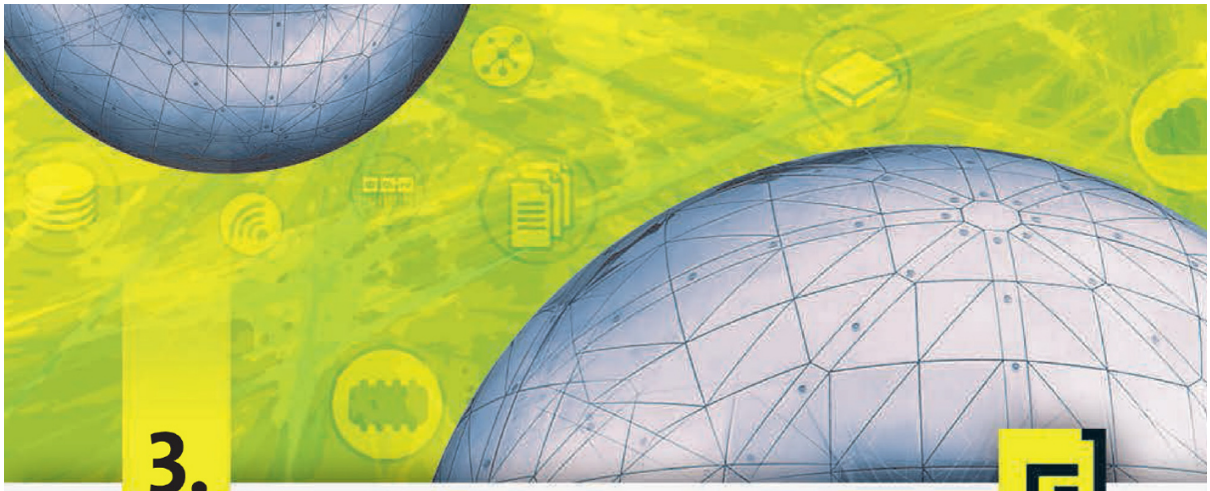

3. Auflage

Ulf Troppens · Nils Haustein

Speichernetze

Grundlagen, Architekturen, Datenmanagement

iX EDITION

dpunkt.verlag

Inhalt

9.1.1 Definition: »Pervasive Computing«

9.1.2 Dezentrale Erzeugung, Verarbeitung und Speicherung von unstrukturierten Daten

9.1.3 Höheres Datenvolumen

9.1.4 Höhere Skalierbarkeit

9.1.5 Höhere Anpassbarkeit

9.1.6 Geringere Veränderungsrate

9.1.7 Verfügbarkeit wichtiger als Konsistenz

9.1.8 Höhere Fehlertoleranz

9.1.9 Geringere Belastung durch Partitionierung

9.1.10 Lose gekoppelte Replikate

9.1.11 Fazit

9.2 Cloud Computing

9.2.1 Definition »Cloud Computing«

9.2.2 Charakteristische Eigenschaften

9.2.3 Dienstmodelle: IaaS, PaaS, SaaS

9.2.4 Bereitstellungsmodelle: Public, Privat, Hybrid

9.2.5 Fallbeispiel: OpenStack

9.2.6 Abgrenzung zu Webanwendung

9.2.7 Abgrenzung zu Virtualisierung

9.2.8 Cloud Computing in Unternehmen

9.3 Servervirtualisierung

9.3.1 Grundlagen und Definition

9.3.2 Vorteile von Servervirtualisierung

9.3.3 Speicher für Servervirtualisierung

9.3.4 Problem: Hypervisor im I/O Pfad

9.3.5 Fallstudie: Speicher für VMware ESXi

9.3.6 Hyperconverged Systems

4 Dateisysteme und Network Attached Storage (NAS)

Achim Christ · Ulf Troppens · Nils Haustein

Ziel des Kapitels

Disksysteme stellen blockorientierten Speicher zur Verfügung. Für Anwender und Anwendungen ist der Umgang mit Blöcken, adressiert über Zylinder, Spuren und Sektoren, sehr unhandlich. Dateisysteme stellen daher eine Zwischenschicht im Betriebssystem dar, die für Anwender und Anwendungen die gewohnten Verzeichnisse (Directories) beziehungsweise Ordner (Folder) sowie Dateien (Files) bereitstellt und diese für Anwender verborgen auf blockorientierten Speicher ablegt. Dieses Kapitel stellt die Grundlagen von Dateisystemen vor und zeigt, welche Rolle sie im Zusammenhang mit Speichernetzen spielen.

Gliederung des Kapitels

In diesem Kapitel beschreiben wir zunächst grundlegende Anforderungen, die an Dateisysteme gestellt werden (Abschnitt 4.1). Danach stellen wir Netzwerk-Dateisysteme vor, die auch in Fileservern und in Network-Attached-Storage-Systemen (NAS-Systemen) zum Einsatz kommen (Abschnitt 4.2). Netzwerk-Dateisysteme ermöglichen es vielen Benutzern, gleichzeitig auf gemeinsamen Daten zu arbeiten. In Abschnitt 4.3 beschreiben wir entsprechende Techniken, die vor dem unerlaubten Zugriff auf Dateien schützen. Dann erläutern wir mögliche Leistungsengpässe von Netzwerk-Dateisystemen und zeigen Ansätze, diese zu vermeiden (Abschnitt 4.4). Wir schließen das Kapitel mit einem Vergleich von blockorientierten Speichernetzen (Fibre Channel SAN, iSCSI SAN) und Network Attached Storage (NAS) (Abschnitt 4.5). Das Kapitel schließt wie alle Kapitel mit einer Zusammenfassung des Kapitels und dem Ausblick auf die nächsten Kapitel (Abschnitt 4.6).

4.1 Lokale Dateisysteme

Basisdienst: Bereitstellen von Verzeichnissen und Dateien

Dateisysteme bilden eine Zwischenschicht zwischen den blockorientierten Laufwerken (Flashmodule, SSDs und Festplatten) und den Anwendungen, wobei zwischen dem Dateisystem und den Laufwerken oftmals ein Volume Manager eingesetzt wird (Abb. 4–1). Beide zusammen verwalten die Blöcke der Laufwerke, die je nach Konfiguration des Speichers physisch oder virtuell sind, und stellen diese Anwendern und Anwendungen über die gewohnten Verzeichnisse und Dateien zur Verfügung.

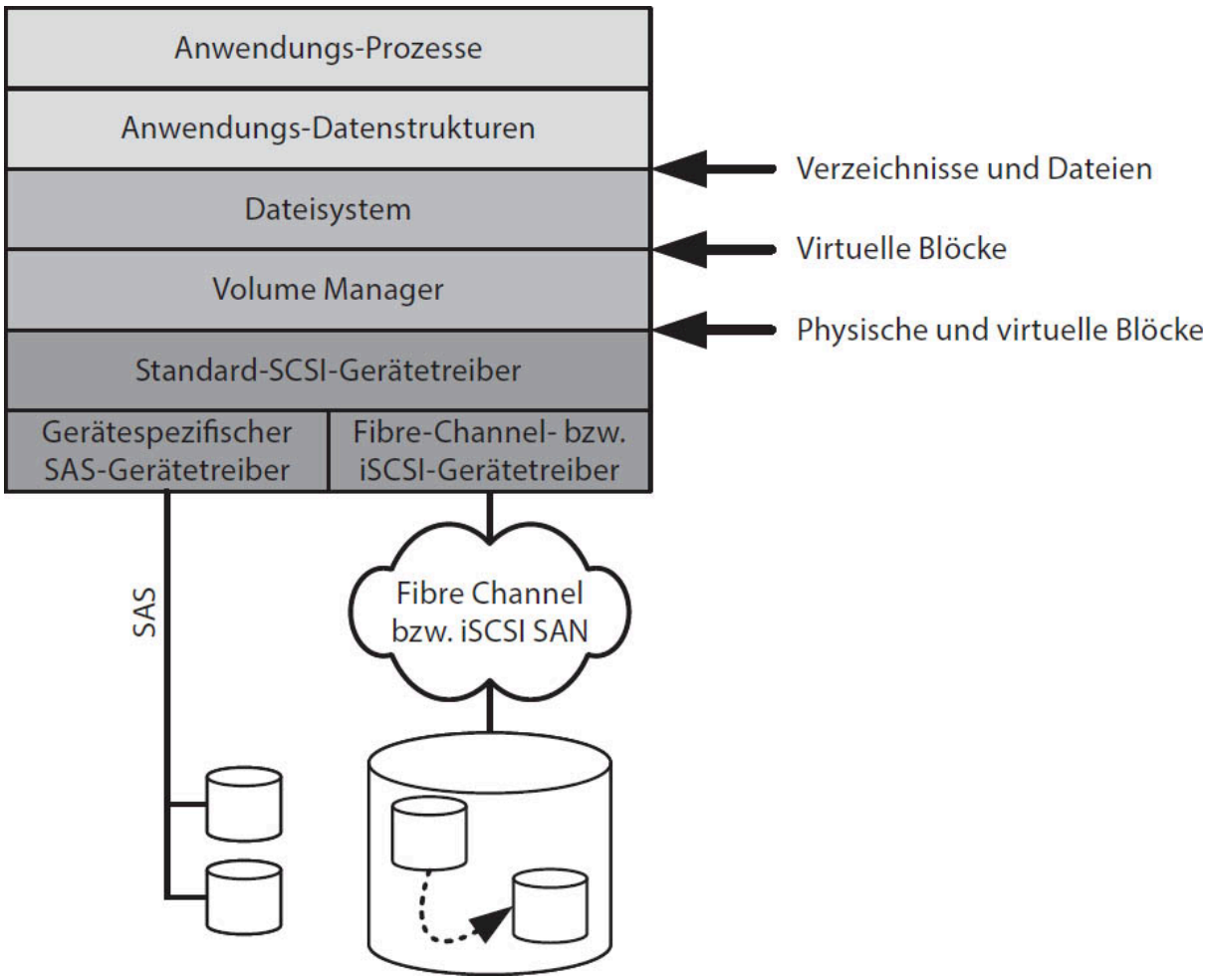

Abb. 4–1 Lokales Dateisystem und Volume Manager

Volume Manager und Dateisystem verwalten die Blöcke der blockorientierten Laufwerke. Anwendungen und Anwender nutzen so die Speicherkapazität der Laufwerke über Verzeichnisse und Dateien.

Weitere Funktionen von Dateisystemen

Moderne Dateisysteme stellen über die Basisdienste hinaus Funktionen wie Journaling, Snapshots, dynamische Erweiterung und Daten-Management bereit. Diese Funktionen erläutern wir in den nachfolgenden Abschnitten näher.

4.1.1 Lokale und verteilte Dateisysteme

Lokales Dateisystem

Dateisysteme können entweder nur innerhalb eines Rechners zur Verfügung stehen oder von mehreren Rechnern benutzt werden. Dateisysteme, die nur innerhalb eines Rechners zur Verfügung stehen, werden lokale Dateisysteme genannt. Jedes Endgerät wie Server, Desktops, Laptops, Smartphones und Tablets hat typischerweise mindestens ein lokales Dateisystem, von dem das Betriebssystem gestartet wird.

Ein lokales Dateisystem steht nur dem Endgerät zur Verfügung, auf dem es installiert ist.

Netzwerk-Dateisysteme

Es gibt auch Dateisysteme, die von mehreren Endgeräten zugreifbar sind. Diese Dateisysteme werden von einem oder mehreren Servern in einem Netzwerk zur Verfügung gestellt, über das mehrere Endgeräte darauf zugreifen können. Diese Dateisysteme werden Netzwerk-Dateisysteme genannt, weil sie über ein Netzwerk bereitgestellt werden. Endgeräte können solche Netzwerk-Dateisysteme in ihre lokalen Dateisysteme einhängen (»mounten«) und so auf Verzeichnissen und Dateien arbeiten, die auf einem anderen Server gespeichert werden.

Weitere Vorgehensweise

Im weiteren Verlauf dieses Abschnitts beschäftigen wir uns mit den Grundlagen von lokalen Dateisystemen. Im nächsten Abschnitt (Abschnitt 4.2) schauen wir uns dann Netzwerk-Dateisysteme näher an. Im Anschluss beschreiben wir die Kontrolle der Zugriffe auf Dateien (Abschnitt 4.3) und ausgewählte weiterführende Themen (Abschnitt 4.4).

4.1.2 Journaling

Journaling

Journaling ist ein Mechanismus, der die Konsistenz eines lokalen Dateisystems über Systemabstürze hinweg gewährleistet. Dazu schreibt das Dateisystem jede Änderung zunächst in ein für Anwender und Anwendungen nicht sichtbares Log, bevor es die Änderung im Dateisystem selbst vornimmt. Sobald die Änderung erfolgreich im Dateisystem vorgenommen wurde, wird auch diese Information wieder im Log vermerkt – die Transaktion ist damit komplett. Nach einem Systemabsturz braucht das Dateisystem nur das Ende der Logdatei nachzufahren, um die Konsistenz des Dateisystems wiederherzustellen.

Problem: File System Check ohne Journaling

Bei Dateisystemen ohne Journaling, vor allem älteren Dateisystemen wie Microsofts FAT32 oder das in Unix-Systemen verwendete UFS-Dateisystem, muss nach einem Systemabsturz die Konsistenz des gesamten Dateisystems überprüft werden (File System Check); bei großen Dateisystemen kann dies mehrere Stunden dauern. Je nach Größe des Dateisystems können somit viele Stunden vergehen, bis die Daten und damit die Anwendungen nach einem Systemabsturz wieder verfügbar sind.

4.1.3 Snapshots

Snapshots: Instant Copies auf Dateisystemebene

Snapshots stellen die gleiche Funktion bereit wie die von Disk- und Flashsystemen her bekannten Instant Copies (Abschnitt 2.4.1): Snapshots frieren den Zustand eines Dateisystems zu einem bestimmten Zeitpunkt ein. Anwender und Anwendungen können über einen speziellen Pfad auf die eingefrorene Kopie zugreifen. Das Erstellen der Kopie erfolgt wie bei den Instant Copies innerhalb weniger Sekunden und ist speicherplatzeffizient. Das heißt, dass die Dateien nicht in den Snapshot kopiert werden, sondern dass nur die Verweise auf die Dateiblöcke eingefroren werden. Bei nachfolgenden Änderungen von Dateiblöcken wird vorher entweder der originale Block in den Snapshot kopiert (copy on write) oder aber der geänderte Block wird neben den originalen Block geschrieben (redirect on write). Wie auch bei Instant Copies ist bei dem Erstellen eines Snapshots darauf zu achten, dass der Zustand der eingefrorenen Daten der jeweiligen Anwendung konsistent ist.

Vergleich: Instant Copies versus Snapshots

Tabelle 4–2 vergleicht Instant Copies und Snapshots. Ein wichtiger Vorteil von Snapshots besteht darin, dass sie mit jeder Hardware realisiert werden können. Dafür entlasten Instant Copies innerhalb eines Disk- oder Flashsystems die CPU und die Busse des Servers, sodass mehr Systemressourcen für die eigentlichen Anwendungen bleiben.

	Instant Copy	**Snapshot**
Realisierungsort	Disk-/Flashsystem	Dateisystem
Ressourcenverbrauch	Belastet Controller und CPU des Disk-/Flashsystems	Belastet CPU des Servers und alle Busse
Verfügbarkeit	Je nach Disk-/Flashsystem (hardwareabhängig)	Je nach Dateisystem (hardwareunabhängig)

Tab. 4–2 Vergleich: Snapshot versus Instant Copy
Snapshots sind hardwareunabhängig; dafür belasten sie die CPU des Servers.

4.1.4 Volume Manager

Grundfunktion: Virtualisierung von Laufwerken

Der Volume Manager ist eine Speichervirtualisierungsschicht innerhalb des Betriebssystems zwischen dem Dateisystem

beziehungsweise der Anwendung und den Laufwerken des angeschlossenen Speichers. Die wichtigste Grundfunktion des Volume Managers besteht darin, mehrere Laufwerke des Speichers zu einem großen virtuellen Laufwerk zusammenzufassen und nach oben hin nur dieses sichtbar zu machen. Für den Volume Manager bleibt es dabei verborgen, ob es sich um physische Laufwerke wie Festplatten oder SSDs oder um virtuelle Laufwerke eines Disksystems handelt. Ein Dateisystem nutzt dann die virtuellen Laufwerke des Volume Managers als Speicher für Dateien, Verzeichnisse und Metadaten. Die meisten Volume Manager bieten die Möglichkeit, diese großen virtuellen Laufwerke wieder in mehrere kleinere virtuelle Laufwerke zu zerlegen sowie deren Kapazität dynamisch zu vergrößern oder zu verkleinern (Abb. 4–3). Diese Speichervirtualisierung innerhalb des Volume Managers ermöglicht es Serveradministratoren, schnell auf veränderte Speicheranforderungen von Anwendungen wie Datenbanksystemen und Dateisystemen zu reagieren, ohne jeweils den Administrator des Disksystems zu kontaktieren.

