jeden über das Netzwerk verbundenen Client-Rechner (Verbindungslizenz) plus Kosten für das Client-Betriebssystem (Windows XP Proffesional), Quelle: www.SienerSoft.de (Software-Handel) Stand: 03.11.2003

unbegrenzte Anzahl an Servern und Clients praktisch Kostenfrei[15] möglich sein. Am UFZ sind über 94 Prozent aller Client-Rechner mit Microsoft-Windows-Betriebssystemen ausgerüstet. Ein Umstieg auf ein freies Betriebssystem (zum Beispiel Linux) wäre durch die benutzten Anwendungsprogramme, welche nur für Microsoft-Betriebssysteme verfügbar sind, unmöglich. Somit bleibt der Betriebssystem-Umstieg auf absehbare Zeit auf die Server-Seite (Fileserver) beschränkt.

Der Fileserver Samba im Detail

Samba ist direkt über die Internetseite http:
www.samba.org zu beziehen. Er ist auch Bestandteil aller Linux-Distributionen. Entwickelt wurde Samba unter Führung von Andrew Tridgell sowie vielen weiteren unabhängigen Programmierern. Er profitiert von seiner modularen Struktur (jede neue Verbindung startet einen neuen Prozess) und unterstützt/nutzt Systeme mit mehreren Recheneinheiten[Len02]. Samba verfügt über 200 konfigurierbare Optionen und wird über eine einzige zentrale Konfigurationsdatei (smb.conf) gesteuert.

Beispiel einer einfachen Konfiguration:

```
[global]
  workgroups = samba
  netbios name = sambaserver
  interface = eth0

[homes]
  path = /pub
  writable = yes

[programme]
  path = /programme
  writable = no
```

Die smb.conf-Datei teilt sich in zwei Hauptabschnitte. Der erste ist der Globale Teil (mit `[global]` bezeichnet). In diesem Teil werden die übergeordneten Parameter festgelegt, welche für Samba und alle Freigaben (siehe 1.4), gelten. Im obigen Beispiel wird mit `workgroups = samba` der Name des Fileservers festgelegt. Mit `netbios name = sambaserver` der NetBios-Name. Mit `interface = eth0` wird die Netzwerkschnittstelle bezeichnet, über welche Samba mit dem Netzwerk und somit den Client-Rechnern verbunden ist und Daten austauscht.

[15] ausgenommen die Kosten welche für Wartung und Pflege entstehen

Der zweite Teil enthält die Freigaben-Definitionen. Im obigen Beispiel wird eine Freigabe mit der Bezeichnung `[homes]` sowie eine Freigabe `[programme]` angelegt. Innerhalb der smb.conf können beliebig viele Freigaben-Abschnitte angelegt werden. Mit dem Parameter `path = /pub` wird der Ort (Verzeichnis) festgelegt, an welchem sich die Daten der Freigabe auf dem Server befinden und entsprechend abgelegt werden. Der Parameter `writable = yes` erlaubt den Benutzern Daten abzulegen/zu schreiben (Daten lesen ist standarmäßig erlaubt). In Abbildung 1.14 auf Seite 30 ist die Freigabe (`homes`) auf einem Client-Rechner mit einem Microsoft-Windows-Betriebssystem zu sehen. Die genaue Bedeutung der einzelnen Optionen wird in Abschnitt 3.2.1 im 3. Kapitel erläutert.

Weiterhin verfügt Samba über eine einfache Administrationsoberfläche (SWAT), welche dem Benutzer erlaubt ihn über einen Internet-Browser zu steuern. Samba besitzt neben der Funktion als Fileserver, die Möglichkeit Netzwerkdrucker bereitzustellen und zu verwalten[u.a99]. Diese Funktion bleibt aber außen vor, da sie bei einem reinen Fileserver nicht von Interesse ist.

2.2.5 Überblick über die Festlegungen

In Tabelle 2.6 sind alle getroffenen Festlegungen für die neuen Fileserverinfrastruktur nochmals zum Überblick aufgeführt. Alle anderen Festlegungen (Server- und RAID-Hardware) waren vorgegeben und konnten nicht beeinflusst werden.

Hardware-Festlegungen	
RAID-Level für die Festplatten-Arrays	RAID-Level 5 eine Hot-Spare-Festplatte je Array
Software-Festlegungen	
genutztes Dateisystem	XFS von Silicon Graphics Version 1.3
Betriebssystem	Red Hat Linux Version 8.0
Fileserver-Software	Samba-CIFS/SMB Fileserver Version 2.2.8a
Speicherplatz-Festlegungen	
pro Benutzer	5 Gigabyte
pro Gruppe	10 Gigabyte Grundkapazität + 1 Gigabyte pro Gruppenmitglied maximal 40 Gigabyte

Tabelle 2.6: Festlegungen der Betriebsparameter für die neuen Fileserver

Kapitel 3

Migration

Der Begriff Migration steht in der Informations-Technologie für den Übergang von einem alten System zu einem Neuen. Unter System wird hier eine Hardware- und/oder Softwarelösung für einen bestimmten Arbeitsablauf verstanden. Die Migration erfordert eine entsprechende Planung der Maßnahmen, um den Übergang mit geringen oder ohne Einschränkungen für den normalen Betrieb zu vollziehen. Die Erstellung des Migrationskonzeptes für den Übergang vom alten Betriebskonzept zum neuen Betriebskonzept ist Thema des Abschnittes 3.1. Der folgende Abschnitt 3.2 behandelt die Konfiguration und Anpassung der Software (hier Fileserversoftware) die notwendig wird, um die benötigte Funktionalität sowohl auf Seiten der Server als auch auf Seiten der Clients herzustellen. In Abschnitt 3.3 werden administrative Themen der Umstellung betrachtet. Der letzte Abschnitt 3.4 behandelt den genauen organisatorischen Ablauf einer Departmentumstellung auf das neue Betriebskonzept.

3.1 Erstellung des Migrationskonzeptes

Der Übergang vom alten Betriebskonzept (siehe 2.1) zum neuen Betriebskonzept (siehe 2.2) erfordert eine überlegte Planung. Während der Umstellung wird ein uneingeschränkter Betrieb des Fileserver-Dienstes gefordert. Der erste Schritt der Umstellung und zugleich auch der zeitintensivste ist die Übertragung der Daten vom altem Fileserver (VENUS) auf die neuen Fileserver.

Das Kopieren der Daten wird pro Department einzeln durchgeführt, um nacheinander auf die neuen Fileserver umzustellen. Die Daten eines Departments bestehen jeweils aus dem gemeinsamen Gruppen-Verzeichnis und den Home-Verzeichnissen der eigenen Mitarbeiter. In Tabelle 2.4 (Seite 55) ist die Aufteilung der jeweiligen Daten-Verzeichnisse auf die neuen Fileserver aufgeführt.

Durch die Verwendung von HSM (siehe 1.6.2) auf dem alten Fileserver, wurden ältere Daten

auf die Sekundärmedien, Band-Roboter oder MO-Roboter, ausgelagert. Die Transferraten von und zu den Robotern (angeschlossen an den alten Fileserver), sowie deren Reaktionszeiten[1] sind nicht mit denen von Datenträgern, wie zum Beispiel Festplatten[2], zu vergleichen. Je nach Auslastungsgrad oder Anzahl der Aufträge, welcher der Roboter zu einem Zeitpunkt gleichzeitig bearbeitet, können bis zur Verfügbarkeit der Daten 10 Sekunden bis 5 Minuten vergehen. Um die Verfügbarkeit der Daten am Tage für die Benutzer nicht durch noch längere Wartezeiten zu mindern, wurde die Übertragung der Daten auf die neuen Fileserver in die Abend- und Nachtstunden verlagert. Die Transferraten der Übertragung lagen je nach Auslastung der Roboter[3] zwischen 5 bis 200 GByte pro Nacht.

Ein weiterer zu beachtender Umstand war die Synchronität der Daten. Die Übertragung der Daten eines Departments dauerte, je nach Größe, mehrere Tage. Arbeiteten die Benutzer aber auf dem alten Fileserver an Daten, welche schon teilweise auf die neuen Fileserver kopiert wurden, ergab sich eine Asynchronität zwischen den Originaldaten und deren Kopien. Um diese Asynchronität zu verhindern wurden die Daten jede Nacht, bis zum Abschluss der gesamten Übertragung und der endgültigen Umstellung eines Departments, jeweils mit dem Linux/Unix-Programm *rsync* synchronisiert. Das Programm *rsync* überträgt dazu nur die Anteile der Daten, welche sich in der Zwischenzeit seit dem letzten Kopiervorgang verändert haben[MG03]. Es stellt somit die Synchronität zwischen Original und Kopie wieder her.

Bevor jedoch die eigentliche Umstellung, nach dem Kopieren der Daten auf die neuen Fileserver, beginnen konnte, mußte die Fileserver-Software Samba installiert und an die Anforderungen im UFZ angepasst werden (siehe 3.2). Das Betriebssystem war bereits auf den Servern herstellerseitig vorinstalliert und wurde mit Hilfe der Red-Hat eigenen Software *up2date* auf den neuesten Stand gebracht. Diese behebt existierende Fehler und festgestellte Sicherheitslöcher in den installierten Anwendungen, Diensten und im Betriebssystem, indem sie die benötigten Korrekturen eigenständig aus dem Internet lädt.

Weiterhin müssen Lösungen (siehe 3.3) für die Authentifizierung der Benutzer gegenüber den neuen Fileservern gefunden werden. Der Speicherplatz für die Benutzer und Gruppen ist zu beschränken (Quota) und eine Synchronisation der Datenbestände zwischen den neuen Fileservern (Backup) ist in regelmäßigen Abständen zu gewährleisten. Sind diese Lösungen gefunden und ist die Datenübertragung eines Departments abgeschlossen, wird dieses auf die neuen Fileserver umgestellt (siehe 3.4).

In Abbildung 3.1 ist der Ablauf der Umstellung zur besseren Übersicht grafisch dargestellt. Parallel zur Übertragung der Daten vom alten Fileserver zu den neuen Fileservern wird an der

[1] das entsprechende Sekundärmedium muss zuerst gesucht werden, geholt werden und dann in die Laufwerke eingeführt werden

[2] die angeforderten Daten sind innerhalb von Millisekunden verfügbar

[3] In den Nachtstunden wird das zentrale Backup (siehe 1.6.2) erstellt und die gespeicherteten Datenbestände gewartet

Anpassung der Fileserver-Software und an der Lösung der administrativen Aufgaben gearbeitet. Sind diese abgeschlossen und die Daten eines einzelnen Departments übertragen, wird mit der Umstellung dessen begonnen.

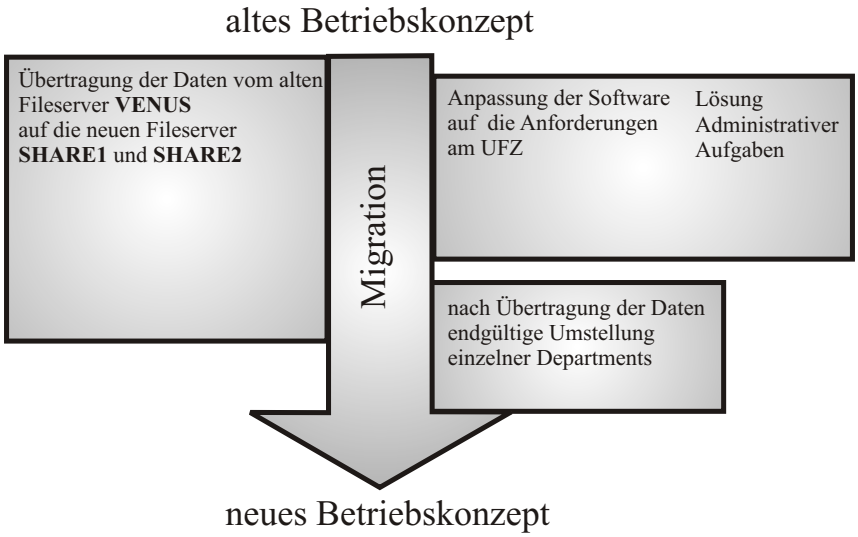

Abbildung 3.1: Ablauf der Migration

3.2 Konfiguration und Anpassung der Software

Der erste Teil dieses Abschnittes erläutert die Konfiguration und Anpassung an die UFZ-Gegebenheiten auf der Server-Seite. Im zweiten Anschnitt werden die Anpassungen auf Seiten der Clients beschrieben.

3.2.1 Die Server-Seite

Konfiguriert wird der Samba-Fileserver, wie schon in Abschnitt 2.2.4 beschrieben, über die zentrale Konfigurationsdatei smb.conf. Im Anhang 6.2 sind die vollständigen Konfigurationsdateien der beiden neuen Fileserver aufgeführt.

Der globale Teil

Der erste Teil (`[global]`) enthält Festlegungen, welche für alle einzurichtenden Freigaben und für den Samba-Fileserver-Dienst gelten. Im folgenden wird anhand der originalen smb.conf-Datei die Bedeutung der einzelnen Optionen im globalen Teil erläutert. Bei beiden

neuen Fileservern ist der globale Teil der smb.conf-Datei identisch. Kommentare innerhalb der smb.conf-Datei werden jeweils mit einem # am Zeilenanfang eingeleitet.

```
[global]
#Top
message command = /usr/bin/mail -s 'message from %f on %m' root < %s; rm %s
server string = Gruppen Service Leipzig
netbios name = share1
os level = 2
local master = no
```

Mit der ersten Option werden sogenannte WinPopup-Meldungen, die an den Fileserver geschickt werden, an eine E-Mail-Adresse umgeleitet. In diesem Fall wird eine E-Mail an den Systemadministrator *root* geschickt. WinPopup-Meldungen sind eine Entwicklung von Microsoft. Das sind Nachrichten, welche vom Betriebssystem verschickt werden, um den Systemadministrator mittels eines Informations-Fensters (auf dem Microsoft-Windows-Desktop) auf bestimmte Ereignisse hinzuweisen (z.B. Fehler beim Starten eines Server-Dienstes).

Die zweite Option legt einen allgemeinen Namen für den Servers fest. Unter diesem ist er für die Benutzer leichter zu erkennen. Dieser Name besitzt keine Relevanz für die Identifizierung innerhalb der Netzwerk-Protokolle.

Mit den folgenden drei Optionen werden NetBIOS Einstellungen konfiguriert. Wie in Abschnitt 1.4.2 beschrieben, setzt Samba auf das Microsoft-NetBIOS-Protokoll auf. Der NetBIOS-Name, `netbios name = `, identifiziert den Rechner, im Gegensatz zu dem mit der Option `server string = ` angegebenen Namen, innerhalb des Netzwerkes eindeutig. Alle Rechner im Netzwerk, welche mit dem NetBIOS-Protokoll arbeiten, besitzen einen sogenannten os-level[4]. Dieser hat je nach Version des Betriebssystems einen Wert[5] zwischen 0 und 255. Allgemein gilt: Je neuer das Betriebssystem desto höher ist dieser Wert.

Der Rechner mit dem Betriebssystem, welches innerhalb des Netzwerkes den höchsten os-level besitzt, sammelt, verwaltet und gibt Auskunft über die NetBIOS-Namen der anderen Computer (NetBIOS-Namensdienst)[Len02]. Dieser Computer ist somit der *local master* (Option `local master = `). Im obigen Auszug wird mit `local master = no` und mit dem niedrigen Wert `os level =2` verhindert, daß der Samba-Fileserver diese Rolle übernimmt. Diese Rolle übernimmt der weiter unten, mit der Option `wins server = 141.65.128.161` angegebene, speziell für NetBIOS-Namensdienste konfigurierte Rechner [Len02].

[4] dt.: Betriebssystem Grad

[5] Nur Microsoft Betriebssysteme besitzen diesen Wert. Ausnahme ist Samba, welches einen Windows-Server simuliert und deshalb einen solchen os-level-Wert besitzt. Typische Werte für den os-level sind: Windows 95/98:1, Windows NT: 16, Windows NT Server: 32

```
#logging
log file = /var/log/smb.log
log level = 1

#Nur diese Rechner sind für eine Verbindung zuzulassen
hosts allow = 141.65.

#Printer -->off
load printers = no
```

Die beiden ersten Optionen im nächsten Abschnitt beschreiben das Protokollieren (Loging) von Ereignissen[6], die während des Betriebes von Samba auftreten. Zuerst wird die zentrale Log-Datei angegeben, sowie deren Position im Dateisystem des Fileservers. Die zweite Option `log level =` gibt an in welchem Umfang Ereignisse in der Log-Datei protokolliert werden. Werte von 0 bis 3 sind möglich. Wird der Wert drei angegeben werden sämtliche Aktionen bis auf NetBIOS-Protokoll-Ebene in der Log-Datei aufgezeichnet. Der Wert 0 bedeutet, daß nur allgemeine Statusinformationen und schwere Fehler protokolliert werden. Der gewählte Wert 1 bietet erweiterte Statusinformationen und ist damit für den Testbetrieb der neuen Fileserver gut geeignet.