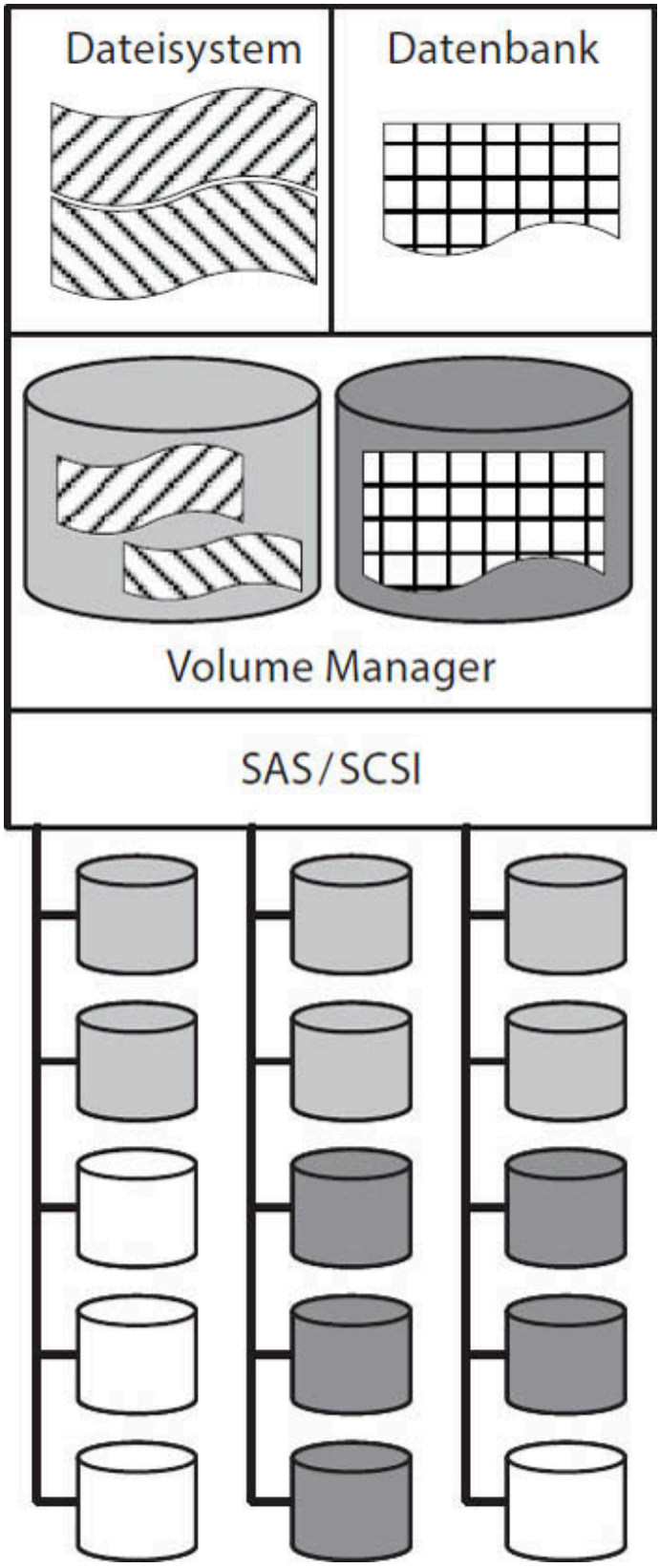

Abb. 4–3 Volume Manager

Der Volume Manager fasst physische Laufwerke zu virtuellen Laufwerken zusammen, die er wiederum in kleinere virtuelle Laufwerke zerlegen kann. In der Abbildung wird ein virtuelles Laufwerk direkt von einem Datenbanksystem benutzt, das andere wird zwischen zwei Dateisystemen aufgeteilt.

Vergleich: Volume Manager versus intelligente Disksysteme

Der Volume Manager kann je nach Ausstattung die gleichen Funktionen bereitstellen wie ein RAID-Controller (Abschnitt 2.2) oder ein intelligentes Disk- oder Flashsystem (Abschnitt 2.4). Ähnlich wie bei den Snapshots gilt auch hier, dass die Realisierung von Funktionen wie RAID, Instant Copies und Remote Mirroring im Volume Manager hardwareunabhängig ist. Gleichermaßen kann ein RAID-Controller oder intelligentes Disksystem die Ressourcen des Servers entlasten, wenn die entsprechenden Funktionen auf die Speichergeräte ausgelagert werden. Die Realisierung von RAID im Volume Manager belastet nicht nur die CPU des Servers, sondern auch dessen Busse (Abb. 4–4).

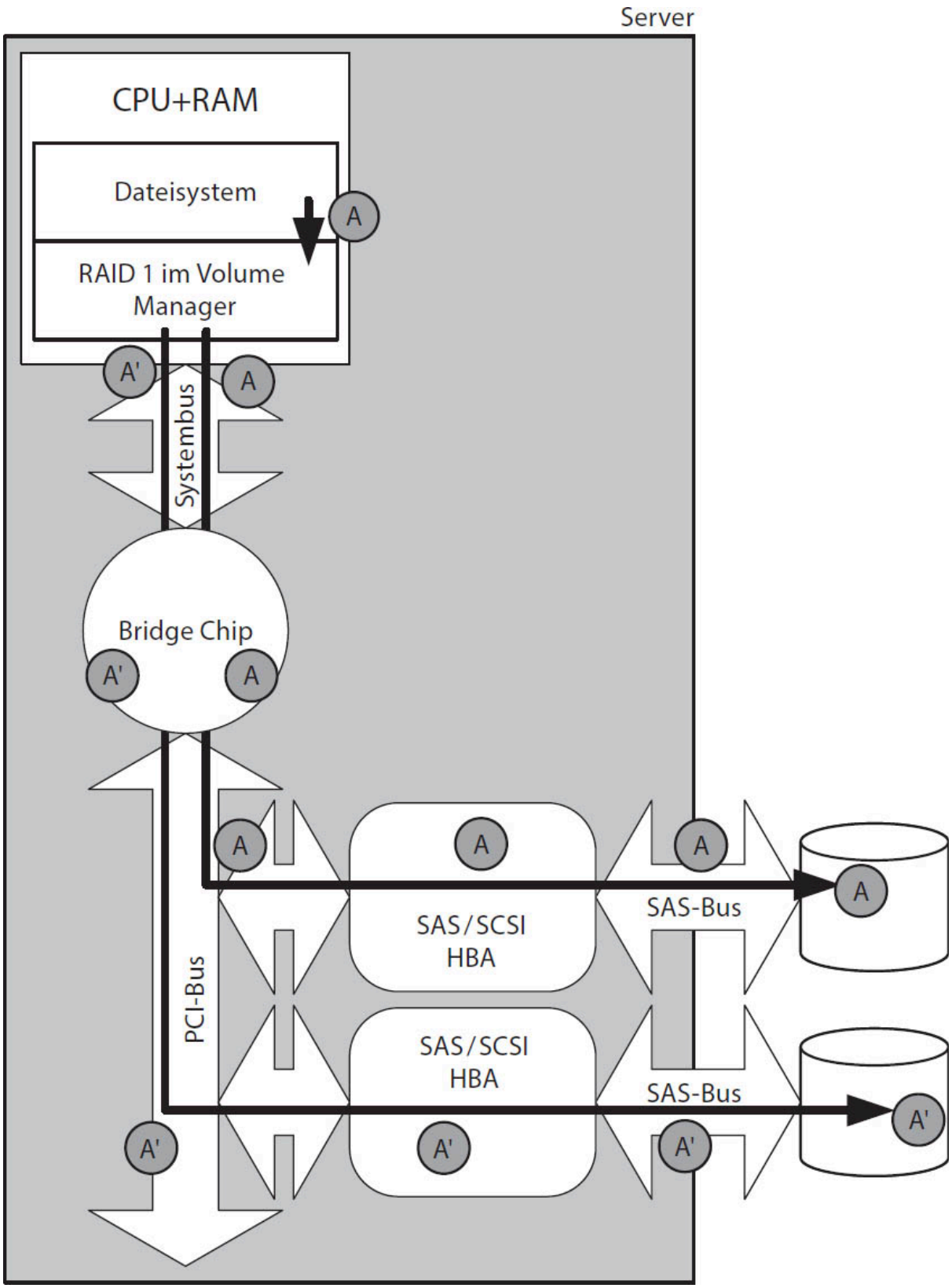

Abb. 4–4 **RAID im Volume Manager**

RAID im Volume Manager belastet Busse und CPU des Servers. Zum Beispiel muss bei RAID 1 jeder vom Dateisystem geschriebene Block zweimal durch alle Busse geschleust werden.

4.1.5 Information Lifecycle Management (ILM)

Integration mehrerer Medientypen

Moderne Dateisysteme erlauben es, verschiedene Medientypen (Flashmodule, SSDs, Festplatten, Objektspeicher, Bänder) zu integrieren, die meist von unterschiedlichen Speichersystemen bereitgestellt werden. Die verschiedenen Medientypen besitzen sehr unterschiedliche Eigenschaften, insbesondere was die Antwortzeit und die Geschwindigkeit beim Lesen und Schreiben anbelangt (Abschnitt 1.1). Schnelle Medien wie Flashspeicher sind meist teurer. Hochkapazitive Festplatten und Bänder erlauben dagegen eine kostengünstigere Speicherung, mit jedoch langsameren Zugriffszeiten. Für sehr große Dateisysteme ergibt sich somit ein erhebliches Einsparpotenzial, wenn man in ein Dateisystem mehrere Medientypen integriert und die Dateien je nach Leistungsanforderung auf einem dazu passenden Medientyp speichert.

Information Lifecycle Management (ILM)

Vor diesem Hintergrund dient das sogenannte Information Lifecycle Management (ILM) der Kostenoptimierung von Dateisystemen. Mit ILM-Funktionen ist es möglich, Daten innerhalb eines Dateisystems auf dem jeweils optimalen Medium zu speichern beziehungsweise dorthin zu verlagern. Dies erlaubt es Administratoren, die spezifischen Leistungsanforderungen beim Zugriff auf Daten gegen den Wert der gespeicherten Informationen abzuwägen und entsprechend zu gestalten. ILM stellt somit ähnliche Funktionen bereit wie die automatische Speicherortverlagerung von Disksystemen (Abschnitt 2.5.3).

Regelbasierte Ablage und Migration

Dateisysteme, die ILM-Funktionen unterstützen, legen Dateien gemäß vom Administrator konfigurierbarer Regeln ab. So könnte eine solche Regel beispielsweise bewirken, dass Dateien einer bestimmten Datenbankanwendung auf Flashmodulen gespeichert werden, während alle anderen Dateien auf langsameren, dafür aber günstigeren Festplatten abgelegt werden. Da die Anforderungen an die Zugriffsgeschwindigkeit von Daten jedoch meist nicht konstant bleiben, sondern sich vielmehr über den Lebenszyklus einer Datei ändern, kann es vorteilhaft sein, Daten von einem Medium auf ein anderes zu verschieben. Ein Administrator könnte beispielsweise veranlassen, dass neue Dateien zunächst auf einem schnellen Medium abgelegt werden und dass nach einiger Zeit, wenn die Daten nur noch selten oder gar nicht mehr gelesen oder verändert werden, auf ein Band verlagert werden, wo sie über viele Jahre hinweg sehr kostengünstig verweilen können.

Voraussetzung: einheitlicher Namensraum

Damit eine solche Verlagerung (Migration) von Dateien von einem Medium auf ein anders realisiert werden kann, ist es notwendig, dass die Dateien zu jedem Zeitpunkt von Anwendern und Anwendungen zugreifbar bleiben – unabhängig davon, auf welchem Medium sie letztendlich abgelegt sind. Ein Anwender erwartet, eine Datei an genau demselben Ort (Verzeichnis) und unter demselben Namen innerhalb eines Dateisystems wiederzufinden, an dem er sie zuvor gespeichert hat, auch wenn die Daten zwischenzeitlich auf ein anderes Medium verlagert wurden. Eine wichtige Voraussetzung für die Verwendung von unterschiedlichen Medien innerhalb eines Dateisystems ist daher ein einheitlicher Namensraum. Bei einer Verlagerung von einem Medium auf ein anderes ändert sich naturgemäß die Zugriffsgeschwindigkeit, keinesfalls jedoch Informationen wie der Name oder der Pfad von Dateien und Verzeichnissen. Die Kostenvorteile von ILM lassen sich nur dann voll ausschöpfen, wenn Dateisysteme beim Zugriff auf Dateien den darunter liegenden Speicherort und damit die verwendeten Medien verbergen können.

4.1.6 Dateisysteme und Datenbanken

Dateisysteme als generischer Dienst

Dateisysteme und Volume Manager stellen ihre Dienste für eine Vielzahl von Anwendungen mit unterschiedlichen Lastprofilen zur Verfügung. Das heißt, sie sind eine generische Anwendung. Ihre Leistung ist in der Regel nicht für eine bestimmte Anwendung optimiert.

Dateisysteme und Datenbanksysteme

Datenbanksysteme wie DB2 oder Oracle können das Dateisystem umgehen und die Blöcke auf physischen oder virtuellen Laufwerken selbst verwalten (Abb. 4–5). Dadurch kann man zwar die Leistung des Datenbanksystems steigern, seine Verwaltung wird aber aufwendiger. In der Praxis werden Datenbanksysteme deshalb meist so konfiguriert, dass sie ihre Daten in Dateien ablegen, die von einem Dateisystem verwaltet werden. Wird für ein spezielles Datenbanksystem mehr Leistung benötigt, so investieren Datenbankadministratoren in der Regel lieber in leistungsfähigere Hardware, um so den erhöhten Administrationsaufwand für den unmittelbaren Betrieb des Datenbanksystems auf den blockorientierten Laufwerken zu vermeiden.

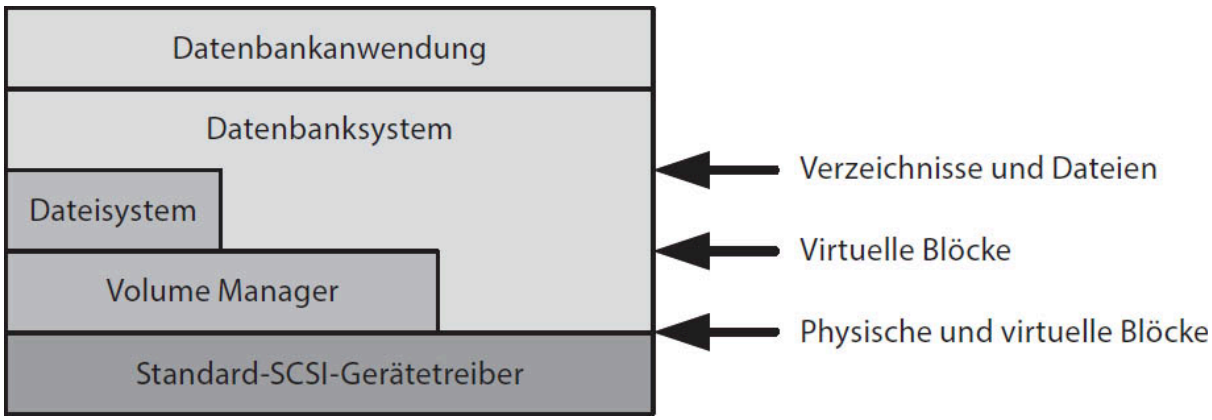

Abb. 4–5 **Dateisysteme und Datenbanksysteme**
Zur Leistungssteigerung können Datenbanksysteme das Dateisystem umgehen und die Blöcke der Laufwerke selbst verwalten.

4.2 Netzwerk-Dateisysteme und Fileserver

Gliederung des Abschnitts

Netzwerk-Dateisysteme sind die natürliche Fortsetzung lokaler Dateisysteme: Mehrere Anwender und Anwendungen können über ein Netzwerk-Dateisystem auf die gleichen Verzeichnisse und Dateien zugreifen, die physisch auf einem anderen Server, dem Fileserver, liegen (Abschnitt 4.2.1). Fileserver stellen Dateisysteme über das Netzwerk bereit. Vorkonfigurierte Fileserver werden als Network Attached Storage (NAS) bezeichnet (Abschnitt 4.2.2). Neben Netzwerk-Dateisystemen und Fileservern gibt es zahlreiche Alternativen, um Dateien über das Netz auszutauschen (Abschnitt 4.2.3).

4.2.1 Grundprinzip

File Sharing im LAN: Netzwerk-Dateisysteme

Die Metapher von Verzeichnissen und Dateien zur Verwaltung von Daten ist so eingängig, dass sie lange Zeit das vorherrschende Modell für den Zugriff auf Daten über Rechnernetze war. Sogenannte Netzwerk-Dateisysteme ermöglichen mehreren Anwendern und Anwendungen Zugriff auf dieselben Daten, die auf einem zentralen Server abgelegt sind (Abb. 4–6).

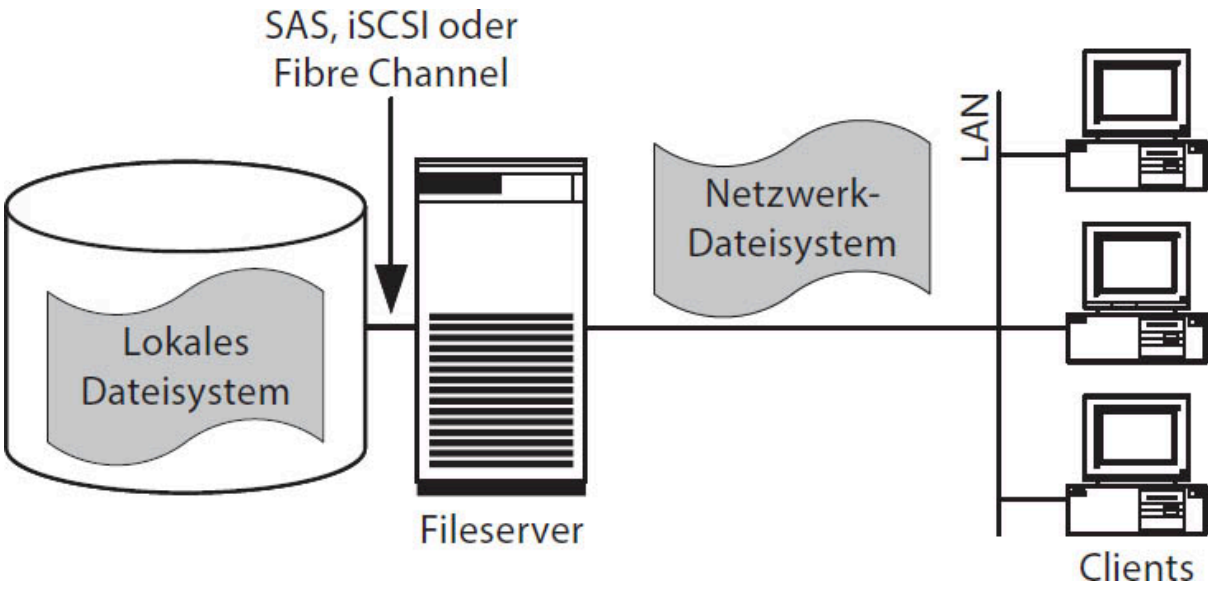

Abb. 4–6 Netzwerk-Dateisysteme

Netzwerk-Dateisysteme machen lokale Dateien und Verzeichnisse über das LAN verfügbar. Mehrere Anwender können so auf gemeinsamen Dateien (zum Beispiel Projektdaten oder Quellcode) arbeiten.

NFS und SMB

Das erste weitverbreitete Netzwerk-Dateisystem war das von Sun Microsystems entwickelte Network File System (NFS), das heute auf allen Unix-Systemen das Standard-Netzwerk-Dateisystem ist. Microsoft entwickelte zusammen mit IBM für seine Windows-Betriebssysteme mit dem Common Internet File System (CIFS), genauer mit dem Server-Message-Block-(SMB-)Protokoll, ein eigenes Netzwerk-Dateisystem, das zu NFS inkompatibel ist.

Entfernter Zugriff bleibt verborgen

Anwender und Anwendungen können mithilfe von Netzwerk-Dateisystemen von verschiedenen Rechnern aus auf einem gemeinsamen Datenbestand arbeiten. Auf Unix-Rechnern muss der Systemadministrator dazu ein von einem NFS-Server exportiertes Dateisystem mit dem »mount«-Befehl in die lokale Verzeichnisstruktur des NFS-Clients einbinden. Auf Windows-Rechnern kann jeder Anwender dies selbst tun, indem er ein »Netzwerklaufwerk verbindet« (Map Network Drive). Anschließend ist der Zugriff auf Dateien, die in einem Netzwerk-Dateisystem liegen, genauso möglich wie der Zugriff auf Dateien im lokalen Dateisystem. Der Anwender merkt den Unterschied bis auf mögliche Leistungsunterschiede nicht, solange das Netzwerk zwischen dem NFS-Client und NFS-Server verfügbar ist. Ist die Netzverbindung unterbrochen, so steht das Netzwerk-Dateisystem nicht mehr zur Verfügung.

4.2.2 Network Attached Storage (NAS)

Definition: Network Attached Storage (NAS)

Fileserver sind in heutigen IT-Umgebungen so wichtig, dass sie sich zu einer eigenständigen Produktgruppe entwickelt haben. Sogenannter Network Attached Storage (NAS) bezeichnet vorkonfigurierte Fileserver. NAS-Systeme bestehen aus einem oder mehreren internen Servern, vorkonfigurierter Laufwerkskapazität und meist einem abgespeckten oder einem speziellen Betriebssystem (Abb. 4–7).

Einsatz

NAS-Systeme werden meist über Ethernet an das LAN angeschlossen, wo sie ihre Speicherkapazität als Netzwerk-Dateisystem zur Verfügung stellen. Klassische Geschäftsanwendungen, die eine Verarbeitung von unstrukturierten Daten erfordern, organisieren ihren Speicherplatz meist in Verzeichnissen und Dateien. Über die Einbindung von Netzwerk-Dateisystemen lassen sich die Daten solcher Anwendungen auf NAS-Systeme verlagern, ohne dass spezielle Anpassungen der Anwendung nötig sind. Netzwerk-Dateisysteme ergänzen oder ersetzen in einem solchen Szenario die lokalen Dateisysteme und ermöglichen dadurch, Speicher für klassische Anwendungen flexibel zu erweitern. Große NAS-Systeme bieten zusätzliche Funktionen wie Snapshots, Remote Mirroring und Backup über Fibre Channel SAN an.

Optimiert für File Sharing

NAS-Systeme wurden speziell für das File Sharing entwickelt. Dies hat zwei Vorteile: Weil der Einsatzzweck von NAS-Systemen definitionsgemäß bekannt ist, kann man NAS-Betriebssysteme wesentlich besser optimieren als universelle Betriebssysteme. NAS-Systeme können deshalb schneller sein als Fileserver auf vergleichbarer Hardware, die auf einem universellen Betriebssystem aufsetzen.

Einfache Installation, geringer Wartungsaufwand

Der zweite Vorteil von NAS ist, dass NAS-Systeme Plug&Play-Dateisysteme bereitstellen: Anschließen – Hochfahren – Benutzen. Im Vergleich zu einem universellen Betriebssystem werden Funktionen entfernt, die nicht für das Netzwerk-Dateisystem notwendig sind. Dafür werden Funktionen hinzugefügt, die seine Benutzung erleichtern. NAS-Systeme können deshalb durch geringen Installations- und Wartungsaufwand glänzen, was Systemadministratoren entlastet.

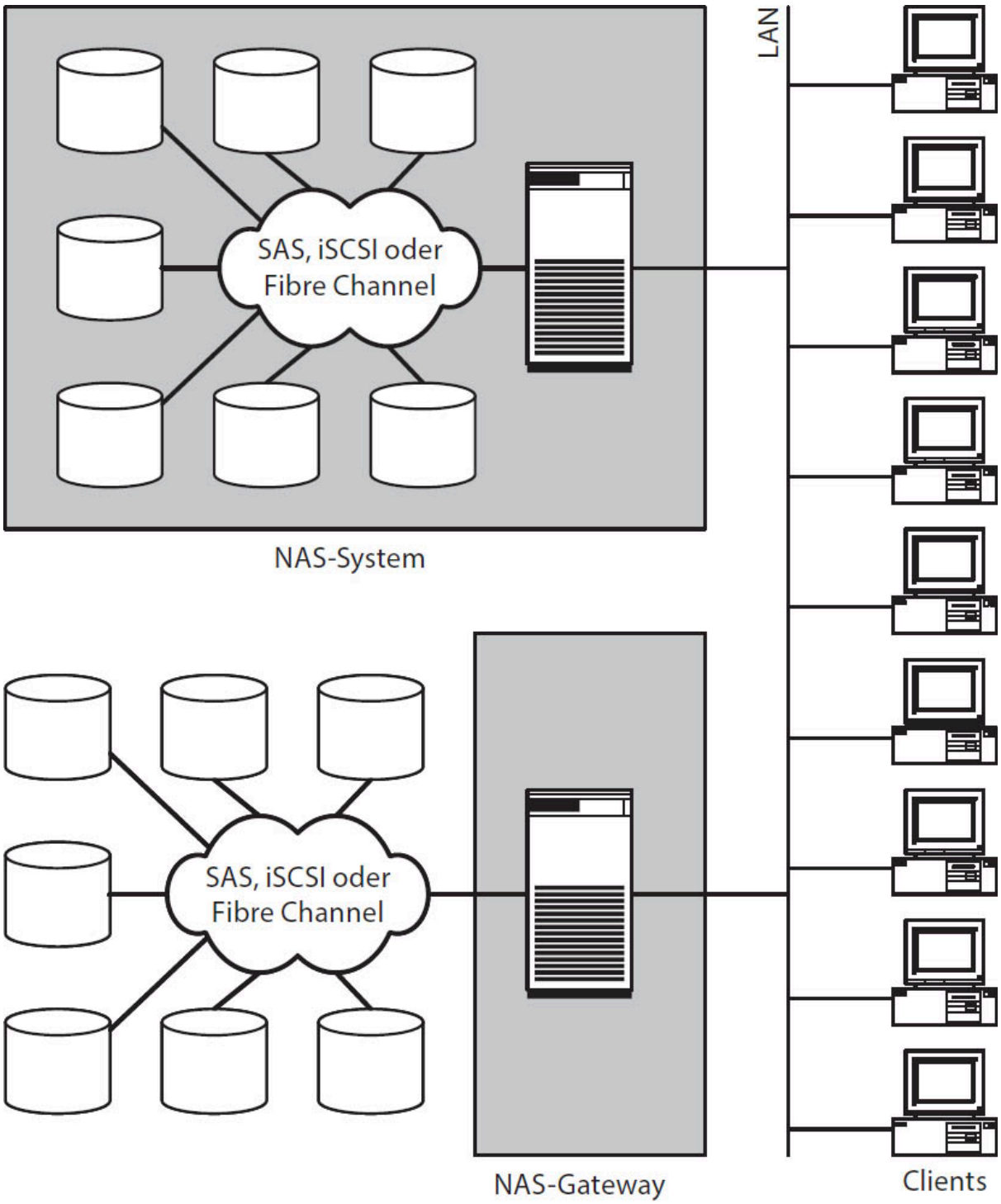

Abb. 4–7 **NAS-System und NAS-Gateway**

Ein NAS-System ist ein vorkonfigurierter Fileserver mit internen Laufwerken, der seine Speicherkapazität über das LAN zur Verfügung stellt. Ein NAS-Gateway ist ein vorkonfigurierter Fileserver, der im Speichernetz verfügbare Speicherkapazität über das LAN bereitstellt.

Skalierbarkeit!?

NAS-Systeme sind gut skalierbar. Beispielsweise kann ein Systemadministrator für jedes Projekt oder für jede Abteilung ein eigenes NAS-System aufsetzen. Auf diese Weise lässt sich

schnell weiterer Speicherplatz bereitstellen. Allerdings erzeugen neue Anwendungen des Pervasive Computings (Kap. 9) wie Internet of Things und Big Data so große Datenmengen, dass sehr viele NAS-Systeme benötigt werden. Obwohl ein einzelner vorkonfigurierter Fileserver leicht zu administrieren ist, steigt der Verwaltungsaufwand mit jedem zusätzlichen NAS-System an.

Namensraum?

Einen weiteren Nachteil stellt die Tatsache dar, dass mehrere unabhängige NAS-Systeme in der Regel keinen einheitlichen Namensraum bereitstellen. Soll beispielsweise eine Anwendung von einem auf ein anderes NAS-System verlagert werden, etwa weil dort noch genügend freier Speicher vorhanden ist, müssen unweigerlich Anpassungen an der Anwendung vorgenommen werden. Diese muss nach der Verlagerung nämlich über einen anderen Pfad auf ihre Daten zugreifen. Wenn sehr viele verschiedene Anwendungen eingesetzt werden, kann es sich hierbei ebenfalls um einen nicht zu unterschätzenden Verwaltungsaufwand handeln.

Upgrade-Pfad unklar

Ein weiterer Nachteil von NAS-Systemen besteht in ihrem unklaren Upgrade-Pfad: Beispielsweise kann der interne Server nicht einfach durch einen leistungsstärkeren Server ausgetauscht werden, weil dies gegen das Prinzip des vorkonfigurierten Fileservers verstößt. Hier ist man ganz den Upgrade-Optionen ausgeliefert, die der Hersteller des betreffenden NAS-Systems anbietet.

4.2.3 Alternativen zu Netzwerk-Dateisystemen

File Sharing im Internet: FTP

Lange vor dem World Wide Web stand mit dem File Transfer Protocol (FTP) ein Mechanismus zur Verfügung, mit dem Benutzer über das Internet Dateien austauschen konnten. Auch heute noch sind FTP-Server ein verbreitetes Mittel zur Verteilung von frei verfügbarer Software und frei verfügbaren Dokumenten. Der Zugriff auf FTP-Server ist im Gegensatz zu Netzwerk-Dateisystemen jedoch aus Sicht des Anwenders anders: Es wird ein spezieller FTP-Client benötigt, mit dem er Dateien zwischen dem FTP-Server und dem lokalen Rechner hin und her kopieren kann.

Das World Wide Web (WWW)

Die Hypertext Markup Language (HTML) und das Hypertext Transfer Protocol (HTTP) änderten in den 1990er Jahren das Nutzungsmodell des Internets radikal: Die Daten im Internet sind im Gegensatz zu FTP über HTML-Dokumente miteinander verknüpft. Der Benutzer im Internet greift heute nicht mehr auf einzelne Dateien zu; stattdessen »surft« er im World Wide Web (WWW). Er schaut sich in seinem Browser HTML-Dokumente an, die statisch in Form von Dateien auf einem HTTP-Server

(Webserver) liegen oder dynamisch erzeugt werden. Grafische HTTP-Clients, die Browser, haben einen FTP-Client integriert, mit dem sich Dateien bequem »downloaden« lassen.

Enterprise File Sync&Share (EFSS)

Eine weitere Neuerung für die gemeinsame Nutzung von Dateien ist das Enterprise File Sync&Share (EFSS). EFSS ersetzt Fileserver durch einen Dienst in der Cloud wie zum Beispiel Dropbox, über den viele Benutzer gemeinsame Dokumente verwalten und austauschen können. EFSS stellen wir später in Abschnitt 9.4.2 vor.

Objektspeicher

Schließlich ist bei den Alternativen zu Netzwerk-Dateisystemen noch Objektspeicher zu nennen. Objektspeicher werden wie Fileserver an das Rechnernetz angeschlossen und ermöglichen es vielen Benutzern und Anwendungen, Daten abzulegen und austauschen. In vielen Einsatzbereichen wird heute Objektspeicher eingesetzt, wo früher Fileserver verwendet wurden. Objektspeicher beschreiben wir ausführlich in Kapitel 6.

4.3 Authentisierung und Autorisierung

Motivation

Dateisysteme ermöglichen es Benutzern und Anwendungen, auf gemeinsame Dateien zuzugreifen. Dazu muss sich ein Benutzer zunächst im Betriebssystem anmelden, indem er in der Regel seinen Benutzernamen und sein Passwort eingibt. Mithilfe des Passworts stellt das Betriebssystem sicher, dass der Benutzer tatsächlich die Person ist, für die er sich ausgibt. Nach erfolgreicher Authentisierung überwacht das Betriebssystem mittels Access Control Lists (ACLs) den Zugriff auf Dateien und Verzeichnisse und gestattet nur erlaubte Zugriffe (Autorisierung).