Die Option `hosts allow =` hat als Wert eine IP-Adresse oder einen IP-Adressen-Bereich. Alle Rechner, welche eine IP-Adresse besitzen die sich mit der angegebenen IP-Adresse oder dem angegebenen IP-Adressen-Bereich deckt, dürfen die Ressourcen des Samba-Fileservers nutzen[7]. Der hier gewählte Wert besteht aber nur aus einer zweistelligen IP-Adresse (normal vier Stellen IPV4 [Mil99]). Diese Angabe bezeichnet einen IP-Adress-Bereich. Alle Rechner, welche eine IP-Adresse mit den ersten beiden Stellen `141.65.X.X` besitzen[8], dürfen mit dem Fileserver Daten austauschen. Mit dieser Festlegung wird sichergestellt, daß nur Rechner aus dem UFZ die Ressourcen der Fileserver nutzen.

Die letzte Option `load printers =` gibt an, ob Samba sich zusätzlich als Netzwerkdruck-Server anbietet (siehe 2.2.4). Da hier aber ausschließlich die Fileserver-Funktion benötigt wird, wurde diese Option mit dem Wert `no` versehen.

```
#Authentification
workgroup = LEIPZIG
security = domain
password server = 141.65.128.161
encrypt passwords = yes

#Nameservice
wins server = 141.65.128.161
```

[6] zum Beispiel: Erfolgreicher Verbindungsaufbau mit einem Benutzer, Authentifizierung eines Benutzers ist fehlgeschlagen usw. [u.a99]

[7] Unabhängig vom Benutzer, welcher diesen Rechner benutzt

[8] Alle Rechner am UFZ besitzen diese ersten beiden genannten Stellen in ihrer IP-Adresse

```
#Character set
character set = ISO8859-1

#connection recycling
deadtime = 15
```

Der erste Teil, welcher mit `Authentification` bezeichnet ist, wird in Abschnitt Administration/Authentifizierung (siehe 3.3) erläutert. Die Option `wins server =` wurde weiter oben in diesem Abschnitt schon kurz erwähnt. Diese verweist auf den Rechner (über die IP-Adresse), welcher alle NetBIOS-Namen der im Netzwerk befindlichen Rechner sammelt, verwaltet und auf Verlangen Auskunft über diese gibt (NetBIOS-Namensdienst).

Mit `character set =` wird der aktive Zeichensatz für die auf dem Fileserver zu speichernden Datei- und Verzeichnisnamen festgelegt . Wichtig ist diese Option für die korrekte Interpretation sprachlicher Sonderzeichen, wie zum Beispiel in der deutschen Sprache das ä, ü oder ö. Hier wurde der Westeuropäische Zeichensatz nach ISO8859-1 Norm festgelegt.

Samba startet bei jeder erfolgreichen Verbindung mit einem Benutzer einen zusätzlichen Prozess, welcher diese Verbindung verwaltet und bedient (siehe 2.2.4). Die letzte Option, `deadtime = 15` , sorgt nach einer festgelegten Zeit von 15 Minuten ohne eine Aktivität (in Bezug auf Dateitransfer-Vorgänge) für eine automatische Trennung der Verbindung (Client-Server Verbindung) zum Benutzer. Trotz einer erfolgten Trennung bemerkt der Benutzer keine Zeitverzögerung bei dem erneuten Versuch Dateien zu transferieren. Eine neuen Verbindung wird automatisch von Samba aufgebaut. Diese Vorgehensweise dient dazu die Anzahl der inaktiven Samba-Prozesse gering zu halten und somit die Ressourcen (Rechenkapazität, Arbeitsspeicher) des Fileservers zu schonen.

Der Freigaben-Teil

Bereitzustellen sind die vier Freigaben: die UFZ-All-Freigabe, die Programme-Freigabe, die Home-Freigabe und die Gruppen-Freigabe. Diese Freigaben werden jeweils in der Samba-Konfigurationsdatei auf dem jeweiligen Server eingerichtet (Aufteilung siehe Tabelle 2.4). Die ersten beiden stellen keine besonderen Anforderungen dar und werden ähnlich dem in Abschnitt 2.2.4 vorgestellten smb.conf Beispiel, dort Freigabe `[programme]`, eingerichtet.

Die Programme-Freigabe

```
Ausschnitt aus der smb.conf von SHARE2

[programme]
comment = Programmverzeichnis
path = /groups/programme
```

```
read only = no
write list = freymond hanke
```

Mit dem vorliegenden Ausschnitt wird die Programme-Freigabe im Verzeichnis `/groups/programme` auf Fileserver SHARE2 eingerichtet. Der Schreibzugriff ist erlaubt, `read only = no`, allerdings nur für die beiden Benutzer mit dem Namen `freymond` und `hanke`. Diese Einschränkung wird durch die Option `write list =` erreicht, welche als Wert eine Liste der Benutzer erhält, die Schreibrecht auf die Freigabe erhalten. Mit `comment =` wird ein Kommentar vereinbart, welcher die Freigabe genauer umschreibt.

Zusammenfassung (Programme-Freigabe): Die Benutzer können nach Bedarf von dieser Freigabe ihre Programme und Anwendungen auf ihre lokalen Rechner kopieren und installieren. Schreiben dürfen nur die Benutzer, welche für die Verwaltung dieser Freigabe zuständig sind.

UFZ-ALL-Freigabe

```
Ausschnitt aus der smb.conf von SHARE1

[ufzall]
path= /spare/ufzall/
read only = no
create mask = 0777
directory mask = 0777
```

Dieser Abschnitt legt die UFZ-All-Freigabe auf Fileserver SHARE1 im Verzeichnis `/spare/ufzall` an. Mit `read only = no` wird der Schreibzugriff ermöglicht. Die letzten vier Optionen behandeln die Art und Weise mit welchen Rechten die Dateien auf dem Server abgelegt werden[9]

Die hier aufgeführten Zahlen stellen eine Vereinfachung des Linux/Unix-Rechtesystems dar. Wie in Abschnitt 1.3.2 dargestellt, existieren drei Rechte-Attribute für eine Datei oder ein Verzeichnis: Lesbar, Beschreibbar, Ausführbar. Jeder dieser Attribute wird mit einer Zahl folgendermaßen vereinfacht dargestellt: Lesbar = 4, Beschreibbr = 2, Ausführbar = 1 und keine Rechte = 0. Eine Kombination (Addition) der Zahlen ist möglich. Ein Beispiel ist die Zahl 7. Sie steht sowohl für das Recht zum Lesen, Schreiben und Ausführen. Die Zahl 5 hingegen steht für das Recht zum Lesen und Ausführen. Weiterhin gelten diese Rechte-Attribute jeweils für die drei Benutzergruppen: Eigentümer, Gruppemitgliedschaft der/des Datei/Verzeichnisses und restliche Welt.

Mit der Option `create mask = 0777` wird jede Datei, die von einem Benutzer auf dem Server erstellt oder kopiert wird, mit folgenden Rechten versehen: der Eigentümer (Benutzer),

[9] unter ablegen wird hier das erstellen oder das kopieren einer Datei auf den File-Server verstanden

die Gruppe in der die Datei Mitglied ist und der Rest der Benutzer darf Lesen, Schreiben und Ausführen (Zahl:7). Die Option `directory mask = 0777` legt die gleichen Rechte für Verzeichnisse fest.

Zusammenfassung (UFZ-All-Freigabe): Jeder UFZ-Benutzer darf alle auf der UFZ-All-Freigabe befindlichen Daten lesen, schreiben oder ausführen. Somit wird die UFZ-All-Freigabe als zentrale Daten-Austausch-Freigabe am UFZ genutzt.

Home-Freigabe

Mit der Einrichtung dieser Freigabe waren die folgenden Anforderungen verbunden:

- Automatisches Anlegen von Homeverzeichnissen, falls diese noch nicht existieren
- Einrichten der Speicherplatzbeschränkung (Quota)
- Protokollierung jedes Benutzer-Zugriffes auf seine Home-Freigabe

Zusätzlich wird sichergestellt, daß nur der jeweilige Benutzer die Lese und Schreibrechte auf seiner Home-Freigabe besitzt. Zur Erfüllung dieser Anforderungen wurde die Home-Freigabe wie folgt eingerichtet.

```
Ausschnitt aus der smb.conf von SHARE1

[home]
comment = Homelaufwerke
root preexec = /usr/bin/perl /etc/samba/userinit.pl %U %G %M
path = /user/samba/%U
read only = no
create mask = 0700
directory mask = 0700
valid users = @rz @nl @ballr @uoe @alok @oekus @ana @san @grundwa
```

Mit der Festlegung der Rechte-Masken (`create mask = 0700` und `directory mask = 0700`) ist es nur dem Besitzer erlaubt auf seiner Home-Freigabe zu lesen, schreiben oder Anwendungen auszuführen. Mit der Option `root preexec =` wird vor dem Verbindungsaufbau eines Benutzers mit dem Fileserver das als Wert angegebene Skript[10], hier `/etc/samba/userinit.pl`[11], gestartet. Durch diese Möglichkeit, Skripte

[10] Als Skript wird eine Folge von Befehlen bezeichnet, welche in einer Skript-Datei aufgeführt sind und bei deren Ausführung durch einen Skript-Interpreter hintereinander gestartet werden. Es existieren dazu sogenannte Skript-Sprachen, welche zusätzlich komfortable Funktionen wie Schleifen, bedingte Anweisungen, Variabeln usw. zur Verfügung stellen.

[11] Hier handelt es sich hier um ein Perl-Skript, aufgeführt in Anhang 6.3.1, Perl ist eine Skript-Sprache sowie ein Skript-Interpreter

vor der eigentlichen Verbindung des Benutzers mit der Freigabe auszuführen, ist die Realisierung von administrativen Aufgaben, wie den genannten Anforderungen für die Home-Freigabe, möglich.

Die hinter der zweiten Option (`root preexec = ...`) und dritten Option (`path = ...`) stehenden Zeichen (`%U,%G,%M`) sind Samba-Variabeln (auch Samba-Makros genannt)[u.a99]. Diese Variabeln werden während des Betriebes von Samba durch entsprechende Werte ersetzt. Verbindet sich zum Beispiel der Benutzer mit dem Namen `paul` mit seiner Home-Freigabe, so wird der Pfad dieser (Option `path =`) auf `/user/samba/paul` gesetzt. Somit wird jeder Benutzer automatisch zu seinem eigenen Home-Verzeichnis verbunden. Die Variable `%U` steht für den Benutzernamen, `%G` für den Gruppennamen und `%M` für den Namen des Client-Computers von welchem der Benutzer eine Verbindung zum Samba-Fileserver aufbaut. Weitere Variabeln und deren Bedeutung werden in [u.a99] erläutert.

Dem Perl-Skript `userinit.pl` (zweite Option `root preexec =`) werden drei Variabeln (der Benutzername `%U`, der Gruppenname in welcher er Mitglied ist `%G`, der Name seines Computers `%M`) mit dessen Ausführung übergeben. Mit Hilfe dieser Angaben werden die Anforderungen an die Home-Freigabe mittels des Perl-Skripts realisiert. Dessen Arbeitsweise ist in Abbildung 3.2 als Programmablaufplan zu sehen. Der Programm-Code des Skripts ist im Anhang 6.3.1 aufgeführt. Eine genaue Erläuterung der eingesetzten Skript-Befehle sowie der Skript-Syntax ist in [Dit03] zu finden.

Das Skript prüft zuerst, anhand der vorhandenen Home-Verzeichnisse auf Fileserver SHARE1, ob der Benutzer ein solches bereits besitzt. Ist das der Fall wird dies in der Protokoll-Datei unter Angabe des Namens des Benutzers, dessen Rechners und der aktuellen Zeit vermerkt. Existiert noch kein Verzeichnis, so wird dieses angelegt und mit dem festgelegten Benutzer-Quota (5 GByte, siehe Abschnitt 2.2.2) eingerichtet. Anschließend werden die Rechte für dieses Verzeichnis gesetzt. Nur der Eigentümer (Benutzer) darf lesen, schreiben und ausführen. Auch der Vorgang des Anlegens wird mit dem Namen des Benutzers, dem des Rechners und der aktuellen Zeit protokolliert. Als letzter Schritt wird überprüft, ob das neue Home-Verzeichnis angelegt wurde. Der Erfolg oder Misserfolg wird ebenso protokolliert.

Die letzte Option `valid users =` , im Ausschnitt der Home-Freigabe, gibt die Benutzer oder Gruppen an, welche generell Zugriff auf diese Freigabe erhalten. Einzelne Benutzer werden mit ihren Namen dort eingetragen. Gruppen bekommen ein `@` als Erkennung vor den Gruppennamen gesetzt. Benutzer oder Gruppen, welche nicht angegeben werden, erhalten keinen Zugriff auf diese Freigabe.

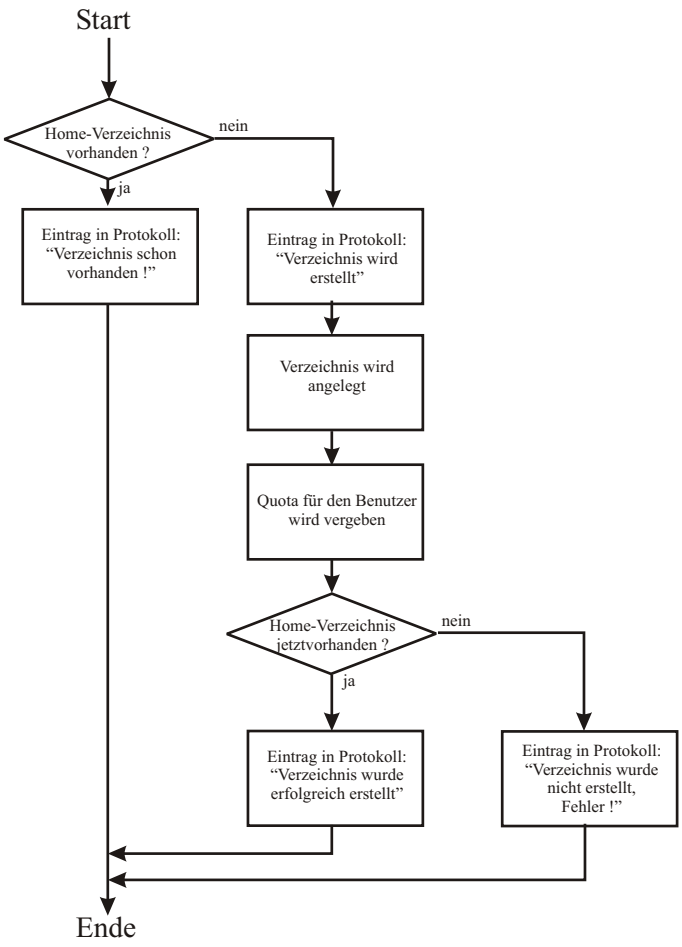

Abbildung 3.2: Programmablaufplan (PAP) des Skriptes für die Home-Freigabe

Zusammenfassung Home-Freigabe: Jeder Benutzer besitzt eine Home-Freigabe, auf welcher nur er lesen, schreiben und Programme ausführen darf. Existiert noch kein Verzeichnis für die Home-Freigabe des Benutzers, so wird dies mittels des Perl-Skripts `userinit.pl` angelegt und mit dem gültigen Quota versehen. Jeder Anmelde-Vorgang mit der Home-Freigabe sowie das Anlegen eines Home-Verzeichnisses wird protokolliert. Mittels der Protokoll-Datei können Fehlfunktionen und Störungen und deren mögliche Ursachen erkannt werden.

Gruppen-Freigabe

Die Einrichtung der Gruppen-Freigabe stellte die umfangreichsten Anforderungen aller Freigaben. Der Zugriff auf diese Freigabe soll allen Mitgliedern einer Gruppe, am UFZ ein sogenanntes Department, möglich sein. Innerhalb der Gruppe soll diese genutzt werden, um Daten untereinander auf Department-Ebene auszutauschen. Die Realisierung einer Gruppen-Freigabe

unter diesen einfachen Bedingungen würde keine Schwierigkeiten bereiten. Ähnlich wie bei der Home-Freigabe würde der Pfad, welcher den Ort des Gruppenverzeichnisses angibt, mit der Option `path = /groups/%G` festgelegt werden. Durch die Nutzung der Samba-Variable `%G` würde der Benutzer jeweils mit dem seiner Gruppenzugehörigkeit entsprechenden Verzeichnis verbunden.