Gliederung des Abschnitts

Das Betriebssystem muss also die Identität der Benutzer feststellen (Identifizierung, Abschnitt 4.3.1) und überprüfen (Authentisierung, Abschnitt 4.3.2), um die in den Dateien gespeicherten Informationen vor unerlaubten Zugriffen zu schützen. In größeren Umgebungen helfen hierbei Verzeichnisdienste wie LDAP und Active Directory (Abschnitt 4.3.3). Nachdem die Identität eines Benutzers überprüft und anerkannt wurde, muss noch die Legitimität von Zugriffen des Benutzers auf das Dateisystem geprüft und je nach Befugnis gestattet oder unterbunden werden. Dieser Schritt wird als Autorisierung bezeichnet und in der Regel über Access Control Lists (ACLs) gesteuert (Abschnitt 4.3.4).

4.3.1 Identifizierung

Identifizierung

Ziel der Identifizierung ist es, die Identität von Benutzern und Infrastrukturkomponenten eindeutig sicherzustellen – und zwar unabhängig davon, auf welchem System ein Benutzer gerade angemeldet ist. Mechanismen, um dies zu realisieren, basieren meist auf numerischen Benutzerkennungen, sogenannten Identifiern (IDs). Alternativ können auch digitale Fingerabdrücke zum Einsatz kommen, wie beispielsweise öffentliche SSH-Schlüssel oder Zertifikate von Webservern.

Identifizierung in Unix-Systeme

Unix-Systeme verwenden Benutzernamen und numerische Benutzerkennungen (Unique Identifiers, UIDs) um Benutzer eines Systems zu repräsentieren. Der Benutzername wird dabei vom Betriebssystem eindeutig einer Benutzerkennung zugeordnet. Der Benutzername (zum Beispiel »troppens«) ist eine menschenlesbare und verständliche alphanumerische Bezeichnung, die Benutzerkennung eine positive Ganzzahl (zum Beispiel »1602«). Meldet sich ein Benutzer mit seinem Benutzernamen an einem Unix-System an, so ermittelt das Betriebssystem zunächst die UID und verwendet diese dann im Folgenden, um den Benutzer zu repräsentieren.

Zuordnung von Benutzernamen und Benutzerkennung

Benutzernamen, Benutzerkennungen sowie deren Zuordnung werden entweder lokal im Betriebssystem gespeichert oder aber in einem zentralen Verzeichnisdienst hinterlegt. Beispiele für solche Verzeichnisdienste sind Active Directory, LDAP und NIS (Abschnitt 4.3.2).

Benutzergruppen

Unix-Systeme verwenden Gruppen, um eine Menge von Benutzern mit gleichen Eigenschaften und gleichen Rechten zu verwalten. Ähnlich wie Benutzernamen und Benutzerkennungen (UIDs) verwenden Unix-Systeme Gruppennamen und Gruppenkennungen (Group Identifier, GIDs). Ein Benutzer kann dabei Mitglied einer oder mehrerer Gruppen sein, wobei genau eine Gruppe die primäre Gruppe darstellt. Neue Dateien, die ein Benutzer erstellt, sind meist sowohl von dem Benutzer selbst als auch von allen anderen Mitgliedern der primären Gruppe zugreifbar. Unix-Gruppen können nicht verschachtelt werden – eine Gruppe enthält ausschließlich Benutzer, aber keine anderen Gruppen. Gruppennamen und deren Zuordnung zu Gruppenkennungen werden wiederum entweder lokal im Betriebssystem gespeichert oder in zentralen Verzeichnisdiensten wie Active Directory, LDAP oder NIS hinterlegt.

Identifizierung in Windows-Systemen

Windows-Systeme verstehen alle Einheiten als Ressourcen, einschließlich Benutzern, Gruppen und Computern. Jede Ressource wird dabei über einen sogenannten Security Identifier (SID) repräsentiert. SIDs sind eindeutige, alphanumerische Zeichenketten (zum Beispiel »S-1-5-21-3876423458-1234742343-4234987547«). Windows-Gruppen können, im Gegensatz zu Unix-Gruppen, verschachtelt werden: Eine Gruppe kann somit einen oder mehrere Benutzer enthalten, aber auch eine oder mehrere Gruppen. Wie bei Unix-Systemen können auch in Windows-Systemen die Namen und Security Identifier (SIDs) von Ressourcen lokal gespeichert werden oder alternativ können zentrale Verzeichnisdienste wie Active Directory oder LDAP (mit Samba-Schema) verwendet werden.

ID Mapping für Windowsund Unix-Systeme

Soll nun von mehreren Windows- und Unix-Systemen auf einen gemeinsamen Datenbestand in einem Netzwerk-Dateisystem zugegriffen werden, so muss sichergestellt werden, dass alle Systeme eine einheitliche, konsistente Zuordnung von Benutzern, Gruppen sowie deren Identifiern (IDs) verwenden. Insbesondere müssen dabei Windows Security Identifier (SIDs) in die entsprechenden Unix-Kennungen (UIDs und GIDs) überführt werden, und umgekehrt. Ein Benutzer, der von einem Windows-System auf seine Daten zugreift, erwartet, die gleichen Rechte zu besitzen wie am Tag zuvor, als er die Daten von einem Unix-System aus geschrieben hat. Um dies zu gewährleisten, müssen alle zugreifenden Systeme die gleiche Zuordnung von Benutzer- und Gruppenkennungen sicherstellen. Dieses sogenannte ID Mapping kann entweder auf allen Systemen lokal oder in zentralen Verzeichnisdiensten wie Active Directory, LDAP und NIS gespeichert werden.

4.3.2 Authentisierung

Authentisierung

Die Identität der zugreifenden Benutzer und Anwendungen muss überprüft werden, um Dateien und Verzeichnisse vor unerlaubten Zugriffen zu schützen. Dazu muss sichergestellt werden, dass diese nur die für sie vorgesehene Identität nutzen können, aber keine beliebigen anderen. Ziel von Authentisierung ist es daher, die Identität, die Benutzer und Infrastrukturkomponenten für sich beanspruchen, zu überprüfen und zu bestätigen. Mechanismen, um dies zu realisieren, basieren meist auf Passwörtern oder dem Austausch digitaler Schlüssel.

Authentisierung in Unix-Systemen

Authentisierung in Unix-Umgebungen ist traditionell maschinenbasiert. Wenn sich ein Anwender an einem Unix-Server mittels Benutzernamen und Passwort anmeldet, also eine bestimmte Identität für sich beansprucht, dann überprüft das Betriebssystem die eingegebenen Werte entweder lokal oder mithilfe von zentralen Authentisierungsdiensten, wie zum Beispiel LDAP. Anschließend ermittelt es die eindeutige Kennung des Benutzers. Von nun an wird der Benutzer auf dem Unix-System selbst als vertrauenswürdig eingestuft, genau wie auf allen anderen Ressourcen, die dem Unix-System vertrauen. Greift der Benutzer nun auf das am Unix-System angeschlossene Netzwerk-Dateisystem über das NFS-Protokoll zu, übermittelt der NFS-Client die Kennung des Benutzers (UID) und die Kennung seiner Primärgruppe (GID) sowie die Kennung aller weiteren Gruppen (GIDs), in denen der Benutzer Mitglied ist. Der NFS-Server, der dem Unix-System vertraut, stuft den Benutzer automatisch als bereits authentisiert ein. Der NFS-Server verlässt sich somit auf den NFS-Client und geht davon aus, dass dieser die Identität des Benutzers entsprechend überprüft hat.

Authentisierung in Windows-Systemen

Im Gegensatz dazu ist Authentisierung in Windows-Umgebungen sitzungsbasiert. Wenn sich ein Anwender mittels Benutzernamen und Passwort an einem Windows-Server anmeldet, also eine bestimmte Identität für sich beansprucht, dann überprüft wiederum das Betriebssystem die eingegebenen Werte entweder lokal oder mittels Authentisierungsdiensten wie Active Directory oder genauer dem in Active Directory enthaltenen Kerberos. Der Benutzer wird daraufhin zwar auf dem Windows-System selbst als vertrauenswürdig eingestuft, muss sich aber dennoch an anderen Netzwerk-Diensten wie einem SMB-Server erneut anmelden, um diese zu nutzen.

Beispiel SMB-Protokoll

In Windows-Umgebungen findet der Zugriff auf Netzwerk-Dateisysteme normalerweise über das SMB-Protokoll statt. Der SMB-Client übermittelt dabei die Anmeldedaten des Benutzers zunächst an einen Authentisierungsdienst (in der Regel Active Directory), der die Anmeldedaten überprüft und bei Erfolg ein verschlüsseltes Zertifikat (Ticket) ausstellt. Dieses Ticket kann anschließend vom SMB-Client verwendet werden, um sich am SMB-Server anzumelden. Die im Ticket enthaltenen Informationen erlauben es dem SMB-Server, den Benutzer eindeutig zu identifizieren. Der SMB-Server entschlüsselt das Ticket und überprüft dessen Herkunft sowie die Rechtmäßigkeit des Zugriffs.

4.3.3 Verzeichnisdienste

Verzeichnis- und Authentisierungsdienste

Wie in den vorangegangenen Abschnitten (Abschnitte 4.3.1 und 4.3.2) dargestellt, spielen zentrale Verzeichnis- und Authentisierungsdienste eine wichtige Rolle beim Zugriff auf Netzwerk-Dateisysteme. Zentrale Verzeichnisdienste speichern – ähnlich wie ein Telefonbuch – öffentliche Informationen über Benutzer und Infrastrukturkomponenten, die von vielen Systemen genutzt werden können, um diese zu identifizieren. Authentisierungsdienste erlauben es, die Identitäten von Benutzern und Infrastrukturkomponenten, die im Verzeichnisdienst hinterlegt sind, zu überprüfen und zu bestätigen.

Lokale Speicherung von Anmeldeinformationen

Zentrale Verzeichnis- und Authentisierungsdienste sind meist optional. Grundsätzlich können die Zuordnung von Benutzernamen zu Benutzerkennungen und die zur Anmeldung erforderlichen Passwörter lokal im Betriebssystem des jeweiligen Rechners gespeichert werden. Die Beispiele in den vorherigen Abschnitten (Abschnitte 4.3.1 und 4.3.2) zeigen, dass eine eindeutige Zuordnung von Benutzern und Gruppen auf allen Systemen wünschenswert ist, genauso wie ein einzelnes Passwort, mit dem sich ein Benutzer an allen Netzwerk-Diensten anmelden kann. Während in kleinen Umgebungen mit wenigen Servern ein manueller Abgleich dieser Informationen zwischen den einzelnen Systemen noch denkbar ist, skaliert ein solcher Ansatz nicht in großen Umgebungen mit Hunderten oder gar Tausenden von Servern und Benutzern.

Vorteile zentraler Speicherung

Verzeichnisdienste erlauben es, diese Benutzer- und Gruppeninformation an zentraler Stelle zu hinterlegen. Andere Dienste und deren Clients können auf die zentralen Informationen zugreifen, anstatt die nötigen Daten jeweils lokal speichern zu müssen. Somit entfällt die Notwendigkeit des manuellen Abgleichs zwischen den einzelnen Systemen. Im Folgenden stellen wir einige wichtige Verzeichnis- und Authentisierungsdienste vor.

Network Information Service (NIS)

Ein Beispiel für einen zentralen Verzeichnisdienst ist etwa der Network Information Service (NIS). Bei NIS werden Konfigurationsinformationen wie beispielsweise Benutzernamen und Benutzerkennungen auf dem NIS-Server in Tabellen hinterlegt, den sogenannten NIS Maps. Diese können auf anderen Servern als globale Benutzerlisten eingebunden werden und ergänzen oder ersetzen somit die lokalen Konfigurationsdateien.

Lightweight Directory Access Protocol (LDAP)

Während NIS die Konfigurationsinformationen in Tabellen verwaltet, basiert beispielsweise das Lightweight Directory Access Protocol (LDAP) auf hierarchischen Baumstrukturen. Die in diesen Strukturen gespeicherten Informationen werden über sogenannte LDAP-Schemata definiert und können durch Einspielen eines neuen Schemas erweitert werden. Dies erlaubt eine deutlich flexiblere Gestaltung des Verzeichnisdienstes. So können organisatorische Strukturen in einem LDAP-Dienst beispielsweise ineinander verschachtelt werden und bestimmte Informationen können innerhalb dieser Strukturen vererbt werden. Da dies auch die Abbildung komplexer Unternehmensstrukturen mit Abteilungen und Unterabteilungen erlaubt, hat LDAP heute NIS in weiten Teilen abgelöst.

Active Directory (AD)

Insbesondere Windows-Umgebungen basieren meist auf Active Directory (AD), einer Komponente von Microsofts Windows Server. Active Directory vereint dabei sowohl einen Verzeichnisdienst als auch Authentisierungs- und Namensdienste. Der Authentisierungsdienst basiert dabei auf dem sitzungsbasierten Kerberos-Protokoll, das verschiedene Arten von digitalen Tickets beschreibt. Kerberos-Implementierungen existieren auch außerhalb von Active Directory und gewährleisten etwa in Unix-Umgebungen eine deutlich höhere Sicherheit beim Zugriff auf Netzwerk-Dienste.

4.3.4 Autorisierung und Zugriffskontrolle

Motivation

Nachdem ein Benutzer sich mit seinem Benutzernamen an einen Rechner angemeldet hat (Abschnitt 4.3.1) und seine Identität mittels seines Passworts authentisiert wurde (Abschnitt 4.3.2), stellt sich anschließend die Frage, auf welche Bereiche des Systems er Zugriff erhält. Dies betrifft auch Dateien und Verzeichnisse, die in lokalen oder in lokal eingehängten Netzwerk-Dateisystemen gespeichert sind.

Autorisierung

Die sogenannte Autorisierung dient der Steuerung von Zugriffen auf Dateien und Verzeichnisse und findet grundsätzlich erst nach der erfolgreichen Authentisierung statt. Ziel von Autorisierung ist es, authentisierten Benutzern den Zugriff auf bestimmte Ressourcen entweder zu gewähren oder zu verweigern. Mittels Zugriffskontrolle muss dabei sichergestellt werden, dass nur authentisierte und autorisierte Benutzer Zugang zu den jeweiligen Ressourcen erlangen. Die Autorisierung dient somit auch dem Schutz von Dateien und Verzeichnissen vor unerlaubten Zugriffen.

POSIX Mode Bits

Ein wichtiges Grundprinzip von modernen Dateisystemen besteht in der Unterscheidung zwischen berechtigten und unberechtigten Zugriffen. Während berechtigte Zugriffe auf Daten sichergestellt werden müssen, gilt es, unberechtigte Zugriffe zu unterbinden. Hierzu wurden in Unix-Systemen ursprünglich die sogenannten POSIX Mode Bits eingesetzt. Dabei handelt es sich um Attribute, die für jede Datei und für jedes Verzeichnis beschreiben, welche Benutzer zu welchen Operationen berechtigt sind. Der erste Eintrag gewährt dem Besitzer des Objekts bestimmte Rechte (Owner), der zweite Eintrag beschreibt die Rechte für die Gruppe (Group), und der dritte Eintrag berechtigt alle anderen Benutzer (Others). Über POSIX Mode Bits können die entsprechenden Benutzer jeweils zum Lesen (Read), zum Schreiben (Write) oder zum Ausführen (Execute) einer Datei berechtigt werden (Abb. 4–8). Das Betriebssystem prüft im Zuge der Autorisierung bei jedem Zugriff auf eine Datei oder ein Verzeichnis, ob dieser Zugriff gemäß der Mode Bits zu erlauben oder zu verweigern ist.

```
$ ls -l directory

total 16

drwxrwxr-x 2 user1 proj1    6 Feb 16 06:35 dir1

drwx-- 2 user1 proj1   70 Feb 16 06:36 dir2

-rw-rw-r— 1 user1 proj1 3678 Feb 16 06:38 file1

-rw-r-— 1 user1 proj1 1884 Feb 16 06:37 file2

-rw--- 1 user1 proj1    0 Feb 16 06:25 file3

-rwxr-xr-x 1 user1 proj1 1834 Feb 16 06:37 script1

-r-x-- 1 user1 proj1  920 Feb 16 06:37 script2

$
```

Abb. 4–8 POSIX Mode Bits

Das ls-Kommando für Unix-Systeme zeigt die POSIX Mode Bits von Dateien und Verzeichnissen, z.B »rw-r-—« für file2. Die Datei »file2« darf von user1 gelesen und geschrieben werden (»rw-«). Andere Mitglieder der Group proj1 dürfen sie lesen (»r—«) und alle anderen Benutzer haben keinen Zugriff (»–«).