Am UFZ sind aber Benutzer vorhanden, die in mehreren Departments arbeiten und somit Mitglied in mehr als einer Gruppe sind. Entsprechend benötigen diese Benutzer Zugriff auf mehrere Gruppen-Freigaben. Die Realisierung dieses Umstandes ist nur durch den Einsatz der Skript-Funktion (Option `root preexec =`) von Samba zu lösen. Zusätzlich ergeben sich noch die folgenden Anforderungen:

- Automatisches Anlegen von neuen Gruppen-Verzeichnissen
- Protokollierung jedes Benutzer-Zugriffes auf die Gruppenlaufwerke

Für die Realisierung der Problematik, Mitgliedschaft von Benutzern in mehreren Gruppen, wird eine Eigenschaft von Linux/Unix-Betriebssystemen genutzt. Sogenannte Verknüpfungen (engl.: Links)[Sie99] ermöglichen es an einem beliebigen Ort im Dateisystem, für Dateien oder Verzeichnisse, Pseudonyme zu erstellen. Mit diesen Pseudonymen wird wie mit normalen Dateien oder Verzeichnissen gearbeitet. Alle Änderungen an diesen Pseudonymen wirken sich direkt auf die originalen Dateien oder Verzeichnisse aus. Gleichzeitig besitzen diese die gleichen Rechte wie die mit ihnen verknüpften Originale.

Nach dem Kopieren aller Gruppen-Verzeichnisse auf Fileserver SHARE2 wird für jeden Benutzer zusätzlich auf diesem ein separates eigenes Verzeichnis erstellt (im weiteren als Gruppen-Home-Verzeichnis bezeichnet). Mittels eines Perl-Skriptes (wird bei jedem Verbindungsaufbau mit Fileserver ausgeführt, Option `root preexec =`) werden, entsprechend der Gruppen-Mitgliedschaft eines Benutzers, Verknüpfungen von den betreffenden Gruppenverzeichnissen zu seinem Gruppen-Home-Verzeichnis angelegt. Dieser Vorgang wird in Abbildung 3.3 am Beispiel des Benutzers `paul` dargestellt.

Dargestellt wird ein Ausschnitt aus dem Verzeichnis-Baum von SHARE2. Der Benutzer `paul` ist Mitglied von zwei Gruppen, `alok` und `uoe`. Somit benötigt er zwei Verknüpfungen (in Abbildung 3.3 mit gestrichelten Pfeillinien dargestellt). Diese verbinden die jeweiligen Gruppenverzeichnisse, `\groups\samba\alok` und `\groups\samba\uoe`, mit seinem Gruppen-Home-Verzeichnis, `\groups\links\paul`. Mit diesen beiden Verknüpfungen in seinem Gruppen-Home-Verzeichnis kann der Benutzer `paul` auf den Gruppen-Freigaben arbeiten.

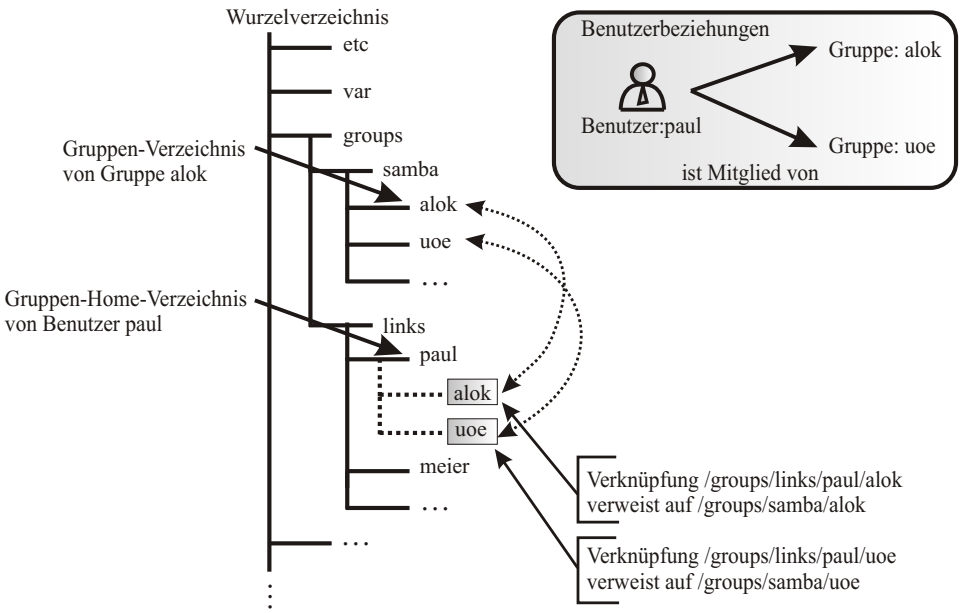

Abbildung 3.3: Beispiel für Verknüpfungen für die Gruppen-Freigabe

Diese Lösung führt zu der folgenden Freigaben-Definition. Mit der Pfadangabe `path = /groups/links/%U` wird der Benutzer mit seinem Gruppen-Home-Verzeichnis verbunden. In diesem befinden sich, je nach Gruppenmitgliedschaft, die Verknüpfungen, erstellt durch das Perl-Skript `groups10.pl`, zu seinen Gruppen. Er besitzt Lese-/Schreibrechte und kann Programme ausführen, während die Gruppenmitglieder Dateien nur lesen (4) können. Um ein Wechseln in ein Verzeichnis für die Gruppenmitglieder zu ermöglichen, muß es lesbar und ausführbar sein (4+1=5). Benutzer, die nicht Gruppenmitglieder sind, haben keinen Zugriff (0). Mit `valid users` werden auch hier die Benutzer-Gruppen festgelegt, welche Zugriff auf die Gruppen-Freigabe erhalten.

```
Ausschnitt aus der smb.conf von SHARE2

[gruppen]
comment = Gruppenlaufwerke
root preexec =/usr/bin/perl /etc/samba/groups10.pl %U %G %M
path= /groups/links/%U
read only = no
create mask = 0740
directory mask = 0750
valid users = @rz @nl @ballr @uoe @alok @oekus @ana @san @grundwa
```

Die Realisierung der Anforderungen und die Problematik der Mitgliedschaft von Benutzern in mehreren Gruppen wird vom Perl-Skript `groups10.pl` verwirklicht. Es unterteilt sich in zwei Hauptteile:

1. – Automatisches Anlegen von neuen Gruppenverzeichnissen
   – Festlegung der Rechte-Attribute (Gruppenverzeichnisse)

2 – Anlegen von Verknüpfungen (falls diese noch nicht existieren oder neue Gruppenmitgliedschaften hinzugekommen sind), je nach Gruppenmitgliedschaft des sich anmeldenden Benutzers
   – Überprüfen vorhandener Verknüpfungen auf Gültigkeit im Vergleich mit der momentanen Gruppenmitgliedschaft des Benutzers
   – gegebenfalls Löschen von nicht mehr berechtigten Verknüpfungen (Benutzer ist nicht mehr Mitglied in einer Gruppe)

Erster Hauptteil: Das Perl-Skript `groups10.pl` ist im Anhang 6.3.1 aufgeführt. Zum Ablauf des Skriptes ist in Abbildung 3.4 der Programmablaufplan des ersten Hauptteiles dargestellt. Alle Ereignisse und Fehler werden in einer Protokoll-Datei (`group.log`) mit Zeitpunkt, Name des Benutzers und Name des Benutzer-Rechners protokolliert.

Der erste Schritt ist die Sammlung der benötigten Informationen. Über das Abfragen des im UFZ existierenden LDAP-Verzeichnisdienstes (siehe 1.5) werden alle Gruppen, welche am UFZ existieren, ermittelt. In der Datei `validgroups` sind die Namen der gültigen Gruppen vermerkt. Gültig sind nur Benutzergruppen. Keine gültigen Gruppen sind zum Beispiel: Gruppen welche für Anwendungen und deren Funktionen nötig sind, administrative Gruppen oder Test-Benutzergruppen. Die Datei `validgroups` ist eine durch Leerzeichen getrennte sequentielle Liste:

```
Inhalt der Datei "validgroups":
rz nl ballr uoe alok oekus ana san grundwa
```

Zusätzlich werden die bereits zu einen früheren Zeitpunkt angelegten Gruppenverzeichnisse (bereits vorhandenen) ermittelt. Mit den gesammelten Informationen wird jetzt gearbeitet.

Anhand der Liste aller existierenden Gruppen (über den LDAP-Verzeichnisdienst ermittelt) wird jede dieser Gruppen auf ihr Vorhandensein als Gruppenverzeichnis überprüft. Ist dies der Fall, wird mit der nächsten Gruppe fortgefahren. Ist diese noch nicht vorhanden wird überprüft ob sie in der Liste (`validgroups`) der gültigen Gruppen aufgeführt ist. Fällt dieser Test positiv aus, wird das Gruppenverzeichnis neu angelegt und mit den Rechten der jeweiligen Gruppen versehen. Ist das Ergebnis negativ, wird mit der Überprüfung der nächsten Gruppe fortgefahren. Dieser Ablauf wird vollzogen bis alle über den LDAP-Verzeichnisdienst ermittelten Gruppen überprüft wurden. Existieren Gruppenverzeichnisse, die nicht mehr im LDAP-Verzeichnis als Benutzer-Gruppen aufgeführt sind (Benutzer-Gruppe wurde zum Beispiel aufgelöst), so werden diese nicht gelöscht. Das Löschen wird administrativ manuell vorgenommen.

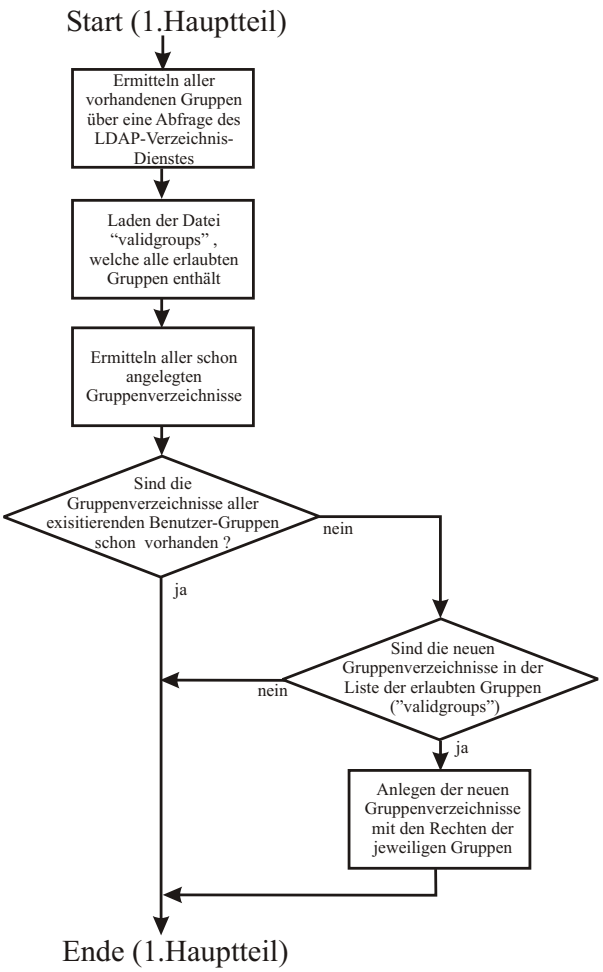

Abbildung 3.4: 1. Hauptteil, Programmablaufplan des Skriptes der Gruppen-Freigabe

Zweiter Hauptteil In Abbildung 3.5 ist der Programmablaufplan des zweiten Hauptteiles zu sehen. Zuerst wird festgestellt, ob das Gruppen-Home-Verzeichnis (siehe Abbildung 3.3) des sich anmeldenden Benutzers vorhanden ist. Fällt dieser Test negativ aus, wird es angelegt. Über eine LDAP-Anfrage werden alle aktuellen Gruppenmitgliedschaften dieses Benutzers ermittelt und die jeweiligen Verknüpfungen in seinem Gruppen-Home-Verzeichnis angelegt (Beispiel siehe Abbildung 3.3, für den Benutzer `paul`). Anschließend werden alle im Gruppen-Home-Verzeichnisses vorhandenen Verknüpfungen mit den über die LDAP-Anfrage ermittelten aktuellen Gruppenmitgliedschaften verglichen. Diese Überprüfung dient der bei einem Ausschluss des Benutzers aus einer Benutzergruppe (Benutzer ist nicht mehr Mitglied in einer Gruppe) nötigen Löschung der Verknüpfung auf die jeweilige Gruppe. Somit erhält der Benutzer nur Zugriff auf die Gruppen (Gruppenverzeichnisse), in welchen er zu diesem Zeitpunkt Mitglied ist.

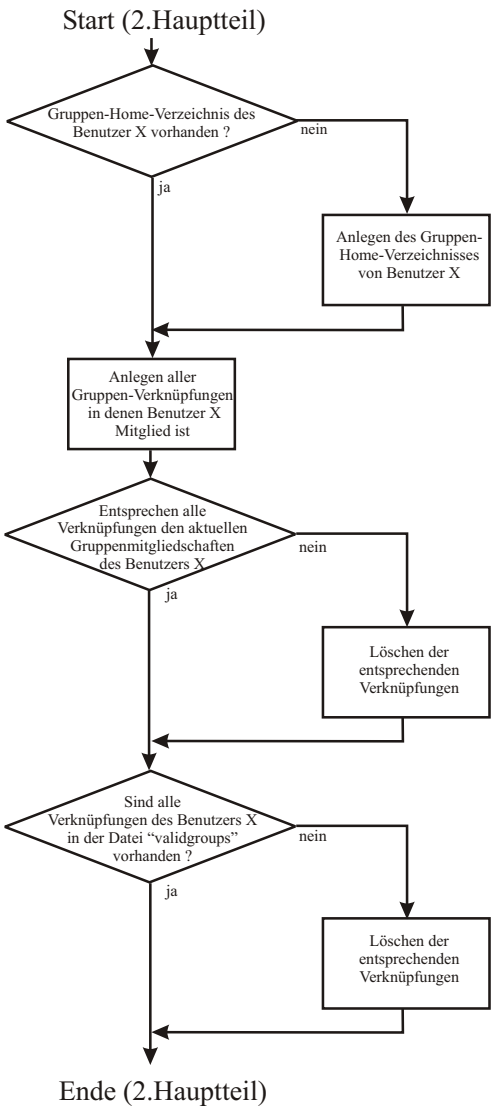

Abbildung 3.5: 2. Hauptteil, Programmablaufplan des Skriptes der Gruppen-Freigabe

Im letzten Schritt findet mittels der `validgroups`-Datei eine weitere Überprüfung statt. Nur Verknüpfungen im Gruppen-Home-Verzeichnis des sich anmeldenden Benutzers, welche in der `validgroups`-Datei aufgelistet sind (gültige Benutzergruppen), bleiben bestehen. Alle anderen werden gelöscht.

Zusammenfassung Gruppen-Freigabe: Jeder Benutzer erhält Zugriff auf die Verzeichnisse der Gruppen in denen er aktuell Mitglied ist. Diese Funktionalität wird durch das Ausführen des Perl-Skript `groups10.pl` (siehe Anhang 6.3.1) bei jedem Anmeldevorgang an die Gruppen-Freigabe gewährleistet. Neue Gruppenverzeichnisse werden entsprechend vorher angelegt. Alle Ereignisse und Aktionen werden protokolliert.

3.2.2 Die Client-Seite

Auf der Client-Seite werden am UFZ drei Betriebssysteme von Microsoft eingesetzt[12] (Windows NT, Windows 2000, Windows XP). Jedes dieser Betriebssysteme kann während seiner Startphase Skripte auszuführen. Diese Skripte können sowohl lokal, auf dem jeweiligen Rechner, wie auch auf einem zentralen Rechner im Netzwerk abgelegt sein. Am UFZ ist das der Rechner mit dem Namen `adcenter`. Er ist auch für die Authentifizierung der Benutzer zuständig (siehe 3.3.1). Die zentrale Ablage hat den Vorteil, daß neue Skripte und Änderungen an diesen nur einmal (statt an alle betreffenden Rechner) verteilt werden müssen.

Mit dem eingesetzten Skript `ldaplogin.vbs` werden die auf den neuen Fileservern SHARE1 und SHARE2 konfigurierten Freigaben (siehe 3.2.1) auf den Clients automatisch, während deren Startphase (nach Einschalten des Computers), eingerichtet. Unter Einrichten wird hier der Aufbau der Verbindung mit der jeweiligen Freigabe und der Verknüpfung mit einem Laufwerksbuchstaben verstanden. Die Freigabe stellt sich anschließend als ein virtueller Datenträger dar. Somit ist der Benutzer in der Lage auf diesem virtuellen Datenträger wie auf einem lokalen Datenträger zu arbeiten. Ein Beispiel ist in Abbildung 1.14 auf Seite 30 zu sehen. Dort wurde die Freigabe `homes` als virtueller Datenträger mit dem Laufwerksbuchstaben `L` verknüpft.

In der Phase der Umstellung (Migration) vom alten Betriebskonzept (siehe 2.1) zum neuen Betriebskonzept (siehe 2.2), gab es einige besonderen Umstände zu beachten. Ein neuer Mitarbeiter sollte von Beginn an Zugriff auf die neuen Freigaben (auf den neuen Fileservern) erhalten, auch wenn das jeweilige Department noch nicht umgestellt war. Um solche Benutzer von den anderen zu unterscheiden, wurden diese Mitglied in der für diesen Zweck eingerichteten administrativen Gruppe `newgroup`.

Das Transferieren der Home-Freigaben vom alten Fileserver VENUS, im Vergleich mit den anderen zu übertragenden Freigaben, nahm wegen deren höheren Datenvolumens circa das 20-fache an Zeit in Anspruch. Aus diesem Grund erhielten die Benutzer, deren Home-Verzeichnis noch nicht auf die neuen Fileserver übertragen wurden, Zugriff auf ihre Home-Freigabe auf dem alten Fileserver VENUS. Erst nach der kompletten Übertragung der Daten beziehen die Benutzer dann ihre Daten ausschließlich von den neuen Fileservern. Um diese Benutzer zu erkennen wird eine administrative Gruppe `movegroup` eingerichtet. Benutzer, welche in dieser Gruppe Mitglied sind, wurden bereits auf die neuen Fileserver-Freigaben umgestellt. Alle anderen beziehen nur noch die Home-Freigabe vom alten VENUS-Fileserver[13]. Nach dem Abschluss der Umstellung auf das neue Betriebskonzept (siehe 2.2) am gesamten UFZ werden

[12] Es werden auch UNIX-Betriebssysteme eingesetzt (SUN OS von SUN Microsystems), welche ihre Daten aber über NFS siehe 1.4.1 und einen anderen Fileserver austauschen

[13] Bevor die Umstellung eines Departments beginnen kann, muss somit mindestens die Gruppen-Daten des Departments auf die neuen Fileserver übertragen worden sein (die Daten der Programme-Freigabe und der UFZ-ALL-Freigabe (NEU) sind auf den neuen Fileservern schon vorhanden)

diese administrativen Gruppen wieder entfernt. Das Skript beschränkt sich dann nur noch auf das automatische Einrichten der Laufwerks-Verknüpfungen mit den neuen Freigaben.

Das Skript ist in der von Microsoft für seine eigenen Betriebssysteme empfohlenen Skript-Sprache Visual-Basic-Script[Wel00] entwickelt. Im Anhang 6.3.2 ist das Skript aufgeführt. Es folgt die Beschreibung der Arbeitsweise des Skriptes (`ldaplogin.vbs`), welches in Abbildung 3.6 als Programmablaufplan dargestellt ist.

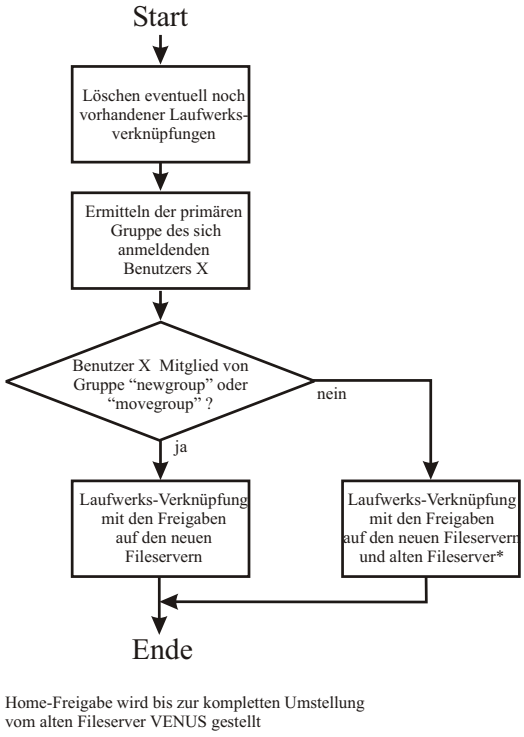

Abbildung 3.6: Programmablaufplan des Client-Skriptes

Benutzer von Microsoft-Windows-Betriebssystemen verfügen über die Möglichkeit Freigaben-Laufwerks-Verknüpfungen manuell einzurichten. Diese werden anchließend bei jedem Betriebssystemstart automatisch wieder angelegt. Um eine Überschneidung dieser manuell angelegten Freigaben-Laufwerks-Verknüpfungen mit denen des vom zentralen Client-Skript angelegten zu verhindern, werden diese im ersten Schritt gelöscht. Das Einrichten der Freigaben-Laufwerks-Verknüpfungen über das zentrale Client-Skript führt zu einem weiteren Vorteil. Ändern sich die Pfade[14] zu den Freigaben ist nur eine Korrektur im zentralen Client-Skript nötig. Dies ist zum Beispiel der Fall wenn einer der Fileserver ausfällt und der andere Fileserver

[14] mit Pfaden sind hier zum Beispiel die Namen der Fileserver oder der Freigaben gemeint

dessen Aufgaben als zeitlicher Ersatz übernimmt (Pfadänderung). Daher werden die Freigabe-Laufwerks-Verknüpfungen bei jedem Start des Betriebssystems (nach dem Start des Rechners) über das Client-Skript mit den jeweils aktuellen Pfaden neu eingerichtet.

Im nächsten Schritt werden über die Abfrage des zentralen UFZ-LDAP-Verzeichnis-Dienstes[15] die Gruppen ermittelt, in welchen der Benutzer Mitglied ist. Ist der Benutzer Mitglied einer der administrativen Gruppen `movegroup` oder `newgroup`, werden die virtuellen Laufwerke komplett mit den Freigaben auf den neuen Fileservern verknüpft (Benutzer wurde vollständig umgestellt oder ist neu am UFZ). Ist der Benutzer kein Mitglied einer der beiden administrativen Gruppen, so wird sein Home-Laufwerk noch mit der Home-Freigabe auf dem alten Fileserver VENUS verknüpft. Alle anderen Freigaben bezieht er aber schon von den neuen Fileservern. Über eingeblendete Windows-Informations-Fenster wird der Benutzer auf die erfolgreiche oder nicht erfolgreiche Verknüpfung der virtuellen Laufwerke mit den einzelnen Freigaben hingewiesen.

3.3 Administration

Dieser Abschnitt befasst sich mit den verschiedenen administrativen Aufgaben, die sowohl in der Phase der Umstellung wie im späteren Regel-Betrieb der neuen Fileserver von Bedeutung sind.

3.3.1 Authentifizierung

Mit der Authentifizierung wird sichergestellt, daß nur die Benutzer einen Zugriff auf die Ressourcen[16] erhalten, welche die dazu nötigen Voraussetzungen besitzen. Gültige Voraussetzungen sind: Mitarbeiter am UFZ, als EDV-Benutzer registriert, Mitglied in einem Department und ein gültiger Benutzername mit zugehörigem geheimen Passwort.

Über seinen Benutzernamen und das geheime Passwort authentifiziert sich der Benutzer gegenüber dem Authentifizierungsserver (siehe 1.5). Am UFZ sind zwei Authentifizierungsserver vorhanden. Der erste ist der schon genannte zentrale LDAP-Verzeichnis-Dienst, welcher sich auf dem Rechner mit dem Namen `hera` befindet. Dieser wurde mit der Software SUN One Directory Server von SUN Microsystems realisiert. Über diesen werden alle Benutzer sowie deren Attribute (inklusive der entsprechenden Passwörter) verwaltet und bei Bedarf neu eingerichtet. Der zweite Authentifizierungsserver mit dem Namen `adcenter` stellt einen Active Directory-Verzeichnisdienst von Microsoft zur Verfügung. Gegenüber diesem authentifizieren

[15] Verzeichnisdienste siehe 1.5 (Benutzerverwaltung)

[16] mit Ressourcen sind hier die Freigaben und die darauf enthaltenen Daten gemeint

sich die Benutzer, welche sich mit einem Microsoft Windows Betriebssystem anmelden[17]. Dieser zweite Authentifizierungsserver ist nötig, da sich die Microsoft Windows Clients momentan nur über ein Microsoft Authentifizierungsserver erfolgreich anmelden.

Der Active Directory Server erhält seine Informationen, in Bezug auf die vorhanden Benutzer, deren Gruppenmitgliedschaften und entsprechender Attribute, in Abständen von fünf Minuten vom LDAP-Verzeichnis-Dienst. Somit bleibt der Datenbestand auf dem Active Directory Server immer aktuell.

Meldet sich ein Benutzer mit seinem Benutzernamen und seinem geheimen Passwort an die neuen Fileserver an (was automatisch durch das Client-Skript geschieht), so stellen diese eine Anfrage an den Active Directory Server über die Korrektheit der Angaben. Sind diese richtig, erhält der Benutzer Zugriff auf die entsprechenden Ressourcen (Freigaben) der Fileserver. Diese Vorgehensweise entspricht der in der Abbildung 1.18 in Abschnitt 1.5 dargestellten Authentifizierungsmethode.

In Abschnitt 3.2.1 wurde der globale Teil der zentralen Konfigurationsdatei smb.conf des Samba-Fileservers behandelt. Dabei wurde die Erläuterung des Authentifizierungs-Ausschnittes in diesen Abschnitt des Kapitels gelegt. Es folgt die Erläuterung der Konfiguration der beiden neuen Fileserver in Bezug zu der Authentifizierung der Benutzer gegenüber dem Active Directory-Server.

```
Authentifizierungs-Ausschnitte der smb.conf von SHARE1 und SHARE2

#Authentifation
workgroup = LEIPZIG
security = domain
password server = 141.65.128.161
encrypt passwords = yes
```

Mit der ersten Option `workgroup = LEIPZIG` wird die Arbeitsgruppe oder Domäne[18] festgelegt, in welcher sich alle Rechner einschließlich der Server befinden [Kup00]. An diese Arbeitsgruppe/Domäne melden sich die Microsoft-Windows-Clients am Leipziger UFZ an. Der Name `LEIPZIG` wurde hier wegen des Standortes des UFZ gewählt. Es existieren daneben noch andere Standorte, wie zum Beispiel Halle und Magdeburg, mit deren jeweiligen Arbeitsgruppen/Domänen. Mit der Option `security = domain` authentifiziert der Samba-Fileserver die Benutzer gegen den mit der Option `password server = 141.65.128.161` angegebenen Authentifizierungsserver.

[17] mit anmelden wird hier der Zugriff auf den Client selbst, sowie auf die Ressourcen (Freigaben) der Fileserver verstanden

[18] Domänen und Arbeitsgruppen sind sogenannte Organisationseinheiten in welchen Benutzer, Gruppen, Rechner usw. organisiert werden. Diese Organisationseinheiten, auch OU genannt, sind eine Erfindung von Microsoft [Kup00]

Die angegebene IP-Adresse bezieht sich auf den weiter oben in diesem Abschnitt genannten Active-Directory-Server `adcenter`. Mit der letzten Option `encrypt passwords = yes` werden die Passwörter zwischen den Fileservern, den Clients und dem Authentifizierungsserver verschlüsselt und somit sicher übertragen.

3.3.2 Quota-Skript

Im neuen Betriebskonzept wurden für die Benutzer und Gruppen Speicherplatzbeschränkungen auf den neuen Fileservern festgelegt (siehe 2.2.2). Für jeden einzelnen Benutzer stehen, nach diesen Festlegungen, 5 Gigabyte an Speicherplatz in seinem Home-Verzeichnis zur Verfügung. Die Festlegung dieses Quotas wird beim Einrichten des Home-Verzeichnisses mit dem in Abschnitt 3.2 (Home-Freigabe) beschriebenen Perl-Skript `userinit.pl` (Quellcode siehe 6.3.1) über die Quota-Software vollzogen.

Für die Speicherplatzbeschränkung der Gruppenfreigaben wurden 10 Gigabyte + 1 Gigabyte je Gruppenmitglied, jedoch maximal 40 Gigabyte, festgelegt. Da es zu Schwankungen in der Anzahl der Mitglieder der einzelnen Gruppen kommt (Benutzer werden Mitglied oder wechseln in eine andere Gruppe), wird der jeweils zustehende Speicherplatz einmal pro Tag mittels des `cron`[19]-Programmes dynamisch mit der jeweils aktuellen Mitgliederanzahl über das Perl-Skript `quota.pl` berechnet und entsprechend angeglichen.

Das Perl-Skript `quota.pl` ist im Anhang 6.3.3 aufgeführt. Von jeder Gruppe wird über die Abfrage des UFZ-LDAP-Verzeichnis-Dienstes die Anzahl der Mitglieder bestimmt. Anschließend wird der jeweils zustehende Speicherplatz berechnet und mit der Quota-Software entsprechend festgelegt. Auf die Darstellung des Programmablaufplanes (PAP) wurde wegen der geringen Komplexität des Skriptes verzichtet.

3.3.3 Synchronisierung der Datenbestände (Backup)

Die gegenseitige Synchronisierung der Datenbestände wurde im neuen Betriebskonzept festgelegt. Daher werden die auf SHARE1 gespeicherten Benutzer-Daten (Home-Freigabe) auf SHARE2 kopiert und durch Synchronisation auf dem aktuellen Stand gehalten. In gleicher Weise wird der Datenbestand der Gruppen (Gruppen-Freigabe) von SHARE2 auf SHARE1 kopiert und ständig aktualisiert.

Zur Realisierung dieser Aufgabe wurde wieder die Software *rsync* eingesetzt (siehe 3.1). Diese kopiert jeweils nur die Daten, welche sich seit der letzten Synchronisation geändert haben. Dieser Vorgang wird jeweils jeden Tag in den Nachtstunden automatisch durchgeführt. Die

[19] Programm welches in festlegbaren Zeitabständen (monatlich, wöchentlich, täglich, stündlich) frei bestimmbare Aktionen (Programme oder Skripte) ausführt

Vorhaltung des jeweils tagesaktuellen Gesamt-Datenbestandes, auf beiden neuen Fileservern, erlaubt es in Zukunft die Ausfallsicherheit weiter zu erhöhen. Fällt einer der beiden Fileserver aus, so wird der andere dessen Fileserver-Dienste übernehmen. Dazu nutzt er den jeweiligen, auf ihm gesicherten, Datenbestand des ausgefallenen Fileservers. Diese Funktionalität wird erst zu einem späterem Zeitpunkt realisiert.

Im alten Betriebskonzept befinden sich Veritas-NetBackup-Server (siehe 1.6.2) und Fileserver auf einem Rechner (VENUS). Mit dem neuen Betriebskonzept wurden die verschiedenen Dienste örtlich neu verteilt. Der zentrale Backup-Service (NetBackup) bleibt auf dem Rechner VENUS, während die Fileserver-Dienste auf die beiden neuen Server SHARE1 und SHARE2 gelegt wurden. Das Backup der Fileserver-Daten wird im neuen Betriebskonzept folgendermaßen realisiert. Auf beiden neuen Fileservern ist ein NetBackup-Client installiert. In den Nachtstunden werden jeden Tag die Daten der neuen Fileserver über das NetBackup auf die Sekundärmedien der Roboter im Rechenzentrum gesichert.

3.4 Ablauf einer Department-Umstellung

Für die Umstellung eines Departments auf das neuen Betriebskonzept wurde ein Ablaufplan entwickelt. Dieser ist in Tabelle 3.1 wiedergegeben.

Arbeits-Schritt	
1.	Die Gruppen-Daten des jeweiligen Departments müssen komplett auf den neuen Fileserver (SHARE2) übertragen worden sein.
2.	Mit der Übertragung der Benutzer-Home-Laufwerke auf SHARE1 wird begonnen.
3.	Benutzer nach Benutzer erhalten Zugriff auf die neuen Fileserver (das Start-Skript wird durch das in Abschnitt 3.2.2 erläuterte zentrale Client-Skript ersetzt).
4.	Ist das Home-Laufwerk eines Benutzers auf den neuen Fileserver (SHARE1) übertragen, wird er als Mitglied der Gruppe `movegroup` hinzugefügt, und erhält somit vollständigen Zugriff auf die neuen Fileserver siehe auch Abschnitt 3.2.2
5.	Die Umstellung des Departments ist abgeschlossen, wenn alle Home-Laufwerke der Benutzer übertragen sind. Alle Benutzer sind Mitglieder in der Gruppe `movegroup` und sind auf die neue Fileserverstruktur am UFZ umgestellt.

Tabelle 3.1: Ablaufplan zur Department-Umstellung auf das neuen Betriebskonzept

Es folgen Erläuterungen zu einigen Schritten des Ablaufplanes. Vor dem ersten Schritt sind die neuen Server auf ihre Aufgabe als Fileserver bereits konfiguriert. Darunter wird die Einrichtung des Betriebssystems sowie der Fileserversoftware SAMBA verstanden (siehe Kapitel 3). Um Zugriff auf die neuen Fileserver zu erlangen, dritter Schritt, müssen die Benutzer ihre Rechner (Betriebssysteme) neu starten. Dies ist notwendig damit das zentrale Client-Skript (siehe 3.2.2) ausgeführt wird. Zu beachten ist, daß es in dieser Zeit, in welcher einige der Benutzer (umgestellte) bereits die neue Gruppenfreigabe und andere noch die alte Gruppenfreigabe auf dem Fileserver VENUS nutzen, zwischen diesen Freigaben zu einer Asynchronität[20] der gespeicherten Daten kommt. Daher wird stündlich mit dem *rsync*-Tool (siehe 3.1) bis zur kompletten Umstellung aller Benutzer des Departments, die beiden Gruppen-Freigaben synchronisiert. Bevor sein Home-Laufwerk, in Schritt vier, komplett übertragen ist, erhält der Benutzer noch Zugriff auf dieses über den alten Fileserver VENUS (siehe 3.2.2). Ist die Übertragung der Home-Freigabe abgeschlossen und der Benutzer wurde der administrativen Gruppe `movegroup` hinzugefügt, muß dieser seinen Rechner (Betriebssystem) wieder neu starten, um das Start-Skript erneut auszuführen. Erst dann erhält er kompletten Zugriff auf alle Freigaben auf den neuen Fileservern und ist somit auf die neue Fileserverstruktur und das neue Betriebskonzept umgestellt!