Access Control List (ACL)

In moderneren Dateisystemen werden statt Mode Bits sogenannte Access Control Lists (ACLs) eingesetzt. Üblicherweise besitzt jede Datei und jedes Verzeichnis eine eigene ACL. Diese »haftet« gewissermaßen an der Datei beziehungsweise dem Verzeichnis und beschreibt genau, welche Benutzer und Gruppen zu welchen Operationen berechtigt sind.

Access Control Entry(ACE)

Eine ACL besteht dabei aus einer oder mehreren sogenannten Access Control Entries (ACEs). Jede ACE beschreibt die Operationen, für die ein bestimmter Benutzer oder eine bestimmte Gruppe berechtigt ist. So könnte eine ACL für ein Verzeichnis beispielsweise zwei ACEs beinhalten: Die erste ACE gewährt einem Benutzer Vollzugriff auf das Verzeichnis, während die zweite ACE eine Gruppe ausschließlich zum Lesen berechtigt. Alle anderen Benutzer erhalten keinen Zugriff auf das Verzeichnis (Abb. 4–9). POSIX Mode Bits lassen sich, falls nötig, als ACL mit genau drei Einträgen darstellen (Abb. 4–10).

```
$ getfacl projekte/

  # file: projekte/

  # owner: root

  # group: wheel

    user:admin:rwxpDdaARWcCos:fd--I:allow

    group:users:r-x-a-R-c-:fd--I:allow
```

Abb. 4–9 Komplexe ACL

Eine ACL im NFSv4-Format auf einem FreeBSD-System. Der erste Eintrag gewährt dem Benutzer »admin« Vollzugriff, der zweite Eintrag berechtigt die Gruppe »users« lediglich zum Lesen. Alle anderen Benutzer erhalten keinen Zugriff.

```
$ getfacl file2

  # file: file2

  # owner: user1

  # group: proj1

    user::rw-

    group::r—

    other::-
```

Abb. 4–10 POSIX Mode Bits als ACL

POSIX Mode Bits können auch als ACL dargestellt werden. Die Abbildung zeigt die ACL-Darstellung der Datei »file2« aus Abbildung 4–8.

Vererbung

Insbesondere bei der Zugriffskontrolle von komplexen Verzeichnisbäumen hat sich ein weiterer Mechanismus durchgesetzt: Moderne Dateisysteme erlauben die Vererbung von ACEs. So ist es möglich, ACEs auf Verzeichnissen zu definieren, die dann sowohl für das Verzeichnis selbst als auch für alle darunter liegenden Unterverzeichnisse und Dateien gelten. Dies erlaubt es Administratoren, die Zugriffsrechte in großen Dateisystemen effizient zu steuern, und unberechtigte Zugriffe konsequent zu verhindern.

Problem: Interoperabilität von Protokollen

In den vorherigen Abschnitten haben wir erläutert, dass sich die Authentisierung von Windows- und Unix-Umgebungen beim Zugriff auf Netzwerk-Dateisysteme stark unterscheidet. Dies gilt in gleicher Weise auch für die Autorisierung. Während klassische Unix-Systeme nur sehr wenige, statisch vergebene Rechte für den Besitzer, die Gruppe und die Welt kennen, erlauben Windows-Systeme deutlich umfangreichere ACLs. Traditionelle POSIX-ACLs bestehen immer aus genau drei ACEs für Owner, Group und Other. Windows-ACLs können dagegen beliebig viele ACEs enthalten. Letztere können weiterhin etwa auf Vererbung basieren oder ineinander verschachtelte Gruppen vorsehen. Dies stellt eine besondere Herausforderung für Fileserver und NAS-Systeme dar, die einen

gemeinsamen, konsistenten Zugriff für Clients mit verschiedenen Betriebssystemen ermöglichen sollen.

Lösungsansätze

Im Laufe der Zeit wurden mehrere Ansätze entwickelt, trotz dieser Unterschiede Daten gleichzeitig über NFS und SMB bereitzustellen. Ein solcher Lösungsansatz sieht beispielsweise vor, für jedes unterstütze Netzwerk-Dateisystem einen eigenen Satz an Zugriffsrechten (ACLs) zu speichern und zu verwalten. Bei NFS und SMB hätte eine Datei somit zwei ACLs, eine für Unix und eine für Windows. Je nachdem über welches Protokoll ein Benutzer dann auf eine Datei zugreift, wird entweder die eine oder die andere ACL herangezogen. Dies hat den Vorteil, dass jeder Client die ganze Bandbreite an Funktionen des jeweiligen Protokolls verwenden kann und dabei stets eine (für das jeweilige Protokoll) 100% kompatible Zugriffskontrolle stattfindet. Der Nachteil dieses Ansatzes besteht natürlich darin, dass beide ACLs abgeglichen und gegebenenfalls aneinander angepasst werden müssen. Änderungen der ACLs müssen immer zweimal vorgenommen werden, damit die Benutzer in beiden Welten mit den gleichen Rechten ausgestattet sind.

Alternativen

Ein alternativer Lösungsansatz sieht daher stattdessen nur einen einzigen, gemeinsamen Satz an Zugriffsrechten vor. Soll dieser für Zugriffe über unterschiedliche Protokolle Gültigkeit besitzen, so ergeben sich gewisse Einschränkungen. Meist wird daher auf den Einsatz von Funktionen verzichtet, die nur in einem, nicht aber in allen Netzwerk-Dateisystemen verfügbar sind. Es wird sich gewissermaßen auf eine Teilmenge von Funktionen geeinigt, die für alle Protokolle verfügbar sind. Mächtige Funktionen wie zum Beispiel Vererbung können in diesem Szenario nicht verwendet werden, da beispielsweise traditionelle Unix-Systeme dieses Konzept nicht vorsehen.

4.4 Optimierung für verteilte Zugriffe

Gliederung des Abschnitts

Ein wesentlicher Nachteil von Fileservern und NAS-Systemen besteht in Leistungsengpässen für I/O-intensive Anwendungen wie Datenbanksysteme, Datensicherung, Batch-Prozesse oder Big-Data-Analysen. Wir werden zunächst diese Engpässe herausarbeiten (Abschnitt 4.4.1) und dann Möglichkeiten zur Beschleunigung von Netzwerk-Dateisystemen diskutieren (Abschnitt 4.4.2). Danach stellen wir mit dem Direct Access File System (DAFS) ein Netzwerk-Dateisystem vor, das statt auf TCP/IP auf Remote Direct Memory Access (RDMA) aufsetzt (Abschnitt 4.4.3). Im weiteren Verlauf des Abschnitts gehen wir mit den Shared-Disk-Dateisystemen (Abschnitt

4.4.4) und der Shared-Nothing-Architektur (Abschnitt 4.4.6) auf zwei weitere Ansätze ein, um verteilte Dateisystemzugriffe auf ganz unterschiedliche Weise zu optimieren. Für beide Ansätze stellen wir ein Beispiel näher vor, nämlich das General Parallel File System (GPFS, Abschnitt 4.4.5) und das Hadoop Distributed File System (HDFS, Abschnitt 4.4.7).

4.4.1 Leistungsengpässe in Fileservern

Schwachpunkt: File Sharing für I/O-intensive Anwendungen

Fileserver und NAS-Systeme stellen ihre Speicherkapazität über herkömmliche Netzwerk-Dateisysteme wie NFS und SMB oder über Internetprotokolle wie FTP und HTTPS bereit. Diese eignen sich zwar für das klassische File Sharing; für I/O-intensive Anwendungen wie Datenbanksysteme oder die Videoverarbeitung sind diese Protokolle aber nur bedingt geeignet. Deshalb beziehen Datenbanksysteme mit hohen Leistungs-Anforderungen ihren Speicher häufig direkt von Disk- und Flashsystemen anstatt von Fileservern.

Engpass: Systembus und PCI-Bus

Nehmen wir mal an, dass ein Benutzer auf einem NFS-Client eine Datei lesen möchte, die auf einem NAS-Server mit internen SAS-Platten gespeichert ist. Dann lädt das Betriebssystem des NAS-Systems die Datei zunächst von dem Laufwerk über den SAS-Bus, den PCI-Bus und den Systembus in den Hauptspeicher des NAS-Servers, nur um sie von dort wieder über den Systembus und den PCI-Bus an die Netzwerkkarte weiterzuleiten. Von dort werden die Daten dann an den NAS-Client geschickt. Die Daten werden also auf dem NAS-System zweimal durch den Systembus und den PCI-Bus des NAS-Servers »geschaufelt« (Abb. 4–11). Bei entsprechender Last auf einem Dateisystem können so die Busse des Fileservers zum Leistungsengpass werden.

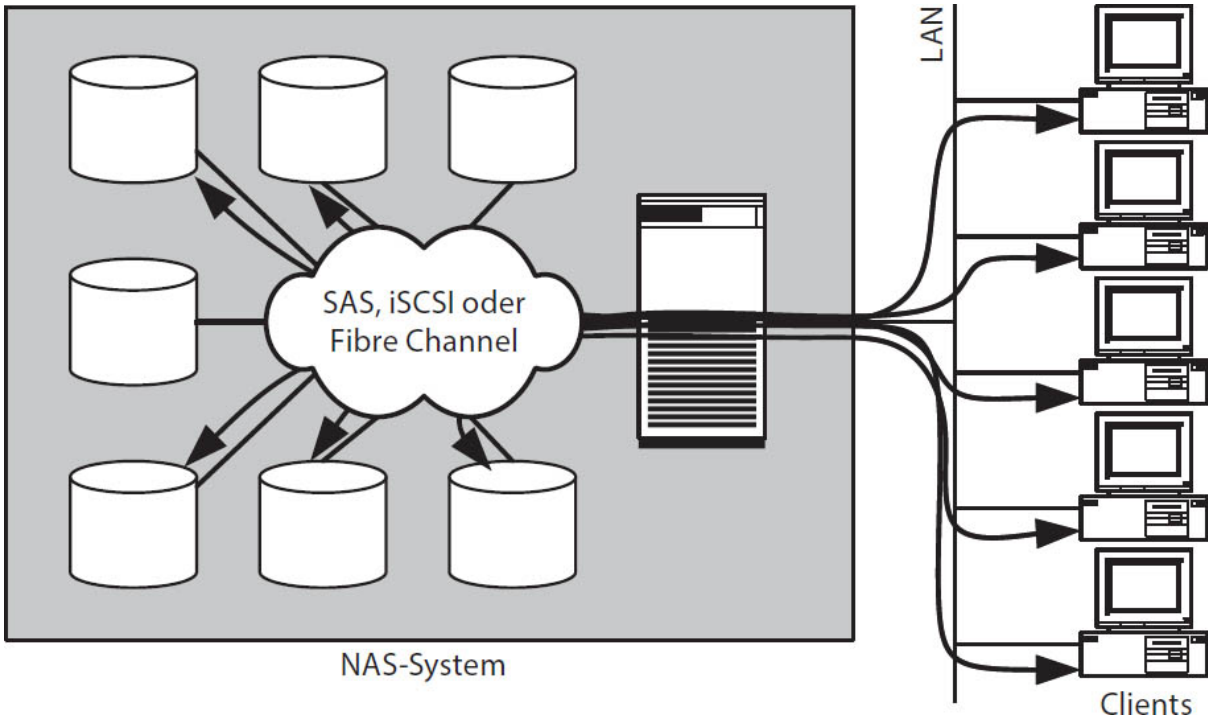

Abb. 4–11 Nadelöhr Fileserver

Auf dem Weg zwischen Laufwerk und Client passieren alle Daten zweimal die internen Busse des Fileservers.

Belastung von CPU und Systembus durch Kopieroperationen

Beim Einsatz klassischer Netzwerk-Dateisysteme werden auf dem sendenden Rechner die zu transportierenden Daten zusätzlich vom privaten Speicherbereich der Anwendung in den Buffer Cache des Kernels kopiert, bevor dieser die Daten über den PCI-Bus in den Packet Buffer der Netzwerkkarte kopiert. Durch jeden einzelnen Kopiervorgang erhöht sich somit die Latenz der Kommunikation, die Belastung der CPU durch teure Prozesswechsel zwischen Anwendungsprozessen und Kernelprozessen sowie die Belastung des Systembusses zwischen CPU und Hauptspeicher.

Hohe CPU-Last durch TCP/IP

Von der Netzwerkkarte aus wird die Datei dann über IP und Ethernet zum NAS-Client übertragen. Beim heutigen Stand der Technik können die meisten Ethernet-Karten nur einen kleinen Teil des TCP/IP-Protokolls eigenständig abwickeln, sodass die CPU den Rest des Protokolls selbst abwickeln muss. Die Kommunikation von der Ethernet-Karte zur CPU wird über Interrupts initiiert. Beides zusammen kann viel CPU-Zeit kosten (Abschnitt 3.5.1).

4.4.2 Beschleunigung von Netzwerk-Dateisystemen

Beschleunigung des File Sharings

Schaut man sich den I/O-Pfad von der Anwendung zu den an einem NAS-System angeschlossenen Laufwerken an, so bieten sich zwei Ansatzpunkte, das File Sharing zu beschleunigen: (1.) das zugrunde liegende Kommunikationsprotokoll (TCP/IP) und (2.) das Netzwerk-Dateisystem (NFS, SMB) selbst.

TCP/IP Offload Engine (TOE)

TCP/IP wurde ursprünglich für den zuverlässigen Datenaustausch über unzuverlässige Transportwege entwickelt. Der TCP/IP-Protokollturm ist dementsprechend komplex und CPU-intensiv. Eine erste Verbesserung bringen hier spezielle Hardwarekomponenten, sogenannte TCP/IP Offload Engines (TOEs). Im Gegensatz zu herkömmlichen Netzwerkkarten arbeiten diese einen großen Teil des TCP/IP-Protokollturms auf einem eigenen Prozessor ab und entlasten somit die Server-CPU erheblich (Abschnitt 3.5.1).

Einsatz von VI und RDMA

Noch besser wäre es, TCP/IP gegen optimierte Kommunikationstechniken auszutauschen. Hierfür kommen Virtual Interfaces (VI) und Remote Direct Memory Access (RDMA) ins Spiel (Abschnitte 3.6.2 und 3.6.3). Dabei handelt es sich um einen Kommunikationsmechanismus, mit dem vom Netzwerk aus, ohne Umweg über die CPU, direkt auf den Arbeitsspeicher eines Servers zugegriffen werden kann.

Socket Direct Protocol (SDP)

Ein Ansatz, Netzwerk-Dateisysteme mit VI und RDMA zu beschleunigen, war das Socket Direct Protocol (SDP), das die Vorteile von TOEs und RDMA-basiertem Datenaustausch kombinierte (Abschnitt 3.6.3). Damit konnten Protokolle, die wie NFS und SMB auf TCP/IP aufsetzen, über den Umweg SDP von einem RDMA-basierten Datenaustausch profitieren, ohne dass diese Protokolle angepasst werden mussten. Jedoch hat SDP nur wenig Akzeptanz gefunden und wird deshalb nicht mehr weiterentwickelt.

NFS über RDMA, SMB über RDM

Andere Ansätze bilden bestehende Netzwerk-Dateisysteme auf RDMA ab. So existieren Implementierungen von NFS auf RDMA für Linux- und Unix-Systeme. Microsoft hat mit Version 3 des SMB-Protokolls ebenfalls eine Implementierung über RDMA eingeführt (SMB Direct). Mittels RDMA schreibt der NAS-Client die Daten direkt in den Speicher des NAS-Servers. Der Vorteil dieses Ansatzes besteht darin, dass die über Jahre gereiften Netzwerk-Dateisysteme NFS beziehungsweise SMB lediglich einen neuen Kommunikationsmechanismus »untergeschoben« bekommen. Dies

ermöglicht es, den Entwicklungs- und Testzyklus zu verkürzen, sodass die qualitativen Anforderungen von Produktionsumgebungen vergleichsweise schnell erfüllt werden können.

Parallelisierung von Netzwerk-Dateisystemen

Ein weiterer Nachteil von traditionellen Netzwerk-Dateisystemen wie SMB und NFS besteht darin, dass jedes exportierte Netzwerk-Dateisystem von genau einem Fileserver bereitgestellt wird. Dadurch kommunizieren alle Clients mit demselben Fileserver, der somit die Anfragen aller Clients bearbeiten muss. Für große Lasten ergibt sich dadurch ein Engpass; die Leistung aller NAS-Clients hängt dann nämlich von einem zentralen Fileserver ab. Als Gegenmaßnahme lässt sich ein Netzwerk-Dateisystem in mehrere zerlegen und jeder Teil von einem anderen Fileserver bereitstellen. Allerdings muss sich dann ein Administrator um die Aufteilung kümmern und die Konfiguration aller NAS-Clients entsprechend anpassen. Dies ist sehr aufwendig, wenn sich im Laufe der Zeit das Lastprofil ändert.