[20] umgestellte Benutzer schreiben Daten auf die neue Gruppenfreigabe, während noch nicht umgestellte Benutzer diese neuen Daten auf der alten Gruppenfreigabe nicht vorfinden können

Kapitel 4

Performance-Tests

Während der Umstellung auf das neuen Betriebskonzept (siehe 2.2) wurde die Leistungsfähigkeit der neuen Fileserver im Vergleich mit dem alten Fileserver VENUS getestet. Dieses Kapitel befasst sich mit den nötigen Voraussetzungen (siehe 4.1) für einen Test unter Bedingungen, die sich möglichst nahe an der Realität eines Fileserverbetriebs orientieren. Anschließend ist in Abschnitt 4.2 der eigentliche Test der Server Gegenstand der Betrachtung. Zum Abschluss werden dessen Ergebnisse (siehe 4.3) diskutiert.

4.1 Voraussetzungen

Die typischen Aufgaben eines Fileservers sind das Lesen und Schreiben von Daten aller Art (siehe 1.1). Dabei wechseln sich beide Zugriffsarten ständig ab oder finden gleichzeitig statt. Je mehr Benutzer auf die Freigaben des Fileservers gleichzeitig zugreifen, desto mehr Lese- und Schreibzugriffe muss dieser bewältigen. Durch eine steigende Anzahl der gleichzeitigen Zugriffe erhöht sich die Belastung der zentralen Recheneinheiten. Sind die Recheneinheiten voll ausgelastet und es treffen weitere Zugriffs-Aufträge ein, müssen diese zeitlich verschoben oder verworfen werden. Das Ergebnis dieser Überbelastung sind stagnierende oder sinkende Transferraten sowie steigenden Antwortzeiten zwischen dem Server und den Clients. Eine weitere Beeinflussung der Leistung stellen auch die Verarbeitungsgeschwindigkeiten der übrigen Teile[1] eines Rechners dar. Daher müssen alle Teile eines Rechners auf einander, in Bezug auf ihre Leistungsfähigkeit, abgestimmt und ausgewogen sein.

Ein Test, welcher die Leistungsfähigkeit eines Fileservers ermittelt, muß dessen Arbeitsweise in der Realität nachahmen. Der von Ziff-Davis entwickelte NetBench-Benchmark ist in der Lage die realen Anforderungen an einen Fileserver zu simulieren. Er simuliert, mit bis zu 60

[1] zum Beispiel: maximale Transferraten der Verbindungen (Busse) zwischen der zentralen Recheneinheit und dem Arbeitsspeicher, den Datenspeichern (Festplatten-Arrays) und der Netzwerk-Schnittstelle

Client-Systemen, Zugriffe auf einen Fileserver. Gemessen wird der Datendurchsatz in MBit pro Sekunde und der Mittelwert der Antwortzeiten des Servers auf eine Client-Anfrage. Auf jedem Client-Rechner wird eine NetBench-Client installiert. Über einen Controller-Rechner (NetBench-Controller) werden die Clients gesteuert. Diese führen dann Belastungstests auf dem Fileserver durch und melden die Ergebnisse dem Controller-Rechner, welcher diese dann auswertet. Der Fileserver muß lediglich eine SMB-Freigabe (siehe 1.4.2) zur Verfügung stellen. Über diese führen die Clients die Lese- und Schreibzugriffe für den Test durch. Alle anderen Eigenschaften des Fileservers, wie dessen konkrete Implementierung, dessen Rechner-Architektur sowie dessen Betriebssystem, spielen keine Rolle.

4.2 Der Test

Für die Bewertung der Leistungsfähigkeit der neuen Fileserver, gegenüber dem alten Fileserver, bot der Schulungsraum des UFZ gute Voraussetzungen. In diesem sind 13 Rechner vorhanden, auf welche ein uneingeschränkter Zugriff möglich ist. Die technischen Daten der einzelnen Test-Rechner sind in Tabelle 4.1 aufgeführt.

Hardware	
Prozessoren	Intel Pentium III 650 Mhz
Arbeitsspeicher	256 MByte
Netzwerkschnittstelle	Fast-Ethernet (100MBit/s) 3Com905C-TX
Software	
Betriebssystem	Windows XP Profesional
NetBench	Version 7.0.3
Test-Suite	`dm_nb.tst`

Tabelle 4.1: Technische Daten der Test-Rechner

Auf zwölf Rechnern wurde der NetBench-Client und auf einem Rechner der NetBench-Controller installiert. Alle Rechner sind untereinander über ein Netzwerk mit einer Bandbreite von 100 MBit/s verbunden. Der Schulungsraum selbst ist über eine 100 MBit/s Netzwerk-Leitung mit den Fileservern im Rechenzentrum verbunden[2]. Diese Bandbreite beschränkt die möglichen Ergebnisse des Tests, im Bezug auf die Gesamt-Transferrate, auf 100 MBit/s. Liegen die Ergebnisse der Test bei dieser Marke (95 bis 99 MBit/s), so stellt das Netzwerk zwischen den Clients und den Fileservern einen Engpass dar. In diesem Fall wären Aussagen über die wirkliche Leistungsfähigkeit der Fileserver nicht möglich.

[2] Innerhalb des Rechenzentrums sind die Fileserver über GBit-Ethernet (1 GBit/s) miteinander verbunden (UFZ-Backbone, siehe 2.1). Maßgebend für die maximale Transferrate zwischen den Clients im Schulungsraum und den Fileservern ist aber die geringste benutzte Bandbreite (100 MBit/s).

Um die Ergebnisse der Tests nicht zu verfälschen, wurden diese nach 19.00 Uhr durchgeführt. Erst nach dieser Uhrzeit konnte sichergestellt werden, daß keine Belastung der Fileserver durch normalen Benutzer-Betrieb mehr vorlag. NetBench verfügt über sogenannte Test-Suites in welchen der genaue Ablauf eines Tests vorgegeben wird (Lesezugriffe, Schreibzugriffe oder beides gleichzeitig). Der Benutzer ist in der Lage eigene Test-Suites zu erstellen oder diese entsprechend seinen Anforderungen anzupassen. In diesem Test wurde die Standard Test-Suite `dm_nb.tst` verwendet. Diese simuliert, wie in 4.1 angegeben, die typische Arbeitslast eines Fileservers.

Jeder der beiden Gesamt-Tests (zwei Fileserver) nahm in etwa 20 bis 40 Minuten an Zeit in Anspruch. Der somit über einen langen Zeitraum laufende und in sich geschlossene Test führte zu Ergebnissen mit einer hohen Aussagekraft. Aus diesem Grund wurde hier auf das mehrfache Ausführen gleicher Tests zur Steigerung der Aussagekraft verzichtet. Der Test wurde mit jeweils einem, vier, acht und zwölf gleichzeitig anfragenden Clients durchgeführt. Durch diese Vorgehensweise soll der Einfluß auf die Transferraten und Antwortzeiten bei steigender Client-Zahl aufgezeigt werden. Die technische Daten der beiden Fileserver sind in Tabelle 2.1 auf Seite 49 (VENUS) und in Tabelle 2.3 auf Seite 55 (SHARE2) aufgeführt.

4.3 Die Ergebnisse

In den Tabellen 4.2 (VENUS) und 4.3 (SHARE2) sind die Ergebnisse der Fileserver-Tests aufgeführt. In der ersten Spalte ist die Anzahl der gleichzeitig den Fileserver belastenden Clients angegeben. Der Wert der zweiten Spalte wurde aus der erreichten Gesamt-Menge der transportierten Daten, zwischen den Clients und dem Fileserver, über die Gesamt-Laufzeit des Tests gebildet[3]. Dieser wird in MBit/s angegeben. In der dritten Spalte ist die Antwortzeit des Servers, in Millisekunden, auf die Anfragen der Clients aufgeführt. Dieser Wert stellt einen von NetBench, aus den einzelnen Ergebnissen aller an den Teil-Tests beteiligten Clients, berechneten arithmetischen Mittelwert dar.

Um die Ergebnisse der Fileserver-Tests besser vergleichen zu können wurde ein Diagramm erstellt. In Abbildung 4.1 sind sowohl die erreichte Transferrate und die dazu ermittelte Antwortzeit beider Fileserver zu sehen. Die Abszissenachse steht für die Anzahl der Clients, die linke Ordinatenachse für die Transferrate in MBit/s und die rechte Ordinatenachse für die Antwortzeit in Millisekunden. Mit dieser Darstellung werden Aussagen über das Verhältnis zwischen den Transferraten und den jeweiligen Antwortzeiten der Fileserver getroffen.

[3] Gesamtmenge der transportierten Daten dividiert durch die Laufzeit des Tests (in Sekunden) ergibt die Menge der Daten welche pro Sekunde transferiert wurden

VENUS		
Anzahl der Clients	Transferrate in MBit/s	Antwortzeit des Servers in ms
1	3,420	2,251
4	7,140	7,510
8	8,670	12,070
12	9,418	16,128

Tabelle 4.2: Testergebnisse des Fileserver VENUS

SHARE2		
Anzahl der Clients	Transferrate in MBit/s	Antwortzeit des Servers in ms
1	5,165	0,715
4	20,267	0,779
8	38,108	1,003
12	53,647	1,235

Tabelle 4.3: Testergebnisse des Fileserver SHARE2

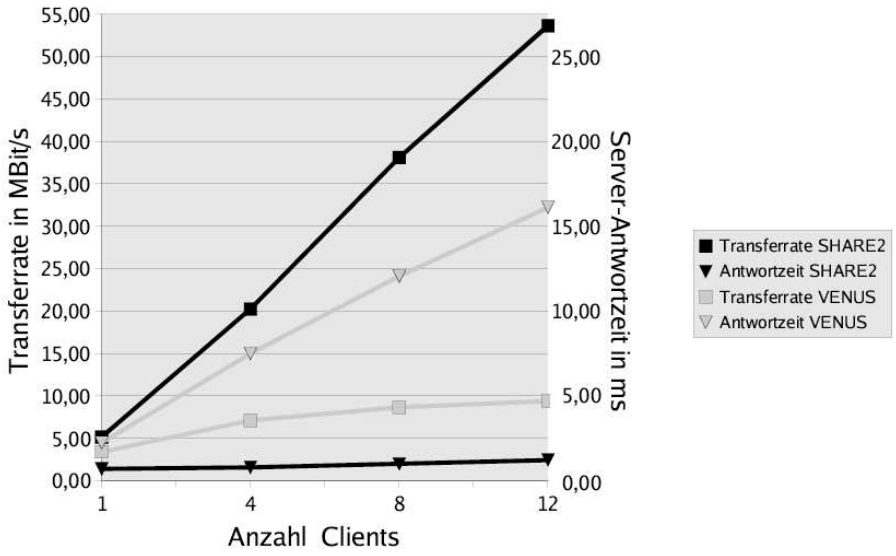

Abbildung 4.1: Ergebnisse der Fileserver Performance-Tests

Bei keinem der Tests wurde die maximal mögliche Transferrate (100 MBit/s) des Netzwerkes, zwischen den Clients und dem Fileserver, erreicht. Somit hatte dieser potentielle Engpass keinen Einfluss auf die Ergebnisse. Deutlich geht aus den Diagramm die Überlegenheit, in Bezug auf

die Transferrate, des neuen Fileservers im Gegensatz zum alten Fileserver hervor. Beide Server liefern bei der Belastung durch nur einen Client in etwa die gleiche Menge an Daten. Aber schon bei vier gleichzeitig anfragenden Clients liefert der neuen Fileserver SHARE2 im Vergleich zum alten Fileservers VENUS in etwa die dreifache Menge an Daten aus. Bei zunehmender Anzahl der Clients steigt die Transferrate des neuen Fileservers nahezu linear an, während diese beim alten Fileserver stagniert und bei etwa 10 MBit/s (12 Clients) ihr Maximum erreicht.

Die Antwortzeiten der Fileserver geben Auskunft über deren Belastung durch die Anfragen der einzelnen Clients. Je höher die Antwortzeiten sind, desto mehr ist der Fileserver mit dem Beantworten und Ausliefern der Daten beschäftigt. Wie in Abbildung 4.1 zu sehen, hält der neue Fileserver SHARE2 eine Antwortzeit von etwa einer Millisekunde im Laufe des gesamten Tests aufrecht, während diese beim alten Fileserver von anfangs zwei Millisekunden (1 Client) auf letztlich 16 Millisekunden (12 Clients) fast linear ansteigen. Daraus lässt sich die folgende Aussage ableiten. Für den neuen Fileserver stellen 12 gleichzeitig anfragende Clients keine Belastung dar (linearer Anstieg der Transferrate und gleichbleibende Antwortzeit). Der alte Fileserver stagniert bei der Transferrate schon in der Phase zwischen einem und vier Clients bei nahezu linear ansteigenden Antwortzeiten. Dieses beobachtete Verhalten der beiden gemessenen Größen (Transferrate und Antwortzeit) bedeutet eine hohe Belastung des alten Fileservers bei zunehmender Client-Anzahl. Somit ist der alter Fileserver schon mit den gleichzeitigen Anfragen von vier bis acht Clients überlastet! Da im normalen Regel-Betrieb von einer wesentlich höheren Anzahl parallel anfragender Clients (bis zu 30) ausgegangen werden muß, könnte dieses Verhalten ein Grund für die angesprochenen Extremsituationen (siehe 2.1) sein, welche zum Absturz des gesamten Fileservers (VENUS) führten!

Das Ergebnis des NetBench-Tests belegt die Überlegenheit, in Bezug auf die Transferraten und Antwortzeiten, der neuen Fileserver im neuen Betriebskonzept. Weiterhin ist zu bemerken, daß sich alle Anfragen, statt wie im alten Betriebskonzept auf einen Fileserver (VENUS), auf die zwei neuen Fileserver (SHARE1 und SHARE2) verteilen und somit deren Durchschnitts-Belastung im Regelbetrieb geringer ausfällt.

Kapitel 5

Zusammenfassung und Ausblick

Der Aufbau einer auf dem Betriebssystem Linux basierenden Fileserverinfrastruktur am Umweltforschungszentrum Leipzig war das Ziel dieser Arbeit. Für dessen Realisierung wurde im Kapitel 2 ein neues Betriebskonzept entwickelt. Das Betriebssystem wurde auf die Linux-Distribution von Red Hat festgelegt. Diese Entscheidung mußte getroffen werden, da diese Linux-Distribution als einzige vom Hersteller der neuen Fileserver (Dell) offiziell unterstützt wird. Mit dieser offiziellen Unterstützung ist die Versorgung von an das Betriebssystem (Red Hat Linux) gebundenen Hardware-Treibern (RAID-Controller, Netzwerkschnittstellen) garantiert.

Als Speicherstrategie wurde die lokale Speicherung gewählt (siehe 1.2). Auf den neuen Festplatten-Arrays wurden RAID-Level 5 Arrays mit jeweils einer Hot-Spare Festplatte eingerichtet. Das gewählte RAID-Level stellt den besten Kompromiss zwischen Speicherplatzbedarf (physisch vorhandener und tatsächlich nutzbarer) und physischer Ausfallsicherheit dar (siehe 1.6.1). Fällt eine der Festplatten des RAID-Arrays aus wird diese sofort durch die immer betriebsbereite Hot-Spare Festplatte ersetzt.

Für das Dateisystem der RAID-Arrays wurde das von Silicon Graphics entwickelte XFS gewählt. Diese Entscheidung war das Ergebnis einer Reihe ausführlicher Tests. Die Testprogramme waren Tiobench und Postmark. Tiobench testet das Dateisystem mit einer festlegbaren Anzahl von gleichzeitig auf eine Datei zugreifenden Threads. Durch stetiges erhöhen der Threadanzahl (Anstieg der Belastung) wird die Skalierbarkeit des Dateisystems getestet. Die Dateisysteme ext3 und ReiserFS zeigten teilweise starke Einbrüche in den Testverläufen (siehe Abbildung 2.6). Dieses Verhalten deutet auf interne Probleme bei der Implementierung oder der Organisation/Verwaltung der beiden Dateisysteme hin. XFS erreichte die höchsten Transferraten und skalierte mit ansteigender Threadanzahl am besten. Der Postmark-Test führt Transaktionen (schreiben, lesen, löschen und anlegen von Dateien) auf eine festlegbare Anzahl von Dateien aus. Die erhaltenen Ergebnisse (siehe 2.2.3) waren enttäuschend. Alle Testverläufe

zeigten das gleiche Bild und ließen keine Schlüsse auf das Performance-Verhalten der getesteten Dateisysteme zu. Somit wurden ausschließlich die Ergebnisse des Tiobench-Tests zur Wahl des Dateisystems genutzt.

Für die Realisierung eines Fileservers wird eine Fileserversoftware benötigt (siehe 2.2.4). Die Wahl fiel auf den freien Fileserver SAMBA von Andrew Tridgell. Dieser stellt eine Server-Implementierung des SMB/CIFS-Protokolls auf Linux dar. Damit ist es möglich Client-Rechnern mit Microsoft-Windows-Betriebssystemen Speicherressourcen über das Netzwerk zur Verfügung zu stellen. Samba stellt einen freien Ersatz für die kommerziellen Microsoft Server-Betriebssysteme, in Bezug auf deren Fileserverfunktionalität, dar. Die Konfiguration des Samba-Fileservers wird komfortabel über nur eine zentrale Datei realisiert.

In Kapitel 3 war die Migration vom alten Betriebskonzept zum neuen Betriebskonzept das zentrale Thema. Der gewählte Fileserver Samba wurde an die organisatorischen Gegebenheiten (siehe 3.2.1) des Umweltforschungszentrums angepasst. Dazu gehörte die Bereitstellung der Home- und der Gruppen-Freigaben. Die dazu nötige Konfiguration der Samba-Fileserversoftware wurde ausführlich behandelt. Die Problematik der Mitgliedschaft von Benutzern in mehreren Gruppen wurde mittels eines Skriptes gelöst. Mit diesem Skript erhält der Benutzer Zugriff auf die Gruppen-Verzeichnisse der Gruppen in welchen er aktuell Mitglied ist. Weiterhin wurde die UFZ-ALL-Freigabe und die Programm-Freigabe angelegt. Das zentral abgelegte Client-Skript (siehe 3.2.2) verbindet die Benutzer-Rechner (Clients) automatisch mit den Ressourcen (Freigaben) der neuen Fileserver. Ein weiteres Skript berechnet täglich die aktuelle Anzahl der Mitglieder eines Departments. Anschließend richtet es die zentral festgelegte Speicherplatzbeschränkung (Quota) für die jeweilige Gruppen-Freigabe (entsprechend der ermittelten Anzahl) ein. Der organisatorische Ablauf einer Departmentumstellung vom alten Betriebskonzept zum neuen Betriebskonzept wurde in Abschnitt 3.4 erarbeitet und erläutert.

Der Vergleich des alten Fileservers mit einem der neuen Fileserver, in Bezug auf Belastbarkeit und Performance, war Gegenstand des Kapitels 4. Hierzu wurde das Testprogramm NetBench von Ziff-Davis benutzt. Dieses ist in der Lage die realen Anforderungen an einen Fileserver zu simulieren. Das Ergebnis war eindeutig. Während der alte Fileserver mit zunehmender Belastung immer weniger Daten ausliefert, zeigt sich der neue Fileserver davon unbeeindruckt. Die Transferrate steigt in etwa linear mit zunehmender Belastung (steigende Anzahl von Client-Rechnern) an (neuer Fileserver). Die parallel gemessenen konstant niedrigen Antwortzeiten bescheinigen dem neuen Fileserver ein hohes Leistungspotential.

Somit wurden alle Fragen beantwortet und alle Anforderungen erfüllt. Die neue Fileserverinfrastruktur ist erweiterbar (steigender Speicherbedarf) und für die zukünftig steigenden Belastungen (steigende Benutzerzahlen) geeignet. Die eingesetzte freie Software (Linux, XFS, Samba) wurde auf die organisatorischen Gegebenheiten des Umweltforschungszentrums angepasst und konfiguriert. Weiterhin stellt die neue Fileserverinfrastruktur eine qualitative Verbesserung

der angebotenen Dienste, in Bezug auf die Ausfall- und Datensicherheit, dar. Die erarbeiteten Lösungen sind nicht auf eine alleinige Anwendung am Umweltforschungszentrum Leipzig beschränkt. Sie können auch in anderen Einrichtungen (Unternehmen, öffentlicher Dienst usw.) eingesetzt werden.

Die Synchronisierung der Datenbestände beider neuer Fileserver wird bereits täglich vollzogen (siehe 3.3.3). Die Entwicklung eines Notfall-Konzeptes ist für die Zukunft vorgesehen. Fällt einer der beiden Fileserver aus, würde der andere Fileserver dessen Arbeit übernehmen. Der Benutzer könnte mit den angebotenen Ressourcen (Freigaben) wie gewohnt weiter arbeiten. Das Management der neuen Fileserver mittels des Simple Network Management Protokolls (SNMP) und die Entwicklung einer entsprechenden Managementanwendung ist vorgesehen. Der Aufbau eines Storage Area Networks (Speichernetzwerk siehe 1.2) am UFZ ist geplant. Diese Speicherstrategie erlaubt eine hohe Flexibilität beim Ausbau der vorhandenen Speicherkapazität und eine noch höhere Ausfallsicherheit. Mit dem Aufbau der neuen Fileserverinfrastruktur sind somit neue interessante Aufgaben entstanden, die als Themen für weitergehende Arbeiten genutzt werden können.

Kapitel 6

Anhang

6.1 Ergebnisse der Dateisystem-Tests

Tiobench-Ergebnisse

In den Tabellen 6.1 und 6.2 sind die Ergebnisse der Tiobench-Tests (siehe 2.2.3) aufgeführt. Zu den Ergebnissen (Mittelwert, Median) zählen die Transferraten in MByte pro Sekunde und die jeweilige Belastung der Recheneinheiten in Prozent (in Klammern). In der Tabelle 6.3 sind die mittleren quadratischen Abweichungen aller Transferraten-Ergebnisse aufgeführt. Der erste Wert ist die berechnete mittlere quadratische Abweichung. In Klammern ist die jeweilige zum Mittelwert (Ergebnisse aus Tabelle 6.1 und 6.2) berechnete prozentuale Abweichung aufgeführt.

	ReiserFS		ext3		XFS	
Thread-anzahl	Mittelwert kByte/s (%)	Median kByte/s (%)	Mittelwert kByte/s (%)	Median kByte/s (%)	Mittelwert kByte/s (%)	Median kByte/s (%)
sequentielles Lesen						
4	67,47 (26,85)	69,31 (27,37)	25,02 (8,44)	25,16 (8,13)	102,53 (43,79)	103,74 (43,17)
8	97,09 (29,14)	96,69 (28,80)	26,17 (11,80)	26,27 (11,57)	120,57 (65,40)	121,36 (65,67)
16	96,69 (30,52)	98,76 (30,17)	26,59 (15,09)	26,30 (14,52)	127,40 (70,84)	128,44 (71,17)
32	104,14 (43,51)	104,68 (43,36)	26,13 (15,52)	25,95 (15,53)	128,90 (78,98)	128,19 (77,92)
64	103,42 (44,89)	102,67 (45,00)	26,51 (15,86)	26,40 (15,43)	130,88 (79,96)	130,11 (79,43)
128	102,57 (47,92)	101,58 (47,55)	26,16 (15,98)	26,11 (15,85)	133,56 (92,05)	133,96 (91,16)
zufälliges Lesen						
4	5,51 (3,81)	5,36 (3,75)	3,89 (8,49)	3,86 (8,33)	5,16 (2,37)	4,51 (2,21)
8	8,22 (6,43)	8,26 (6,05)	5,49 (11,76)	5,41 (11,79)	7,90 (3,35)	7,81 (3,61)
16	11,01 (9,68)	11,38 (9,56)	6,80 (15,06)	6,81 (15,06)	10,79 (5,78)	10,73 (5,36)
32	12,90 (14,82)	13,25 (15,10)	7,90 (6,21)	7,81 (6,27)	14,21 (8,59)	14,55 (9,23)
64	16,20 (12,57)	16,69 (12,33)	8,12 (7,55)	8,09 (6,89)	17,48 (12,28)	17,79 (12,22)
128	17,94 (16,56)	17,92 (15,04)	8,35 (8,34)	8,25 (7,55)	17,44 (12,57)	17,32 (11,96)

Tabelle 6.1: TioBench-Ergebnisse Lese-Transferraten und Belastung der Recheneinheiten (in Klammern)

Thread-anzahl	ReiserFS Mittelwert kByte/s (%)	ReiserFS Median kByte/s (%)	ext3 Mittelwert kByte/s (%)	ext3 Median kByte/s (%)	XFS Mittelwert kByte/s (%)	XFS Median kByte/s (%)
sequentielles Schreiben						
4	11,96 (29,98)	12,07 (30,50)	12,20 (32,66)	12,20 (33,24)	12,31 (8,77)	12,32 (8,53)
8	11,67 (29,44)	11,67 (30,13)	12,20 (32,03)	12,19 (32,00)	12,28 (10,94)	12,28 (10,84)
16	11,37 (28,26)	11,33 (28,75)	12,19 (32,05)	12,19 (32,03)	12,25 (11,62)	12,25 (11,56)
32	11,11 (25,54)	11,16 (24,86)	12,18 (32,82)	12,18 (32,76)	12,22 (13,25)	12,22 (13,46)
64	10,75 (12,57)	10,79 (12,33)	12,15 (32,63)	12,17 (32,54)	12,20 (13,58)	12,20 (13,76)
128	10,31 (29,24)	10,35 (30,25)	12,14 (33,37)	12,15 (33,20)	12,16 (15,08)	12,16 (14,86)
zufälliges Schreiben						
4	2,49 (3,35)	2,50 (3,29)	2,12 (3,32)	2,14 (3,28)	2,19 (3,55)	2,16 (3,68)
8	2,54 (3,53)	2,55 (3,54)	2,14 (3,37)	2,16 (3,29)	2,12 (3,53)	2,15 (3,64)
16	2,52 (3,64)	2,53 (3,64)	2,15 (3,41)	2,14 (3,32)	2,14 (3,80)	2,17 (3,81)
32	2,54 (3,72)	2,55 (3,65)	2,14 (3,24)	2,16 (3,31)	2,14 (3,89)	2,13 (3,93)
64	2,56 (4,00)	2,58 (4,00)	2,21 (3,47)	2,22 (3,45)	2,17 (3,50)	2,19 (3,56)
128	2,57 (4,36)	2,55 (4,31)	2,20 (3,63)	2,20 (3,54)	2,16 (4,10)	2,18 (4,27)

Tabelle 6.2: TioBench-Ergebnisse Schreib-Transferraten und Belastung der Recheneinheiten (in Klammern)

Threadanzahl	ReiserFS	ext3	XFS	ReiserFS	ext3	XFS
	sequentielles Lesen			zufälliges Lesen		
4	6,350 (9,412)	0,512 (2,049)	5,760 (5,618)	0.222 (4,037)	0,103 (2,636)	0,375 (7,269)
8	2,125 (2,189)	0,882 (3,373)	2,959 (2,451)	0,146 (2,189)	0,183 (3,342)	0,502 (6,356)
16	1,660 (1,717)	0,823 (3,095)	2,469 (1,937)	0,728 (6,616)	0,336 (4,950)	0,735 (6,813)
32	4,493 (4,219)	1,445 (5,532)	2,070 (1,606)	1,823 (14,135)	0.192 (2,431)	1,189 (8,374)
64	4,040 (3,906)	0,961 (3,525)	2,701 (2,063)	1,019 (6,294)	0,288 (3,550)	1,058 (6,056)
128	3,030 (2,955)	0,034 (1,602)	3,105 (2,358)	0,935 (5,212)	0,226 (2,706)	1,341 (7,693)
	sequentielles Schreiben			zufälliges Schreiben		
4	0,126 (1,059)	0,010 (0,081)	0,010 (0,087)	0,059 (2,391)	0,045 (2,116)	0,026 (1,206)
8	0,079 (0,679)	0,012 (0,096)	0,014 (0,115)	0,014 (0,539)	0,044 (2,084)	0,114 (5,387)
16	0,080 (0,711)	0,025 (0,205)	0,008 (0,068)	0,013 (0,498)	0,043 (1,997)	0,036 (1,670)
32	0,109 (0,988)	0,027 (0,218)	0,013 (0,107)	0,014 (0,561)	0,034 (1,602)	0,039 (1,828)
64	0,131 (1,222)	0,036 (0,284)	0,012 (0,101)	0,038 (1,473)	0,057 (2,582)	0,043 (1,994)
128	0,137 (1,324)	0,024 (0,194)	0,016 (0,129)	0,029 (1,132)	0,025 (1,132)	0,060 (2,826)

Tabelle 6.3: Werte der mittleren quadratische Abweichung (in Klammern als Prozentwert)

Postmark-Ergebnisse

In Tabelle 6.4 sind die Ergebnisse (Mittelwert, Median) der Postmark-Tests (siehe 2.2.3) aufgeführt. Die jeweilige mittlere quadratische Abweichung ist in Tabelle 6.5 aufgeführt. Der erste Wert ist die berechnete mittlere quadratische Abweichung. In Klammern ist die zum jeweiligen Mittelwert (Ergebnisse aus Tabelle 6.4) berechnete prozentuale Abweichung aufgeführt.

Dateien/ Transaktionen	ReiserFS		ext3		XFS	
	Mittelwert	Median	Mittelwert	Median	Mittelwert	Median
Lese-Test (Ergebnisse in kByte/s)						
1000/20 000	11490,00	10640,00	20220,00	21290,00	3070,00	3040,00
20 000/50 000	690,63	692,55	1099,58	1130,00	830,11	814,99
20 000/100 000	736,36	736,35	1158,00	1170,00	945,34	927,58
Schreib-Test (Ergebnisse in kByte/s)						
1000/20 000	12730,00	11770,00	22350,00	23530,00	3390,00	3360,00
20 000/50 000	1274,00	1280,00	2068,00	2140,00	2130,00	2140,00
20 000/100 000	1034,00	1030,00	1666,00	1680,00	1334,00	1300,00
Transaktions-Test (Ergebnisse in Transaktionen/s)						
1000/20 000	3733,00	4000,00	6332,80	6666,00	1041,60	1052,00
20 000/50 000	258,20	259,00	454,40	462,00	445,80	423,00
20 000/100 000	252,80	252,00	416,00	423,00	385,00	375,00

Tabelle 6.4: Postmark-Ergebnisse

Dateien/Transaktionen	ReiserFS	ext3	XFS
Lese-Test			
1000/20 000	1043,000 (6,080)	2132,000 (10,542)	60,000 (1,954)
20 000/50 000	4,367 (0,632)	71,270 (6,482)	34,966 (4,212)
20 000/100 000	2,913 (0,396)	74,473 (6,429)	42,730 (4,520)
Schreib-Test			
1000/20 000	1176,000 (9,239)	2532,000 (10,522)	68,000 (2,004)
20 000/50 000	8,000 (0,628)	149,853 (7,240)	72,660 (4,781)
20 000/100 000	4,899 (0,474)	108,185 (6,494)	73,375 (5,500)
Transaktions-Test			
1000/20 000	326,762 (8,753)	666,400 (10,523)	20,800 (1,997)
20 000/50 000	2,039 (0,789)	28,723 (6,321)	48,910 (10,972)
20 000/100 000	1,166 (0,461)	28,824 (6,929)	21,090 (5,478)

Tabelle 6.5: Werte der mittleren quadratischen Abweichung (in Klammern als Prozentwert)

6.2 Die Samba-Konfigurationsdateien der neuen Fileserver

smb.conf von Fileserver SHARE1

```
[global]
# Top
message command = /usr/bin/mail -s 'message from %f on %m' root < %s; rm %s
server string = Gruppen Service Leipzig
netbios name = share1
os level = 2
local master = no

#logging
log file = /var/log/smb.log
log level = 1

#Nur diese Rechner sind für eine Verbindung zuzulassen
hosts allow = 127.0.0.1 141.65.

#Printer -->off
```

```
load printers = no

#Authentification
workgroup = LEIPZIG
security = domain
password server = 141.65.128.161
encrypt passwords = yes

#Nameservice
wins server =141.65.128.161

#Character set
character set = ISO8859-1

#connection recycling
deadtime = 15

# Shares
[home]
comment = Homelaufwerke
root preexec = /usr/bin/perl /etc/samba/userinit.pl %U %G %M
path = /user/samba/%U
read only = no
create mask = 0700
directory mask = 0700
valid users = @rz @nl @ballr @uoe @alok @oekus @ana @san @grundwa

[ufzall]
path= /spare/ufzall/
read only = no
create mask = 0777
directory mask = 0777
```

smb.conf von Fileserver SHARE2 Der `[global]`-Teil entspricht dem von SHARE1, daher werden hier nur die Freigaben-Definitionen aufgeführt.

```
# Shares
[gruppen]
comment = Gruppenlaufwerke
root preexec =/usr/bin/perl  /etc/samba/groups10.pl %U %G %M
path= /groups/links/%U
read only = no
create mask = 0740
directory mask = 0750
valid users = @rz @nl @ballr @uoe @alok @oekus @ana @san @grundwa

[programme]
comment = Programmverzeichnis
path = /groups/programme
read only = no
write list = freymond hanke
```