Parallel NFS (pNFS)

Mit der im Jahr 2010 verabschiedeten Version 4.1 wurde NFS um das sogenannte Parallel NFS (pNFS) erweitert. pNFS trennt die eigentlichen Daten von den für den Zugriff von Clients benötigten Metadaten und speichert diese auf unterschiedlichen Servern. Bei pNFS verwaltet ein zentraler Metadaten-Server die Verzeichnisstruktur sowie die Speicherorte der im Dateisystem gespeicherten Dateien, die eigentlichen Daten werden von einem oder mehreren Daten-Servern bereitgestellt. Ein pNFS-Client richtet Anfragen zunächst an den Metadaten-Server, die eigentlichen Daten werden anschließend direkt mit den Daten-Servern ausgetauscht. Selbst einzelne Dateien können hierbei über mehrere Daten-Server verteilt werden. pNFS erlaubt somit eine Parallelisierung von Client-Zugriffen über beliebig viele Server-Ressourcen hinweg. Dadurch wird ein zentraler Flaschenhals vermieden, ohne dass ein Administrator manuell Clients auf mehrere Fileserver verteilen muss. Trotz dieser wesentlichen Vorteile wird pNFS bis heute (2018) nur wenig eingesetzt.

Neue Netzwerk-Dateisysteme

Einen stärkeren Schnitt stellen neu entwickelte Netzwerk-Dateisysteme dar, die von vornherein eine zuverlässige Netzverbindung voraussetzen, wie das Direct Access File System (DAFS, Abschnitt 4.4.3) sowie die Familie der sogenannten Shared-Disk-Dateisysteme (Abschnitt 4.4.4).

4.4.3 Fallstudie: Direct Access File System (DAFS)

Das Direct Access File System (DAFS)

Das Direct Access File System (DAFS) ist ein von Grund auf neu entwickeltes Netzwerk-Dateisystem, das auf den Einsatz von RDMA zugeschnitten ist. Es basiert auf NFS Version 4, setzt auf VI auf und kann dessen Möglichkeiten voll nutzen. DAFS sieht vor, dass mehrere DAFS-Server gemeinsam den Speicherplatz für ein großes Dateisystem bereitstellen (Abb. 4–12). Dem Anwendungsserver bleibt es als DAFS-Client verborgen, auf welchen DAFS-Servern die Daten tatsächlich liegen (Abb. 4–13 auf S. 206).

Kommunikationstechniken für DAFS

Die Kommunikation zwischen DAFS-Client und DAFS-Server erfolgt in der Regel über RDMA. Durch den Einsatz vom RDMA ist der Zugriff auf Daten, die auf einem DAFS-Server liegen, fast genauso schnell wie der Zugriff auf lokale Daten. Zusätzlich werden typische Dateisystem-Operationen wie die Adressumrechnung von Dateien auf SCSI-Blockadressen, die auch CPU verbrauchen, von dem Anwendungsserver auf den DAFS-Server verlagert.

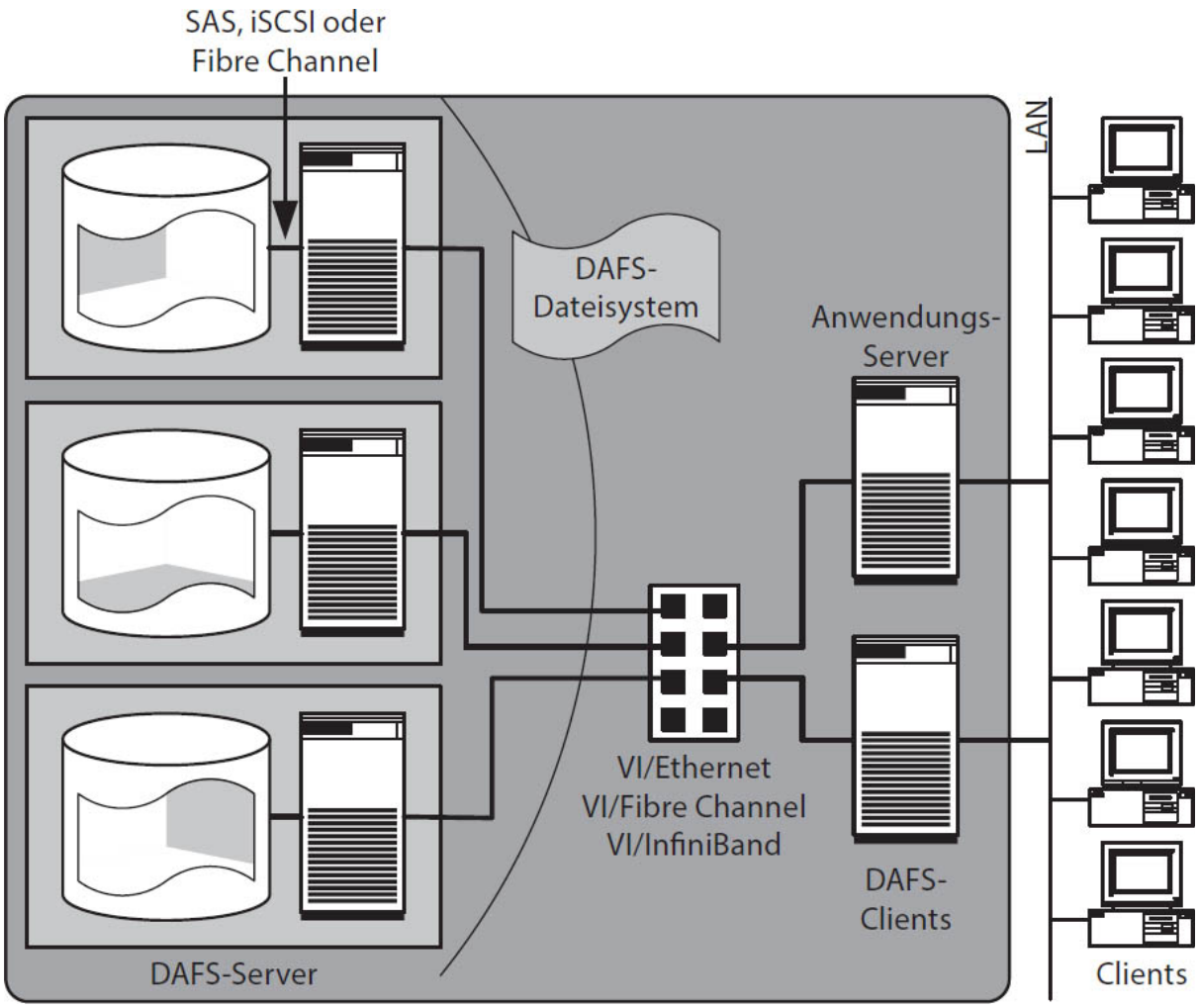

Abb. 4–12 DAFS Cluster

Ein DAFS-Dateisystem kann sich über mehrere DAFS-Server erstrecken. Alle DAFS-Server und DAFS-Clients sind über ein VI-fähiges Netz wie InfiniBand, Fibre Channel oder Ethernet miteinander verbunden.

Locking

Eine wichtige Funktion des File Sharings im Allgemeinen ist die Synchronisation zeitgleicher Zugriffe auf Dateieinträge, also Metadaten wie Dateinamen und Zugriffsrechte, sowie auf Dateiinhalte. Diese Synchronisation erfolgt über Locks, um die Konsistenz der Daten und der Metadaten zu wahren. DAFS erlaubt es, die Locks zu cachen, sodass ein weiterer Zugriff auf dieselben Daten keiner Interaktion mit dem Fileserver bedarf. Benötigt ein Server den Lock-Eintrag eines anderen Servers, so überträgt dieser den Eintrag ohne Verzögerung. DAFS benutzt ein Lease-basiertes Locking, um die dauerhafte Sperrung einer Datei durch Ausfall eines Clients zu vermeiden. Darüber hinaus besitzt es Recovery-Mechanismen für den Fall, dass die Verbindung zwischen DAFS-Client und DAFS-Server kurzfristig unterbrochen ist oder ein anderer Server aus dem Cluster einspringen muss. Ebenso übernimmt

DAFS die Authentisierung von Client und Server und kann darüber hinaus auch einzelne Benutzer gegenüber einer Client-Server-Session authentisieren.

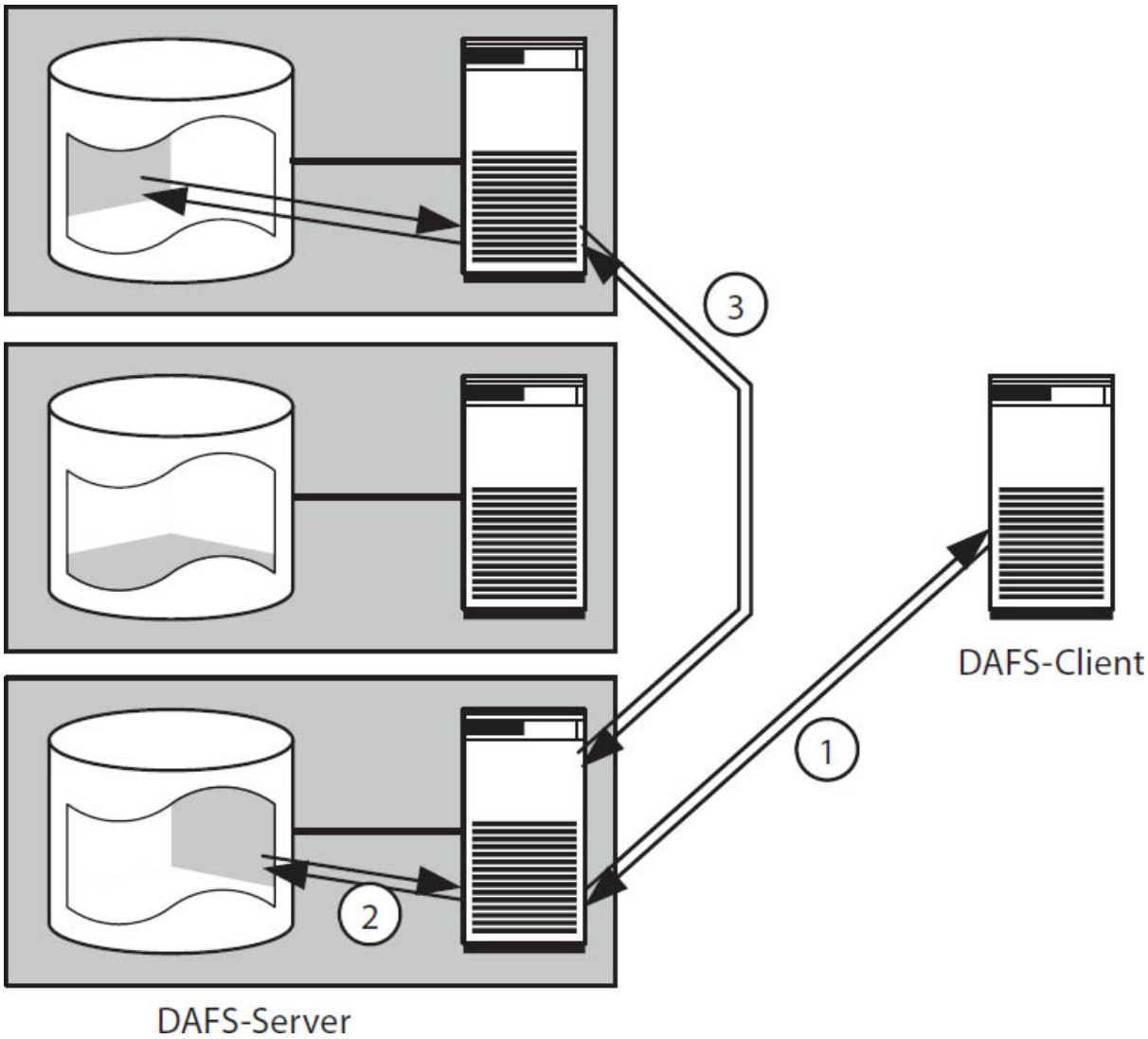

Abb. 4–13 DAFS-Datenfluss

Der DAFS-Client kommuniziert nur mit einem DAFS-Server (1). Dieser bearbeitet Dateizugriffe, deren Blöcke er selbst verwaltet (2). Der DAFS-Server leitet Speicherzugriffe auf Daten, die auf einem anderen DAFS-Server liegen, für den DAFS-Client verborgen an den entsprechenden DAFS-Server weiter (3).

DAFS und Datenbanksysteme

DAFS hat das Potenzial, dass DAFS-fähige NAS-Server I/O-intensive Anwendungen wie Datenbanksysteme, Batch-Prozesse oder Big-Data-Analysen unterstützen können. Dabei laufen die Anwendungen auf DAFS-Clients und sind mittels VI, RDMA und InfiniBand an die DAFS-Server angebunden. Das hat den Vorteil, dass die Adressumrechnung von Dateien in SCSI-Blockadressen von dem Anwendungsserver auf den DAFS-Server verlagert wird und somit die CPU des Anwendungsservers weniger belastet.

Standardisierung von DAFS

Die Entwicklung von DAFS wurde in der Vergangenheit vor allem von der Firma Network Appliance vorangetrieben, einem wichtigen NAS-Hersteller. Die Standardisierung des DAFS-Protokolls zur Kommunikation zwischen Server und Client und der DAFS-API zur Nutzung des Dateisystems durch Anwendungen erfolgte unter dem Dach des DAFS Collaborative. Seit der Verabschiedung der Version 1.0 im September 2001 (Protokoll) beziehungsweise im November 2001 (API) ist die Standardisierung von DAFS zum Stehen gekommen. Ursprünglich wurde DAFS im September 2001 bei der IETF als Internetstandard eingereicht, jedoch hat es in der Speicherindustrie nur wenig Zuspruch gefunden. Stattdessen richtete sich die breite Aufmerksamkeit schnell auf eine andere Alternative, nämlich die Erweiterung von NFS und SMB um RDMA als Transportschicht sowie das Hinzufügen von DAFS-ähnlichen Lock-Semantiken.

Fazit

DAFS ist ein interessanter Ansatz, um NAS-Systeme als Speicher für I/O-intensive Anwendungen einzusetzen. Mangels Standardisierung von DAFS und der fehlenden breiten Unterstützung durch die Speicherindustrie haben sich DAFS-Produkte am Markt jedoch nie nennenswert durchsetzen können. Inzwischen sind Alternativen wie NFS über RDMA oder SMB über RDMA verfügbar, die einige der Vorteile von DAFS wieder aufgreifen. Außerdem ist iSCSI und FCoE für diejenigen interessant, die mit DAFS eine Investition in Fibre-Channel-Netze vermeiden wollen, zumal iSCSI ebenso wie NFS und SMB von TCP/IP Offload Engines und einer direkten Abbildung von iSCSI auf RDMA (iSER) profitieren kann. Daneben bieten die im folgenden Abschnitt beschriebenen Shared-Disk-Dateisysteme ebenfalls die Möglichkeiten eines schnellen Filesharings und die in Kapitel 5 vorgestellte Speichervirtualisierung eine Alternative für die effiziente Bereitstellung von zentralen und sehr leistungsfähigen Speicherressourcen.

4.4.4 Shared-Disk-Dateisysteme

Problem: Engpass Fileserver

Das größte Leistungsproblem von NAS-Systemen und selbst konfigurierten Fileservern besteht darin, dass jede Datei zweimal durch die internen Busse des Fileservers muss, bevor die Daten auf dem Server ankommen, auf dem sie benötigt werden (Abb. 4–11 auf S. 202). Selbst DAFS und Alternativen wie NFS und SMB über RDMA kommen um dieses Nadelöhr nicht herum.

Lösung: direkter Zugriff über das Speichernetz

Speichernetze bieten die Möglichkeit, dass mehrere Server gleichzeitig auf ein Speichergerät zugreifen. Der I/O-Engpass im Fileserver lässt sich

umgehen, wenn alle Clients die Dateien direkt über das Speichernetz von dem Laufwerk holen (Abb. 4–14). Wenn das Speichernetz mit Fibre Channel oder InfiniBand realisiert ist, dann erfreuen sich die Clients einer geringen Latenz, hoher Bandbreite und geringer CPU-Belastung.