6.3 Quellcodes der Scripte

6.3.1 Server-Freigabe-Scripte

Perl-Script *userinit.pl* für die Home-Freigabe:

```
#################################################################
#!/usr/bin/perl -w                                              #
# Skript zum anlegen der Homelaufwerke als Samba preexec        #
# kopieren des Skel in die Homes                                #
#                                                               #
# Version 0.3                                                   #
# Gregor Friedrich und Lars Uhlemann                            #
#################################################################
#Init
$time = ‘date‘; chomp $time;
$User  = $ARGV[0];
$PGrp  = $ARGV[1];
$Masch = $ARGV[2];
$BASE="/home/uhle/user/samba";
$SKEL="/home/uhle/user/skel";
$LOG="/home/uhle/perlldap/users.log";
open(FILE, ">>$LOG");

$bool=0;$index=0;
opendir(DLISTE,"$BASE") or
   print FILE "Fehler beim Verzeichnis lesen: $!\n" and die;

print FILE "$time User $User auf Maschiene $Masch meldet sich an,";
while (defined($file = readdir(DLISTE))) {
   if ($file eq $User) {
      $bool=1; print FILE " sein Verzeichnis existiert schon !\n";}
}
closedir(DLISTE);

if ($bool == 0) {
   print FILE " sein Verzeichnis wird angelegt !\n"; close FILE;
   mkdir "$BASE/$User", 0700; system("cp -r $SKEL/. $BASE/$User 2>>$LOG");
   system("chown -R $User:$PGrp $BASE/$User 2>>$LOG");
   # Setzen der Speicherplatzbeschränkung
   setquota -p default $User /user 2>>$LOG");
}

open(FILE, ">>$LOG");
$bool2=0;
opendir(GLISTE,"$BASE") or
   print FILE "Fehler beim Verzeichnis lesen: $!\n" and die;

while (defined($file = readdir(GLISTE))) {
   if ($file eq $User) { $bool2=1; }}
   if (($bool2 == 0) and ($bool == 0)) {
   print FILE "Fehler beim Anlegen des
```

```
      Homeverzeichnisses von User $User!\n";
}
close FILE;
closedir(DLISTE);
```

Perl-Script *groups10.pl* für die Gruppen-Freigabe

```
#########################################################################
#!/usr/local/bin/perl -w                                                #
# LDAP Abfrageskript fuer die Gruppenmitgliedschaften eines Benutzers   #
# sowie Anlegen der Gruppenverzeichnisse und Gruppenlinks               #
# Version 1.0a Lars Uhlemann                                            #
# Gruppenfile : validgroups                                             #
#########################################################################
#Init
use File::chmod;
use Net::LDAP ;
$User = $ARGV[0] ;
$Group = $ARGV[1];
$Masch = $ARGV[2];

local $Base = "dc=ufz, dc=de" ;
local $Host = "hera.rz.ufz.de" ;

local $LINKSDIR = "/groups/links";
#local $LINKSDIR = "/home/uhle/perlldap/home";

local $GRPDIR = "/groups/samba";
#local $GRPDIR = "/home/uhle/groups";

local $LOG = "/var/log/group.log";
#local $LOG = "/home/uhle/group.log";

local $SAMBA = "/etc/samba";

###############################################
#  Abfrage ob der Username nur aus gueltigen #
#  Zeichen besteht a-z,A-Z,0-9               #
###############################################
open(FILE, ">>$LOG");
flock(FILE,LOCK_EX);

if ($User =~ /^\w+$/) {
}else{ print FILE "Username enthaelt ungueltige
                   Zeichen ! Computer: $Masch\n"; close FILE  and die;}

close FILE;

#Verbindung mit LDAP-Server aufnehmen
$ldap = Net::LDAP->new($Host) or die "$@" ;
$ldap->bind || die "Could not bind to $Host" ;
```

```
##############################################
# 1. Hauptteil                               #
# Alle existierenden Gruppen abfragen,       #
# wenn noetig neue Grp.-verzeichnisse anlegen #
# und ungueltige Gruppen nicht anlegen       #
##############################################
$mesg = $ldap->search (
   base => $Base,
   filter => 'objectclass=posixgroup',
   attrs => ['cn']
   );
$mesg->code && $mesg->error;

# Anzahl gefundener Gruppen
my $max =$mesg->count;

if ($mesg->count == 0){ print "Keine Eintraege gefunden !\n";}

#Uebertragen der Gruppen in ein Array
for( my $index = 0 ; $index < $max ; $index++) {
   my $entry3 = $mesg->entry($index);
   $erg = $entry3->get_value( 'cn' );
   #Array mit allen exist. Gruppen
   $allgrp[$index]=$erg;
}

print "\n Alle exist. Gruppen im LDAP\n";
print "\n@allgrp\n"; #print "\n$max\n";

#Oeffnen der Liste der Gruppen, auf welche der Zugriff erlaubt ist !
open(FILE2,"$SAMBA/validgroups") or print FILE "  Die
           Gruppenliste konnte nicht geladen werden: $!\n";

$liste=<FILE2>;
@grpsplst= split(' ',$liste);
close FILE2;

#Anlegen neuer Gruppenverzeichnisse, falls diese noch nicht existieren !
$index=0;
opendir(GLISTE, $GRPDIR) or
           die "Verzeichnis kann nicht geoeffnet werden: $!\n";

while (defined($file = readdir(GLISTE))) {
   $filearray[$index]=$file;$index++;
}
$max=$index;
#print "$max\n"; #print "@filearray\n";

foreach $x (@allgrp) {
   $bool=0;
   for( $index = 2 ; $index < $max ; $index++) {
      if ($x eq $filearray[$index]){ $bool=1;}
   }
```

```
   if($bool == 0){
      foreach $z (@grpsplst){
         if ($z eq $x){
            mkdir "$GRPDIR/$x", 0777;
            system("/bin/chgrp $x $GRPDIR/$x");
            chmod("-rwxrws---","$GRPDIR/$x");
            print "Das Verzeichniss $x wurde neu angelegt !\n";
         }
      }
   }
}
#Verzeichnis-Objekt wird geschlossen
closedir(GLISTE);

############################################################
# 2.Hauptteil                                              #
# Verzeichnisse und Links je nach Gruppenmitgliedschaft    #
# anlegen oder loeschen                                    #
############################################################
$time = ‘date‘;
chomp $time;
open(FILE, ">>$LOG");
flock(FILE,LOCK_EX);
print FILE "$time User: $User aus Gruppe $Group auf
                    Computer: $Masch hat sich angemeldet\n";
close FILE;

#Oeffnen der Datei: validgroups
#enthaelt Liste der Gruppen, auf welche der Zugriff erlaubt ist !
open(FILE2,"$SAMBA/validgroups") or print FILE "  Die Gruppenliste
                                    konnte nicht geladen werden: $!\n";
$liste=<FILE2>;
@grpsplst= split(’ ’,$liste);

# Home Verzeichniss wird angelegt
if (-e "$LINKSDIR/$User"){
   print "HOME Verzeichnis exstiert schon -> alles OK!\n";
}else{
   mkdir "$LINKSDIR/$User", 0755;
   print FILE "  Gruppenhomeverzeichnis von $User
                 wurde angelegt -> erstes Login !\n";
}

# Links aller Gruppen, in der User Mitglied ist, werden angelegt
$grplst=‘/usr/bin/id -Gn $User‘;
@membergrp= split(’ ’,$grplst);

foreach $x (@membergrp){ symlink "$GRPDIR/$x","$LINKSDIR/$User/$x";}

#exist. Links werden auf Gueltigkeit mit Gruppenmitgliedschaften
#geprueft und ggf. geloescht

$index=0;
```

```
opendir(DLISTE, "$LINKSDIR/$User") or
   print FILE "  User $User, Verzeichnis kann nicht geoeffnet werden: $!\n";

while (defined($file = readdir(DLISTE))){
   if (($file ne '..') and ($file ne '.')){
      @existV[$index]=$file;
      $index++;
      $bool=0;
      foreach $x (@membergrp) {
         if ($x eq @existV[$index-1])  {
            $bool=1;
            #print "@existV[$index-1] existiert in membergrp !\n";
         }
      }
      if ($bool == 0) {
         print FILE "  User $User Link: @existV[$index-1] wird geloescht ->
            ist kein Mitglied mehr in der Gruppe @existV[$index-1] !\n";
         unlink "$LINKSDIR/$User/@existV[$index-1]";
      }
   }
   #Gruppenliste wird abgearbeitet

   $bool2=0;
   foreach $y (@grpsplst) {
      if (($y eq @existV[$index-1]) and ($bool == 1)) {$bool2=1;}
   }
   if (($bool2 == 0) and ($bool == 1)) {
      unlink "$LINKSDIR/$User/@existV[$index-1]";
      print FILE "  Link: @existV[$index-1] wurde nicht
         in der gueltigen Gruppenliste gefunden und geloescht !\n";
   }
}
close FILE2;
close FILE;
#Verbindung zu LDAP wird geschl.
$ldap->unbind ;
closedir(DLISTE);
```

6.3.2 Client-Script

```
''''''''''''''''''''''''''''''''''''''''''''''
'Client Skript fuer WIN NT, WIN 2000, WIN XP'
'Lars Uhlemann 2003                         '
'Version 3, Microsoft Visual Basic Script   '
''''''''''''''''''''''''''''''''''''''''''''''
On Error Resume Next

Set objArguments = Wscript.Arguments
Person = objArguments(0)
Set wshshell = CreateObject("WScript.Shell")

wartezeit = 3 'Sekunden
titel = "Netzlaufwerke-Verbindung"
```

```
text = "Willkommen Benutzer " & Person & ", die Netzlaufwerke " _
   & werden verbunden, bitte haben Sie etwas Geduld !"
antwort = wshshell.Popup(text, wartezeit,titel, vbSystemModal
   + vbInformation)

Set WSHNetwork = WScript.CreateObject("WScript.Network")

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' eventuell existierende Netzlaufwerke werden abgehängt (geloescht) '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
WshNetwork.RemoveNetworkDrive "G:",true,true
WshNetwork.RemoveNetworkDrive "P:",true,true
WshNetwork.RemoveNetworkDrive "U:",true,true
WshNetwork.RemoveNetworkDrive "H:",true,true

''''''''''''''''''''''''''''
'Ermitteln der GIDNUMBER:'
''''''''''''''''''''''''''''
SET resultat = wshshell.Exec("\\ADCENTER\NETLOGON\ldap\ldapsearch.exe " _
   & "-LLL -u -x -h hera.rz.ufz.de -b ""dc=ufz,dc=de"" "_
   & " ""(uid=" & Person & ")"" gidnumber")

strresultat = resultat.StdOut.ReadALL

position = InstrRev(strresultat,"gidnumber:")
gidnumber = mid(strresultat,position+11)

''''''''''''''''''''''''''''''''
'Ermitteln der PRIMAERE GRUPPE:'
''''''''''''''''''''''''''''''''
SET resultat = wshshell.Exec("\\ADCENTER\NETLOGON\ldap\ldapsearch.exe"_
   & " -LLL -u -x -h hera.rz.ufz.de -b ""dc=ufz,dc=de"" " _
   & " ""(&(objectclass=posixgroup)(gidnumber=" & gidnumber & " ))"" cn")

strresultat = resultat.StdOut.ReadALL

position = InstrRev(strresultat,"cn:")
primgrpzw = mid(strresultat,position+4)

primgrp=""
x=0
do until (x = len(primgrpzw)) or (buchstabe=vbcr)
   x=x+1
   primgrp = primgrp & buchstabe
   buchstabe = mid(primgrpzw,x,1)
loop

laenge = len(primgrp)

''''''''''''''''''''''''
'Mitglied von movegrp ?'
''''''''''''''''''''''''
SET resultat = wshshell.Exec("\\ADCENTER\NETLOGON\ldap\ldapsearch.exe"_
```

```
   & " -LLL -u -x -h hera.rz.ufz.de -b ""dc=ufz,dc=de"" " _
   & " ""(&(objectclass=posixgroup)(cn=movegroup)(memberuid= "_
   & " " & Person & "))"" cn")

strresultat = resultat.StdOut.ReadALL

position = InstrRev(strresultat,"cn:")
movegrp = mid(strresultat,position+4,9)

if movegrp = "movegroup" then mvgrp = "true" else mvgrp = "false" end if

''''''''''''''''''''''''''
'Mitglied von newgroup ? '
''''''''''''''''''''''''''
if mvgrp = "false" then
   SET resultat = wshshell.Exec("\\ADCENTER\NETLOGON\ldap\ldapsearch.exe"_
      & " -LLL -u -x -h hera.rz.ufz.de -b ""dc=ufz,dc=de"" " _
      & " ""(&(objectclass=posixgroup)(cn=newgroup)(memberuid= "_
      & " " & Person & "))"" cn")

   strresultat = resultat.StdOut.ReadALL

   position = InstrRev(strresultat,"cn:")
   newgrp = mid(strresultat,position+4,8)

   if newgrp = "newgroup" then newgrp = "true" else newgrp = "false" end if
end if

''''''''''''''''''''''''''''''''''''''''''''''''
'Laufwerke, je nach Umstellungsstatus verbinden'
''''''''''''''''''''''''''''''''''''''''''''''''
if (newgrp = "true") or (mvgrp = "true") then
   WSHNetwork.MapNetworkDrive "G:", "\\share2\gruppen"
   WSHNetwork.MapNetworkDrive "P:", "\\share2\programme"
   WSHNetwork.MapNetworkDrive "H:", "\\share1\nutzer"
   WSHNetwork.MapNetworkDrive "U:", "\\share1\ufzall"
else
   resultat=wshshell.Run("%COMSPEC% /C %WINDIR%\system32\net.exe use _
      H: \\venus\" & primgrp & " /PERSISTENT:NO",1,true)
   WSHNetwork.MapNetworkDrive "G:", "\\share2\gruppen"
   WSHNetwork.MapNetworkDrive "P:", "\\share2\programme"
   WSHNetwork.MapNetworkDrive "U:", "\\share1\ufzall"
end if

WScript.Sleep 4000 'Benoetigte Zeit für das Anlegen der Freigaben-Laufwerke
if (WSHNetwork.EnumNetworkDrives.count => 4) then
   text = "Die Netzlaufwerke stehen zur Verfügung !"
   antwort = wshshell.Popup(text, wartezeit,titel, vbSystemModal _
   + vbInformation)
else
   Msgbox "Die Netzlaufwerke stehen nicht zur Verfügung !"
      & WSHNetwork.EnumNetworkDrives.count, vbSystemModal + vbCritical
end if
```

6.3.3 Quota-Skript

Perl-Skript *quota.pl* zum einrichten der Gruppenquotas

```
#########################################################################
# Automatisches Setzen der Gruppen Quotas,Version 1, Lars Uhlemann      #
#                                                                       #
#1.LDAP Abfragen:                                                       #
#  -Ermitteln der primaeren Gruppen ID                                  #
#  -Ermitteln anhand der primaeren Gruppen ID wieviel "direkte"         #
#   Miglieder diese Gruppe 0besitzt                                     #
#  -Ermitteln der Mitgliederanzahl welcher nur Mitglied der Gruppe      #
#   sind (Benutzer koennen Mitglied in mehreren Gruppen sein !)         #
#                                                                       #
#2.Anhand der Mitgliederzahlen Berechnung der Quotas fuer jede Gruppe   #
#                                                                       #
#3.Setzen der jeweiligen Quotas (unter Beachtung der im UFZ             #
#  gueltigen Quotabestimmungen)                                         #
#########################################################################
#Init
use Net::LDAP;

local $BasisQuota   = 10485670;
local $proUserQuota = 1048567;
local $HardQuotaAufschlag = 2097134;
local $Base = "dc=ufz, dc=de";
local $Host = "hera.rz.ufz.de"; #LDAP Master
local $Host2 = "lhslave.halle.ufz.de"; #LDAP Slave
local $SAMBA = "/etc/samba";

$ldap = Net::LDAP->new($Host) or
$ldap = Net::LDAP->new($Host2) or die "$@";
$ldap->bind || die "Es konnte keine Verbindung aufgenommen werden !\n";

#Die Subroutine search ermittelt die Anzahl der Mitglieder einer Gruppe
sub search($gruppe){
   #Suche nach der primaren Gruppen ID
   $mesg = $ldap-> search (
      base => $Base,
      filter => "(&(objectclass=posixgroup) (cn=$gruppe))",
      attrs => ['gidnumber']
   );
   $mesg->code && $mesg->error;
   my $max = $mesg->count;
   my $entry = $mesg->entry(0);
   $erg = $entry->get_value('gidnumber');

   # Suche der Mitglieder der Gruppe welche primaer zur Gruppe gehoehren
   $mesg = $ldap-> search (
      base => $Base,
      filter => "gidnumber=$erg",
      attrs => ['cn']
   );
```

```
   $mesg->code && $mesg->error;
   $max = $mesg->count;
   if ($mesg->count == 0){print "Keine Eintraege gefunden !\n";}

   # Suche nach weiteren Mitgliedern der Gruppe
   $mesg = $ldap-> search (
      base => $Base,
      deref => always,
      filter => "(&(objectclass=posixgroup) (cn=$gruppe))",
      attrs => ['memberUid']
   );
   $entry = $mesg->entry(0);
   @erg2 = $entry->get_value('memberUid');
   $max2 = @erg2;
   if ($mesg->count == 0){print "Keine memberUid Einträge gefunden!\n";}

   $max = $max+$max2; return $max;
}

# Subroutine zum Berechnen und Festlegen der Gruppenquotas
sub quota($count,$gruppe){
   my $Soft=0; my $Hart=0;
   if ($count < 30){
      $Soft = $proUserQuota * $count + $BasisQuota;
      $Hart = $Soft + $HardQuotaAufschlag;
      system("setquota -g -F xfs $gruppe $Soft $Hart 0 0 /groups/");
   }else{
      #Ist die Mitgliederanzahl groesser als 30 werden
      #40 GByte Quota gesetzt
      system("setquota -g -F xfs $gruppe 41942680 44039814 0 0 /groups/");
   }
}

#Die Gruppendatei mit allen gueltigen Gruppen wird geoeffnet
open(FILE,"$SAMBA/validgroups");
$liste=<FILE>;
@grpliste=split(' ',$liste);
close FILE;
#print "@grpliste\n";

#Die jeweiligen Quotas werden entsprechend der Mitgliederanzahl
#festgelegt
local $index=0;
foreach $x (@grpliste){
   $gruppe=@grpliste[$index];
   $count = &search();
   $index++; &quota();
}
```