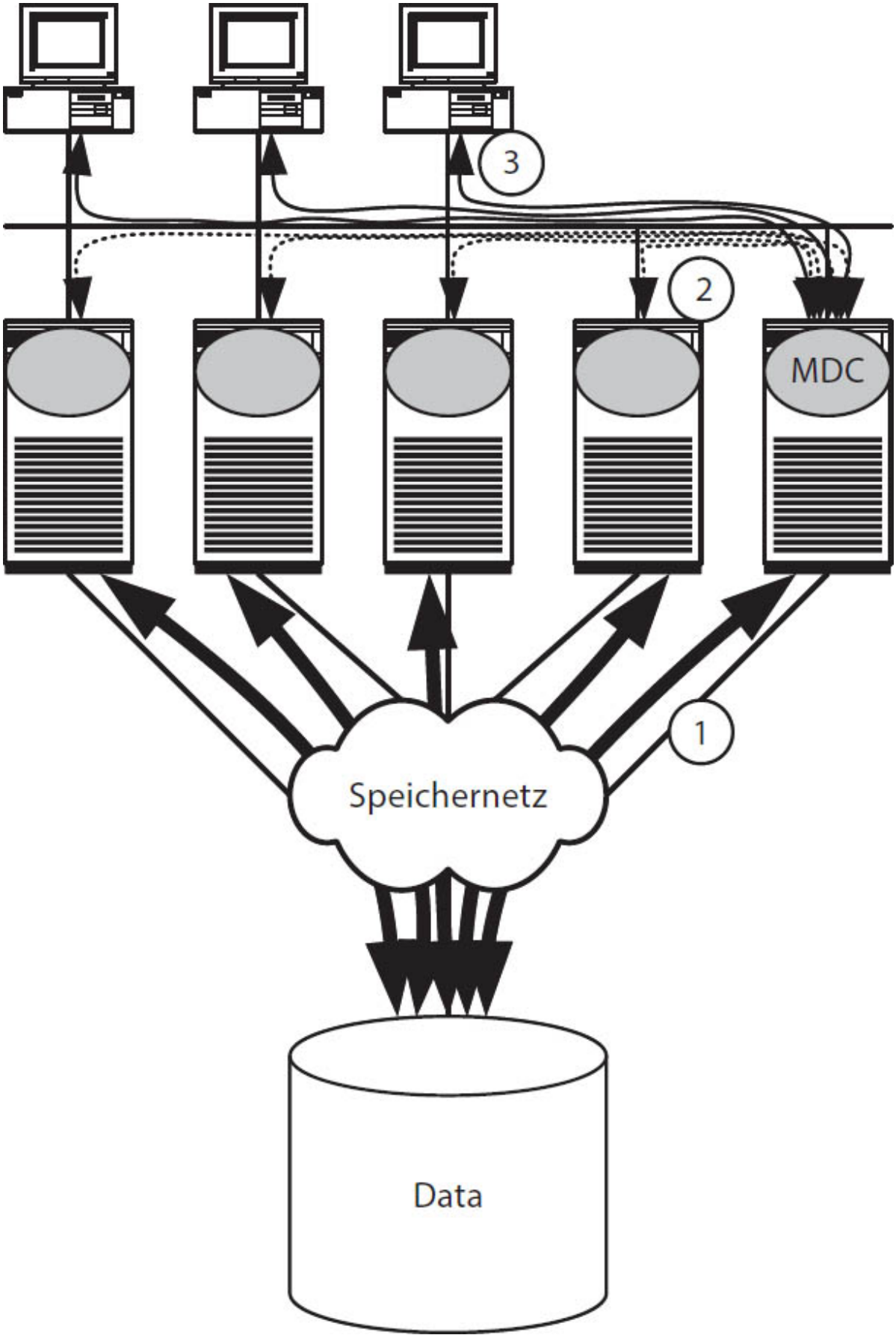

Abb. 4–14 Shared-Disk-Dateisysteme

Bei einem Shared-Disk-Dateisystem können alle Clients direkt über das Speichernetz auf die Laufwerke zugreifen (1). LAN-Datenverkehr ist nur noch zur Synchronisation der Schreibzugriffe notwendig (2). Die Daten des Shared-Disk-Dateisystems können zusätzlich als Netzwerk-Dateisystem mit NFS oder SMB über das LAN exportiert werden (3).

Herkömmliche Dateisysteme betrachten Laufwerke als lokal

Die Schwierigkeit dabei: Herkömmliche Dateisysteme betrachten ihre Speichergeräte als lokal, gehen also von einem exklusiven Zugriff darauf aus. Sie konzentrieren sich auf das Caching und die Zusammenfassung von I/O-Operationen und steigern so die Leistung, indem sie die Anzahl der notwendigen Speicheroperationen reduzieren. Würde man ein solches Dateisystem ohne weitere Maßnahmen gleichzeitig auf zwei Servern aktivieren, die per Fibre Channel oder SAS mit einem gemeinsamen Speichersystem verbunden sind, so käme es zu Inkonsistenzen der Verzeichnisstruktur und der Daten bis hin zu Datenverlusten.

Lösung: Shared-Disk-Dateisysteme

Sogenannte Shared-Disk-Dateisysteme können mit diesem Problem umgehen. Sie haben spezielle Algorithmen integriert, die zeitgleiche Zugriffe von mehreren Servern auf gemeinsamen Laufwerken synchronisieren. Dadurch ermöglichen Shared-Disk-Dateisysteme, dass mehrere Server gleichzeitig auf Dateien zugreifen, ohne Versionskonflikte hervorzurufen.

Synchronisation der Laufwerkszugriffe

Dazu müssen Shared-Disk-Dateisysteme zusätzlich zu den Funktionen herkömmlicher Dateisysteme die Schreibzugriffe synchronisieren. Dabei sollten sie dezentral sicherstellen, dass das Schreiben neuer Dateien auf unterschiedliche Bereiche des Laufwerks erfolgt. Außerdem müssen sie gewährleisten, dass Cache-Einträge gegebenenfalls als ungültig markiert werden, denn bei Shared-Disk-Dateisystemen wird häufig von mehreren Servern auf gemeinsamen Dateien gearbeitet. Hierbei kommt es vor, dass zwei Server dieselbe Datei jeweils in ihrem lokalen Cache puffern und einer der Server diese verändert. Liest später der zweite Server die Datei ein weiteres Mal, so darf er nicht die inzwischen ungültige Kopie aus seinem Cache benutzen. Mechanismen, die dieses Problem adressieren, basieren entweder auf der Verwaltung einer oder mehrerer zentraler Listen, in denen der Status aller gepufferten Daten gepflegt wird, oder aber auf einem zentralen Bus, über den alle teilnehmenden Rechner ihre Dateioperationen kommunizieren.

Direkter Zugriff auf die Laufwerke

Der große Vorteil von Shared-Disk-Dateisystemen besteht darin, dass der auf Dateien zugreifende Server und das betreffende Speichergerät jetzt direkt

miteinander kommunizieren, beispielsweise über ein Fibre Channel SAN, SAS oder InfiniBand. Der Umweg über einen zentralen Fileserver, der in herkömmlichen Netzwerk-Dateisystemen und auch bei DAFS oder NFS und SMB über RDMA den Engpass darstellt, ist nicht mehr erforderlich.

Leistung

Zusätzlich entlastet man auf dem zugreifenden Server die CPU, weil Kommunikation über Fibre Channel, SAS oder InfiniBand weniger Prozessorlast verursacht als Kommunikation über IP und Ethernet. Der sequenzielle Zugriff auf große Dateien kann so den Mehraufwand für Zugriffssynchronisation wettmachen. Bei Anwendungen mit vielen kleinen Dateien oder bei vielen Random-Zugriffen innerhalb derselben Datei sollte man dagegen testen, ob sich der Einsatz eines Shared-Disk-Dateisystems wirklich lohnt, denn hier ist mehr Zugriffssynchronisation notwendig.

Verfügbarkeit

Als Nebeneffekt des File Sharings über das Speichernetz kann die Verfügbarkeit des Shared-Disk-Dateisystems höher sein als die von herkömmlichen Netzwerk-Dateisystemen. Es wird nämlich kein zentraler Fileserver mehr benötigt. Fällt ein Server im Shared-Disk-Dateisystem-Cluster aus, so können die anderen Server weiterarbeiten. Das heißt, die Verfügbarkeit der zugrunde liegenden Speichergeräte bestimmt im Wesentlichen die Verfügbarkeit von Shared-Disk-Dateisystemen. Diese kann beispielsweise durch Datenspiegelung auf mehrere Speichersysteme noch weiter gesteigert werden.

4.4.5 Fallstudie: General Parallel File System (GFPS)

Das General Parallel File System (GPFS)

Wir haben uns entschlossen, mit dem IBM General Parallel File System (GPFS) ausnahmsweise ein Produkt unseres Arbeitgebers vorzustellen, weil es das Shared-Disk-Dateisystem ist, das wir am besten kennen, und die Vorstellung eines konkreten Filesystems das Verständnis von Shared-Disk-Dateisystemen vertieft. GPFS wird seit vielen Jahren auf Cluster-Rechnern eingesetzt und heute (2018) unter dem Namen IBM Spectrum Scale vermarktet. GPFS sorgt für die Konsistenz des Dateisystems, also dafür, dass die Metadatenstruktur des Dateisystems aufrechterhalten bleibt. Beispielsweise werden keine Dateinamen zweimal vergeben. Weiter realisiert GPFS einige RAID-Funktionen wie Striping und Mirroring von Daten und Metadaten.

Virtual Shared Disks (VSD)
Network Shared Disks (NSD)

In früheren Jahren, als es noch keine Speichernetze gab, setzte GPFS auf sogenannte

Virtual Shared Disks (VSDs) auf (Abb. 4–15). Das VSD-Subsystem macht Laufwerke, die physisch an einem Server angeschlossen sind, auf anderen Knoten der SP, einem ehemaligen Cluster-Computer, sichtbar. Mehrere Knoten können somit auf dasselbe physische Laufwerk zugreifen. Das VSD-Subsystem sorgte dabei für die Konsistenz auf Blockebene, was bedeutet, dass ein Block entweder ganz geschrieben wird oder gar nicht. Aus heutiger Sicht könnte man sagen, dass VSDs die Funktion eines Speichernetzes emulieren. Im Laufe der Jahre wurde diese Schicht erneuert. VSDs wurden durch Network Shared Disks (NSDs) ersetzt, die aber immer noch die gleiche Aufgabe erfüllen wie VSDs.

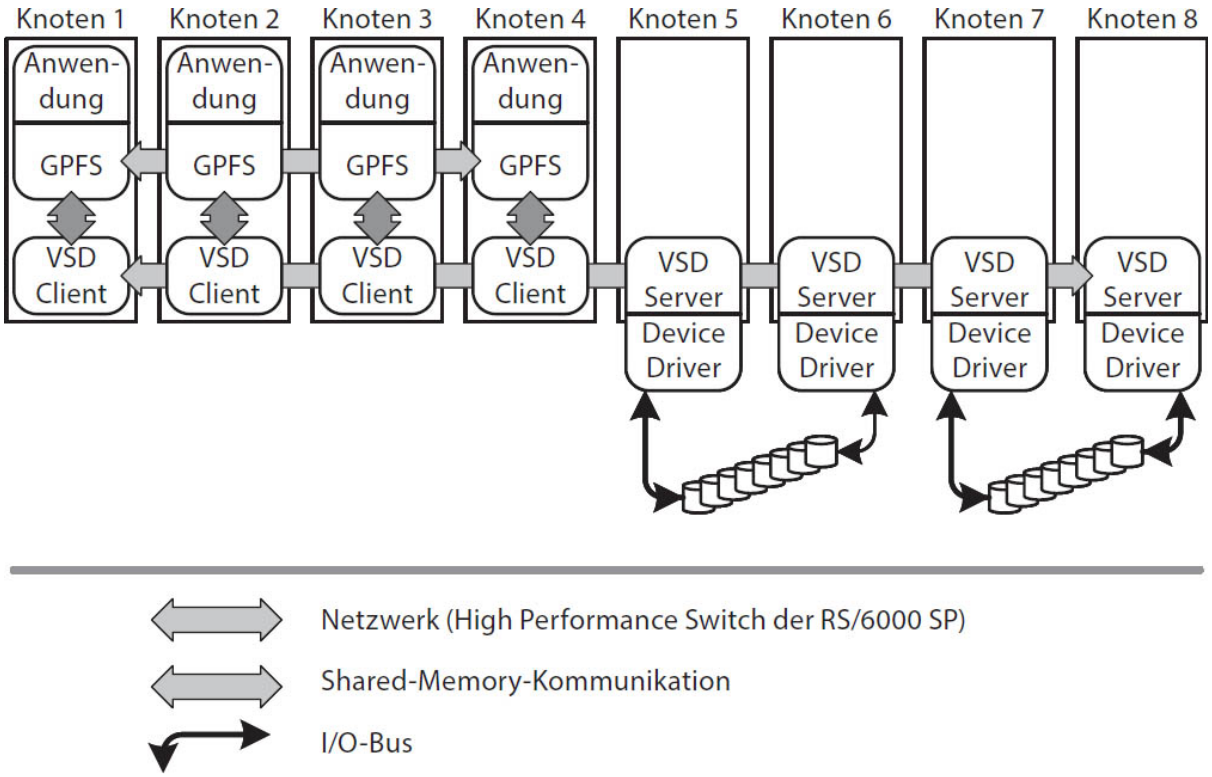

Abb. 4–15 GPFS-Architektur

Anwendungen sehen das GPFS-Dateisystem wie ein lokales Dateisystem. Das GPFS-Dateisystem selbst ist eine verteilte Anwendung, die parallele Zugriffe synchronisiert. Ursprünglich ermöglichte das VSD-Subsystem den Zugriff auf Laufwerke unabhängig davon, wo sie physisch angeschlossen sind.

Vorteile: 1. Striping

Abbildung 4–15 veranschaulicht zwei Vorteile von Shared-Disk-Dateisystemen: Zum einem können sie mittels RAID 0 die Daten über mehrere Laufwerke, mehrere Hostbus-Adapter und sogar über mehrere Flash- oder Disksysteme hinweg stripen, sodass Shared-Disk-Dateisysteme einen sehr hohen Durchsatz erreichen können. Hiervon profitieren alle Anwendungen, die zumindest teilweise ein sequenzielles Zugriffsmuster aufweisen.

2. Entkoppelung von Anwendung und Daten

Zum Zweiten wird der Ort der Anwendungen unabhängig von dem Ort der Daten: Der Systemadministrator kann, wie in Abbildung 4–15 gezeigt, Anwendungen auf den vier GPFS-Knoten starten, auf denen gerade die meisten Ressourcen (CPU, Hauptspeicher, Busse) zur Verfügung stehen. Ein sogenannter Workload Manager kann Anwendungen je nach Last automatisch von dem einen auf den anderen Knoten verlagern. Bei herkömmlichen Dateisystemen ist dies nicht möglich. Dort müssen die Anwendungen auf denjenigen Knoten laufen, auf denen das Dateisystem lokal verfügbar ist, da der Zugriff über ein Netzwerk-Dateisystem wie NFS oder SMB für sehr hohe Lastanforderungen zu langsam ist.

GPFS-Cluster: kein dedizierter Fileserver

Das Besondere an GPFS besteht darin, dass es keinen einzelnen Fileserver gibt. Jeder Knoten im GPFS-Cluster kann ein GPFS-Dateisystem mounten. Für Anwender und Anwendungen verhält sich das GPFS-Dateisystem bis auf die deutlich bessere Leistung wie ein herkömmliches POSIX-kompatibles Dateisystem.

GPFS-Daemon

Auf jedem Knoten im GPFS-Cluster muss der GPFS-Daemon laufen. GPFS ist als verteilte Anwendung realisiert, wobei alle Knoten in einem GPFS-Cluster die gleichen Rechte und Pflichten haben. Dazu muss der GPFS-Daemon über die klassischen Aufgaben eines Dateisystems hinaus je nach Konfiguration des GPFS-Clusters weitere Verwaltungsaufgaben übernehmen. In der Terminologie von GPFS kann der GPFS-Daemon eine oder mehrere der im Folgenden vorgestellten Rollen annehmen.

Cluster Manager

In jedem Cluster nimmt jeweils ein GPFS-Daemon die Rolle des Cluster Managers an. Der Cluster Manager bestimmt für jedes Dateisystem den File System Manager und überwacht das sogenannte Quorum. Das Quorum ist ein gängiges Verfahren in verteilten Systemen, das bei einem Netzwerksplit die Konsistenz der verteilten Anwendung – in diesem Fall des GPFS-Clusters – aufrechterhält. Für GPFS müssen mehr als die Hälfte der Quorum-Knoten eines Clusters aktiv sein. Geht das Quorum in einem Cluster verloren, so werden auf allen Knoten automatisch alle GPFS-Dateisysteme deaktiviert (unmount).