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACL	Access Control List
BIOS	Basic Input Output System
CIFS	Common Internet File System
CPU	Central Processing Unit
GID	Group Identificator
HSM	Hierarchische Speicherverwaltung
iFCP	Fibre Channel Protocol over IP
IP	Internet Protocol
iSCSI	Small Computer System Interface over IP
LDAP	Light Weight Access Protocol
LVM	Logical Volume Manager
MFT	Master File Table
NAS	Network Attached Storage
NDS	Novell Directory Service
NetBEUI	NetBIOS Enhanced User Interface
NetBIOS	Network Basic Input Output System
NFS	Network File System
NIS	Network Information Service
NTFS	Native File System
OSI	Open System Interconnection
RAID	Redundant Array of Independant Discs
RFC	Request For Comments
SAN	Storage Area Network
SCSI	Small Computer System Interface
SMB	Server Message Block Protocol
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
UDP	User Datagramm Protocol
UFZ	Umweltforschungszentrum
UID	User Identificator

Literaturverzeichnis

[Ban01] Jens Banning, *LDAP unter Linux*, Addison-Wesley, 1. Auflage, München, 2001

[Die00] Dr. Oliver Diedrich, *Von Blöcken und Knoten, Das Linux Dateisystem ext2*, Magazin für Computer und Technik, Heise-Verlag, Ausgabe 06/2000, Hannover, 2000

[Die02] Dr. Oliver Diedrich, *Fürs Protokoll*, Magazin für Computer und Technik, Heise-Verlag, Ausgabe 06/2002, Hannover, 2002

[Die03] Dr. Oliver Diedrich, *Gut bewacht, Access Control Lists in modernen Dateisystemen*, Magazin für Computer und Technik, Heise-Verlag, Ausgabe 23/2003, Hannover, 2003

[Dit03] Patrick Ditchen, *Perl in 21 Tagen*, Markt und Technik, 1. Auflage, München, 2003

[Erk03] Ulf Troppens & Rainer Erkens, *Speichernetze*, dpunkt.verlag, 1. Auflage, Heidelberg, 2003

[Fie01] Gary Field, *SCSI*, mitp-Verlag, 1. Auflage, Bonn, 2001

[Hab03] Franz Haberhauer, *Network File System Version 4*, Sun Microsystems, http://de.sun.com/Downloads/Praesentation/2002/Linuxworld/pdf/NFSv4.pdf (27.10), 2003

[Her99] Helmut Herold, *Linux·Unix - Systemprogrammierung*, Addison-Wesley, 2. Auflage, München, 1999

[ISI80] J. Postel ISI, *RFC 768 User Datagram Protocol*, Internet Engineering Task Force, http://www.ietf.org/rfc (23.10.2003), 1980

[Kau97] Franz-Joachim Kauffels, *Moderne Datenkommunikation*, Datacom-Verlag, 2. Auflage, Bonn, 1997

[Kup00] Martin Kuppinger, *Microsoft Windows 2000 Server Das Handbuch*, Microsoft Press, 1. Auflage, Unterschleißheim, 2000

[Lam97] Stefan Lamberts, *Parallele verteilte Dateisysteme in Rechnernetzen*, SHAKER-Verlag, 1. Auflage, Aachen, 1997

[Len02] Volker Lendecke, *Kursskript SAMBA*, Service Network GmbH, http://www.SerNet.de (21.09.2003), 1. Auflage, Göttingen, 2002

[MG03] Dr. Boris Pasternak Dr. Uwe Meyer-Gruhl, *Der Gleich-Macher*, Magazin für Computer und Technik, Heise-Verlag, Ausgabe 10/2003, Hannover, 2003

[Mic87a] Sun Microsystems, *RFC 1014 XDR: External Data Representation Standard*, Internet Engineering Task Force, http://www.ietf.org/rfc (23.10.2003), 1987

[Mic87b] Sun Microsystems, *RFC 1094 NFS: Network File System Protocol Specification*, Internet Engineering Task Force, http://www.ietf.org/rfc (23.10.2003), 1987

[Mic88] Sun Microsystems, *RFC 1057 RPC: Remote Procedure Call Protocol Specification Version 2*, Internet Engineering Task Force, http://www.ietf.org/rfc (23.10.2003), 1988

[Mic95] Sun Microsystems, *RFC 1813 NFS Version 3 Protocol Specification*, Internet Engineering Task Force, http://www.ietf.org/rfc (23.10.2003), 1995

[Mil99] Mark A. Miller, *Troubleshooting TCP/IP*, mitp-Verlag, 1. Auflage, Bonn, 1999

[oSC81a] University of Southern California, *RFC 791 INTERNET PROTOCOL*, Internet Engineering Task Force, http://www.ietf.org/rfc (23.10.2003), 1981

[oSC81b] University of Southern California, *RFC 793 TRANSMISSION CONTROL PROTOCOL*, Internet Engineering Task Force, http://www.ietf.org/rfc (23.10.2003), 1981

[Pec02] Jörg Pech, *Die Technik von 10-Gigabit Ethernet*, Magazin für Computer und Technik, Heise-Verlag, Ausgabe 03/2002, Hannover, 2002

[Ric03] Rolf D. Richter, *STORAGE-Magazin*, Fachzeitschrift für Speichertechnologien, IT Verlag für Informationstechnik GmbH, Ausgabe 1, Sauerlach, 2003

[Rus03] Richard Russon, *NTFS-Documenation*, SourceForge Projet, http://linux-ntfs.sourceforge.net (17.09), 2003

[Sch03] Achim Schmidt, *Linux Basiswissen*, Linux Info, http://www.linuxinfo.de/de/basis/15.html (16.11), 2003

[Sed92] Robert Sedgewick, *Algorithmen*, Addison-Wesley, 1. Auflage, Bonn, 1992

[Sha00] Rawn Shah, *Unix and Windows 2000 Integration Toolkit*, Wiley-Verlag, 1. Auflage, New York (USA), 2000

[Sie99] Ellen Siever, *Linux in a Nutshell*, O'Reilly, 2. Auflage, Köln, 1999

[Ste03] Andreas Steinwede, *Sicherungskopie, Strategien gegen Datenverlust*, Magazin für Computer und Technik, Heise-Verlag, Ausgabe 08/2003, Hannover, 2003

[u.a99] Robert Eckstein David Collier-Brown u.a., *Using Samba*, O'Reilly, 1. Auflage, Sebastopol (USA), 1999

[u.a00] Sun Microsystems C. Beame Hummingbird Ltd. u.a., *RFC 3010 NFS version 4 Protocol*, Internet Engineering Task Force, http://www.ietf.org/rfc (23.10.2003), 2000

[u.a01] Ray Bryant Dave Raddatz u.a., *PenguinoMeter: A New File-I/O Benchmark for Linux*, Times N Systems, http://pgmeter.sourceforge.net/pgmeter.pdf (10.08.2003), 2001

[u.a02] Ray Bryant Ruth Forester u.a., *Filesystem Performance an Scalability in Linux*, USENIX Association, http://oss.sgi.com/projects/xfs/publications.html (10.08.2003), 2002

[vOu96] A.Menezes P. van Ooschot u.a., *Handbook of Applied Cryptography*, CRC Press, 1. Auflage, Boca Ralon (USA), 1996

[Wel00] Tobias Weltner, *Windows Skriptinghost*, Franzis Verlag, 1. Auflage, Poing, 2000

[wG87a] Network working Group, *RFC 1001 PROTOCOL STANDARD FOR A NetBIOS SERVICE ON A TCP/UDP TRANSPORT: CONCEPTS AND METHODS*, Internet Engineering Task Force, http://www.ietf.org/rfc (23.10.2003), 1987

[wG87b] Network working Group, *RFC 1002 PROTOCOL STANDARD FOR A NetBIOS SERVICE ON A TCP/UDP TRANSPORT: DETAILED SPECIFICATIONS*, Internet Engineering Task Force, http://www.ietf.org/rfc (23.10.2003), 1987

Glossar

ACL's Access Control Lists (deutsch: Zugriffs Kontroll Listen), sind ein Rechtevergabesystem auf Dateien und Verzeichnisse welche eine feinere, detailliertere, Vergabe von Rechten ermöglicht. Mit ACL's ist es möglich Zugriffsrechte auf einzelne Benutzer und Gruppen zu setzen.

Administration ist die Organisation, Verwaltung, Steuerung oder Wartung einer Hard- und/oder Softwarelösung für einen beliebigen Einsatzzweck.

Archivierung ist die sequentielle Sicherung von Daten zu bestimmten Zeitpunkten, um später wieder auf die Datenstände (zum jeweiligen Archivierungszeitpunkt) zugreifen zu können[Ste03].

Backup stellt eine Sicherheitskopie von Datensätzen für den Fall eines unerwarteten Datenverlustes dar. Tritt dieser Fall ein, wird die Sicherheitskopie genutzt, um die verlorenen Daten wieder zu ersetzen.

Block ist die kleinste Daten-Einheit auf einem Datenträger. Blockgrößen von 512 Byte bis 4096 Byte sind heutzutage üblich.

CIFS Common Internet File System, siehe SMB.

Client Rechner über den ein Benutzer, angebotene Dienste in Anspruch nimmt.

CPU englisch: Central Processing Unit, zentrale Recheneinheit, Hauptkomponente der Rechner-Hardware.

Datei stellt eine Sammlung von Datensätzen beliebigen Inhalts dar.

Fibre Channel wird in Speichernetzen als Übertragungstechnologie eingesetzt. Eigenschaften sind zum Beispiel serielle Übertragung für wei-

te Entfernungen und hohe Geschwindigkeiten, geringe Rate an Übertragungsfehlern und eine geringe Verzögerung (Latenz) der übertragenen Daten. Die Übertragungsgeschwindigkeiten betragen 100 Mbyte/s (800 MBit/s) oder 200 Mbyte/s (1600 MBit/s).

Freigabe ist der Teil von Dateien und Verzeichnissen, welcher sich auf einem Fileserver befindet und für den Zugriff über das Netzwerk auf den Clients zur Verfügung gestellt wird (Fernzugriff).

Home-Verzeichnis Enthält persönliche Daten des Benutzers wie zum Beispiel Dokumente und persönliche Programm-Einstellungen. Nur der Eigentümer besitzt Zugriff (Lesend/Schreibend) auf den Inhalt seines Home-Verzeichnisses.

HSM Hierarchische Speicherverwaltung; täuscht dem Benutzer unendlich große physikalische Datenträger vor. Sie lagert Daten, auf welche längere Zeit (Zeitspanne ist konfigurierbar) nicht zugegriffen wurde, von den lokalen Datenträgern (zum Beispiel vom Fileserver) auf den Backup-Server aus. Wird eine ausgelagerte Datei benötigt, überträgt HSM diese automatisch zurück. Dieser Vorgang bleibt für den Benutzer vollkommen verborgen.

iFCP ist eine Implementierung des Fibre Channel-Protokolls für die Übertragung von Daten über ein IP-Netzwerk.

InfiniBand wird in Speichernetzen als Übertragungstechnologie eingesetzt. Übertragungsgeschwindigkeiten, unter Einsatz von Übertragungskanalbündelung, von 10 GBit/s bis zu maximal 30 GBit/s.

iSCSI ist eine Implementierung des bekannten SCSI-Protokolls[Fie01] zur Übertragung von Daten über IP (Internet Protokoll)[Kau97]. Die Funktionen und Eigenschaften entsprechen dem normalen SCSI-Standard, zusätzlich angepasst auf die Übertragung über ein IP-Netzwerk.

Journaling bewahrt die Konsistenz des Dateisystems durch protokollieren aller Transaktionen auf Blockebene. Durch System-Abstürze verursachte Inkonsistenz wird durch das aufgezeichnete Transaktionsprotokoll wiederhergestellt.

Links/Verknüpfungen ermöglichen es an einem beliebigen Ort im Dateisystem, für Dateien oder Verzeichnisse, Pseudonyme zu erstellen. Mit diesen

Pseudonymen wird wie mit normalen Dateien oder Verzeichnissen gearbeitet. Alle Änderungen an diesen Pseudonymen wirken sich direkt auf die originalen Dateien oder Verzeichnisse aus. Gleichzeitig besitzen diese die gleichen Rechte wie die mit ihnen verknüpften Originale.

Migration Migration beschreibt den Übergang von einem System zu einem anderen System, unter der Garantie, daß mindestens der Funktionsumfang des ursprünglichen Systems im neuen System erhalten bleibt.

NAS Network Attached Storage (Speicherstrategie) Speicherverbünde, welche über das normale Netzwerk mit den Fileservern verbunden werden und dateiorientiert arbeiten.

NFS Network File System, von SUN Microsystems im Jahre 1984 entwickeltes Netzwerk-Dateisystem. Es baut auf dem Client-Server Modell auf, wobei der Server Dateien und Verzeichnisse in Form von Freigaben dem Client zur Verfügung stellt. NFS ist ein eigenständiges Protokoll, welches auf die UDP-Protokoll-Schicht (NFS-Version 1 bis 3) oder die TCP-Protokoll-Schicht (NFS-Version 4) aufsetzt.

RAID Redundant Array of Independant Discs, verfolgt das Ziel die Ausfallsicherheit (Redundanz) von Festplatten-Verbünden zu erhöhen (Schutz vor Datenverlust). Redundanz speichert zusätzlich Informationen, so daß der normale Betrieb nach Austausch eines defekten Datenträgers mit einem neuen fehlerfreien Datenträger fortgesetzt werden kann. Wird je nach genutzten Redundanz-Verfahren in verschiedene RAID-Levels eingeteilt.

SAN Storage Area Network (Speicherstrategie) SAN's sind unabhängige Speicherverbünde die sich außerhalb der eigentlichen Fileserverhardware befinden und über ein spezielles Netzwerk, dem Speichernetzwerk, mit diesen verbunden sind.

Serverzentrierter Speicher ist direkt an einen einzelnen Fileserver angeschlossen. Zugriff auf diesen ist nur über den Fileserver selbst möglich.

Skripte Als Skript wird eine Folge von Befehlen bezeichnet, welche in einer Skript-Datei aufgeführt sind und bei deren Ausführung (Skript-Datei) durch einen Skript-Interpreter hintereinander ge-

startet werden. Es existieren dazu sogenannte Skript-Sprachen, welche zusätzlich komfortable Funktionen wie Schleifen, Bedingte Anweisungen, Variabeln usw. zur Verfügung stellen.

SMB Server Message Protokoll, von IBM (1985) zur Dateiübertragung innerhalb von Netzwerken entworfen. Es baut auf dem Client-Server Modell auf, wobei der Server Dateien und Verzeichnisse in Form von Freigaben dem Client zur Verfügung stellt. SMB wurde von Microsoft wesentlich weiterentwickelt und ist in allen Microsoft Betriebssystemen (seit Windows for Workgroups) enthalten. Mit der neuen Protokoll-Bezeichnung CIFS (Common Internet File System) wurde der alte Name SMB (seitens Microsoft) abgelöst.

Verzeichnisse enthalten listenartig Namen von Dateien (auch Verzeichnissen), sowie Verweise auf derren Standort. Sie sind selbst Dateien und sequentiell oder baumartig organisiert.

Zugriffsrechte bezogen auf Datei und Verzeichnisse, erlauben eine Zugriffsregulierung auf deren Inhalte. Drei Benutzergruppen (Besitzer/Ersteller, Gruppe, Alle) besitzen ein oder mehrere der drei grundlegende Rechte (Lesen, Schreiben, Ausführen) auf eine Datei oder Verzeichnis.

Index