File System Manager

Ein GPFS-Cluster kann mit mehreren GPFS-Dateisystemen konfiguriert sein. Für jedes Dateisystem übernimmt ein GPFS-Daemon die Aufgabe des File System Managers. Der File System Manager ist für folgende Aufgaben zuständig:

- Konfigurationsänderungen des Dateisystems

- Verwaltung der Laufwerksblöcke
- Token-Verwaltung
- Verwaltung und Überwachung der Quotas
- Sicherheitsdienste

Token-Verwaltung zur Cache-Synchronisation

Besonders hervorzuheben ist die Token-Verwaltung: Eines der Designziele von GPFS ist die Unterstützung von parallelen Anwendungen, die von verschiedenen Knoten aus gemeinsame Dateien lesen und verändern. GPFS puffert wie jedes Dateisystem Dateien oder Dateifragmente im Hauptspeicher, um die Leistung zu steigern. GPFS verwendet einen Token-Mechanismus, um bei parallelen Schreib- und Lesezugriffen die Cache-Einträge auf verschiedenen Servern zu synchronisieren (Abb. 4–16). Diese Synchronisation stellt aber lediglich sicher, dass sich GPFS genauso verhält wie ein lokales Dateisystem, das nur auf einem Server gemountet werden kann. Das heißt, wie bei jedem Dateisystem müssen parallele Anwendungen auch bei GPFS die Zugriffe auf gemeinsame Dateien – etwa durch Locks – synchronisieren.

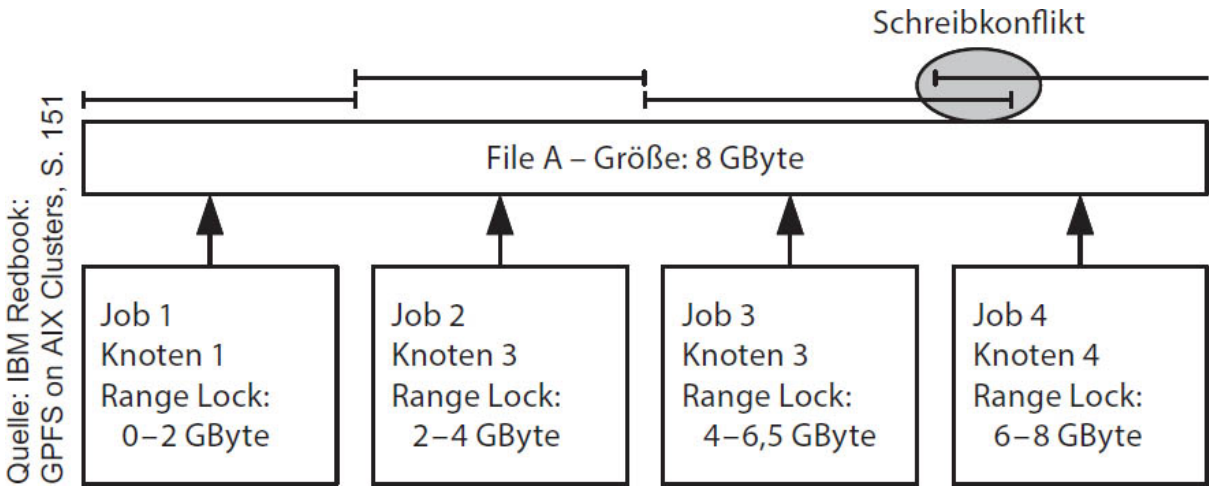

Abb. 4–16 GPFS-Token-Verwaltung zur Cache-Synchronisation

GPFS synchronisiert Schreibzugriffe für Dateibereiche. Fordern mehrere Knoten das Token für denselben Bereich an, so weiß GPFS, dass es Cache-Einträge synchronisieren muss.

Metadata Manager

Schließlich übernimmt für jede offene Datei ein GPFS-Daemon die Rolle des Metadata Managers. GPFS gewährleistet die Konsistenz der Metadaten einer Datei, indem nur der Metadata Manager die Metadaten verändern darf. In der Regel ist für eine Datei der Knoten der Metadata Manager, auf dem sie als Erstes geöffnet wurde. Die Zuordnung des Metadata Managers einer Datei zu einem Knoten kann sich in Abhängigkeit vom Zugriffsverhalten der Anwendungen ändern.

Fazit

Das Beispiel GPFS zeigt, dass ein Shared-Disk-Dateisystem eine ganze Menge mehr leisten muss als ein herkömmliches lokales Dateisystem, das nur auf einem Rechner verwaltet wird. GPFS wird wie andere Shared-Disk-Dateisysteme seit vielen Jahren erfolgreich in sehr großen Cluster-Rechnern eingesetzt und kontinuierlich um neue Funktionen ergänzt.

4.4.6 Shared-Nothing-Dateisysteme

Shared-Nothing-Dateisysteme

Shared-Nothing-Dateisysteme sind ein alternativer Ansatz zu den Shared-Disk-Dateisystemen. Bei Shared-Nothing-Dateisystemen erbringt ein Cluster von Servern die Speicherkapazität, wobei die Daten ausschließlich auf internen Laufwerken der Server gespeichert werden (Abb. 4–17). Der Cluster bildet somit einen Shared-Nothing Cluster (Abschnitt 8.4.3). Verfügbarkeit wird dadurch erreicht, dass mehrere Server eine Kopie der Dateien halten.

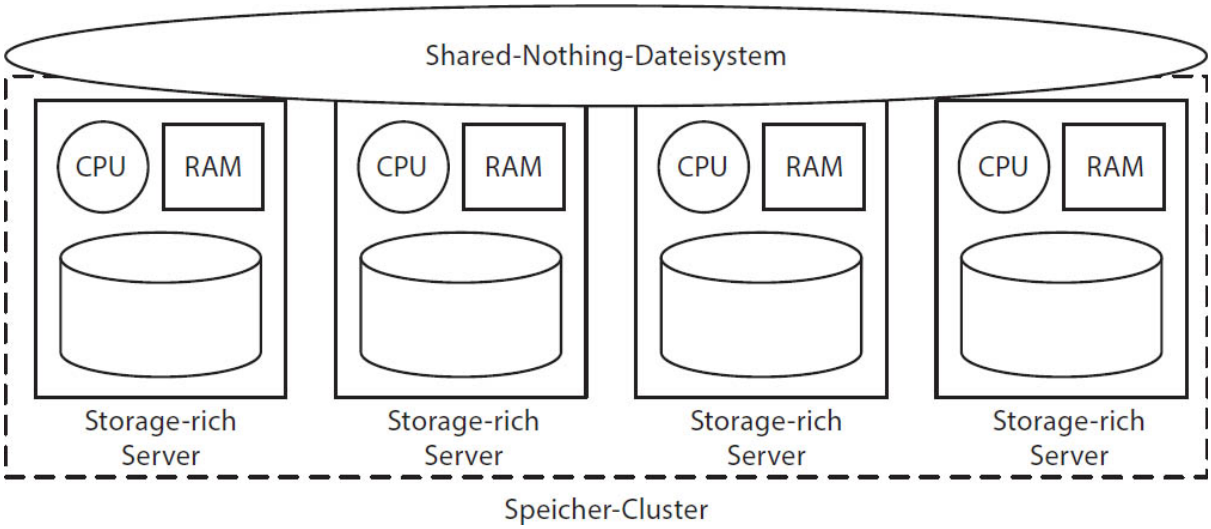

Abb. 4–17 Shared-Nothing-Dateisysteme
Ein Shared-Nothing-Dateisystem verzichtet auf den Einsatz zentraler Speicherkomponenten. Stattdessen werden die Daten ausschließlich auf internen Laufwerken der Server gespeichert.

Einordnung

Shared-Nothing-Dateisysteme haben viele Gemeinsamkeiten mit Objektspeicher, den wir in Kapitel 6 ausführlich beschreiben. Sie eignen sich gut für Anwendungen, bei denen die Lesezugriffe überwiegen. Die einzelnen Knoten können dann weitestgehend autark arbeiten, sodass eine Synchronisation mit anderen Knoten nicht notwendig ist. Bei Schreibzugriffen muss die neue Version der Daten

dagegen auf allen Knoten synchronisiert werden, die jeweils eine Kopie der Daten halten. Dies ist zeitaufwendig und beansprucht die Netze zwischen den Knoten.

Einsatzbeispiele

Shared-Nothing-Dateisysteme eignen sich gut für die Analyse unstrukturierter Daten, da hierbei überwiegend Daten gelesen werden. In nächsten Abschnitt stellen wir das Hadoop Distributed File System (HDFS) näher vor, das heute (2018) am weitesten verbreitete Shared-Nothing-Dateisystem.

4.4.7 Fallstudie: Hadoop Distributed File System (HDFS)

Das Apache Hadoop Framewor

Apache Hadoop ist ein Software-Framework, mit dessen Hilfe umfangreiche Big-Data-Anwendungen realisiert werden können. Das Open-Source-Projekt basiert dabei auf Konzepten, die ursprünglich von Google formuliert und veröffentlicht wurden. So war eine der ersten Anwendungen für das von Hadoop vorgegebene Programmiermodell das effiziente Durchsuchen und Indizieren von Webseiten. Hadoop und dessen Nachfolger Spark sind aber keinesfalls auf diesen Anwendungsfall beschränkt. Beide genießen inzwischen große Beliebtheit in vielen Bereichen, in denen sehr große Mengen unstrukturierter Daten analysiert werden müssen.

Fehlerhafte Hardware als Normalfall

Die Architektur von Apache Hadoop basiert auf Clustern, die aus vielen Rechnern bestehen, sogenannten Cluster-Knoten. Ein Cluster-Knoten verfügt über CPU- und Speicherressourcen und führt eine entsprechende Software-Komponente aus. Hadoop stellt dabei keine speziellen Anforderungen an die Verfügbarkeit der verwendeten Server; auf zentrale Speichersysteme wird grundsätzlich verzichtet. Stattdessen wird eine Philosophie vertreten, wonach Anwendungen Hardware-Fehler und Ausfälle tolerieren müssen, da diese in großen Clustern unumgänglich sind. Hadoop stellt zu diesem Zweck Software-Bibliotheken zur Verfügung, mit deren Hilfe solche fehlertoleranten Anwendungen realisiert werden können. Die beiden Kernkomponenten unterstützen Anwendungsentwickler bei der Datenverarbeitung (Map Reduce) und bei der Datenspeicherung (Hadoop Distributed File System, HDFS).

Das Map-Reduce-Programmiermodell

Mit dem sogenannten Map-Reduce-Programmiermodell lassen sich Algorithmen entwickeln, die auf sehr vielen Cluster-Knoten parallel ausgeführt werden können. Dabei wird zwischen einer »Map«-Phase und einer »Reduce«-Phase unterschieden. Während der Map-Phase verarbeiten die Software-Komponenten, die auf den Cluster-Knoten laufen, jeweils lokale Daten

und produzieren ihre Zwischenergebnisse. Die Zwischenergebnisse aller Cluster-Knoten werden anschließend in der Reduce-Phase zusammengeführt. Meist findet während der Map-Phase eine Filterung und Sortierung ausgewählter Datensätze statt – erst in der Reduce-Phase findet dann eine Verknüpfung und Zusammenfassung der Ergebnisse mehrerer Datensätze statt. Durch diese strikte Trennung können Map-Reduce-Algorithmen auf sehr vielen Knoten gleichzeitig und daher mit sehr großen Server-Clustern parallel ausgeführt werden.

Lokalität von Daten

Ein wichtiges Grundprinzip von Hadoop besteht darin, dass diejenigen Cluster-Knoten die jeweiligen Berechnungen durchführen, die möglichst direkten Zugriff auf die Daten haben. Ein Transfer der Daten über das Netzwerk wird, so weit wie möglich, vermieden. Dazu sind genaue Informationen über den Speicherort der Daten erforderlich. Aus diesem Grund erlaubt es Hadoop, für jeden Knoten zu definieren, an welchen Netzwerk-Switch dieser angeschlossen ist. Berechnungen werden dann entweder auf dem Knoten durchgeführt, auf dem die Daten bereits gespeichert sind, oder aber auf einem Knoten am selben Switch, um den Netzwerkverkehr zu minimieren. Darum eignen sich Shared-Nothing Cluster (Abschnitt 8.4.3) sehr gut für das Map-Reduce-Programmiermodell, da jeder Knoten über eigenen, lokalen Speicher verfügt.

Das Hadoop Distributed File System (HDFS)

Vor diesem Hintergrund stellt das Hadoop Distributed File System (HDFS) ein hochverfügbares Dateisystem zur Speicherung sehr großer Datenmengen bereit. Dateien werden darin zunächst in Datenblöcke fester Länge zerlegt. Diese werden redundant auf den internen Laufwerken mehrerer Cluster-Knoten gespeichert. Standardmäßig werden von jedem Datenblock drei Kopien abgelegt: eine auf einem beliebigen Knoten, eine weitere auf einem anderen Knoten am selben Switch und eine dritte Kopie auf einem Knoten an einem anderen Netzwerk-Switch. Durch dieses Verfahren wird einerseits der nötige Netzwerkverkehr minimiert, andererseits bleibt ein Zugriff auf den Datenblock auch dann noch möglich, wenn einzelne Netzwerkkomponenten ausfallen. Da die Verfügbarkeit der Daten im Cluster gewährleistet wird, kann auf traditionelle RAID-Verfahren innerhalb der Cluster-Knoten verzichtet werden.

Name Nodes und Data Nodes

Generell unterscheidet das HDFS-Dateisystem zwischen sogenannten Name Nodes und Data Nodes.
Ein Name Node verwaltet den globalen Namensraum und die nötigen Metadaten, die eigentlichen Datenblöcke werden jedoch auf mehreren Data Nodes gespeichert. Der Name Node verteilt die Datenblöcke auf möglichst viele Data Nodes und legt die nötigen Kopien gemäß der Topologie auf anderen Knoten wie

oben beschrieben im Cluster ab. Das HDFS-Dateisystem auf den Data Nodes verwendet für die eigentliche Speicherung der Daten wiederum lokale Dateisysteme des jeweiligen Betriebssystems, wie etwa das Linux-ext4-Dateisystem. Die Kommunikation zwischen den Knoten findet über ein eigenes, TCP/IP-basiertes Protokoll statt.

Verzicht auf POSIX-Schnittstelle

HDFS selbst ist in der Programmiersprache Java geschrieben und bietet native Schnittstellen in ebendieser Sprache. Über ein virtuelles FileSystem-Objekt haben (Java-)Anwendungen Zugriff auf die Daten des Clusters, ohne sich Gedanken über deren Speicherort machen zu müssen. Der Name Node stellt einen globalen Namensraum bereit und bedient Anwendungen, die auf Daten des Clusters zugreifen, vom jeweils nächstgelegenen Data Node aus, mit dem Ziel, den nötigen Netzwerkverkehr zu minimieren. Es wird jedoch auf eine traditionelle POSIX-Schnittstelle, wie sie beispielsweise von lokalen Dateisystemen bekannt ist, verzichtet. Dank der Popularität von Hadoop sind inzwischen unzählige Erweiterungen und Schnittstellen verfügbar, die es erlauben, HDFS-Dateisysteme in jeglichen Anwendungen und Prozessen zu integrieren.

Fazit

HDFS wird erfolgreich in Hadoop-Implementierungen eingesetzt. Eine Schwachstelle ist jedoch die fehlende POSIX-Schnittstelle. Während diese für reine Big-Data-&-Analytics-Projekte nicht benötigt wird, erschwert diese Lücke die Integration in bestehende IT-Strukturen und Anwendungen von Unternehmen, sodass HDFS-Dateisysteme heute (2018) isoliert neben anderen bereits vorhandenen Dateisystemen stehen, die gesondert konfiguriert, verwaltet und überwacht werden müssen. Deshalb ist es wünschenswert, bestehende Dateisysteme um eine HDFS-konforme Schnittstelle zu erweitern beziehungsweise das HDFS-Dateisystem mit anderen Dateisystemschnittstellen wie POSIX, NFS oder SMB auszustatten.

4.5 Vergleich: NAS und SAN

SAN und NAS

Blockbasierte Speichernetze (Storage Area Networks, SANs) und Dateibasierte Speichernetze (Network Attached Storage, NAS) sind zwei alternative Techniken, mit denen Speichernetze realisiert werden können. Abbildung 4–18 stellt die I/O-Pfade der beiden Techniken gegenüber und Tabelle 4–19 fasst die wichtigsten Unterschiede zusammen.

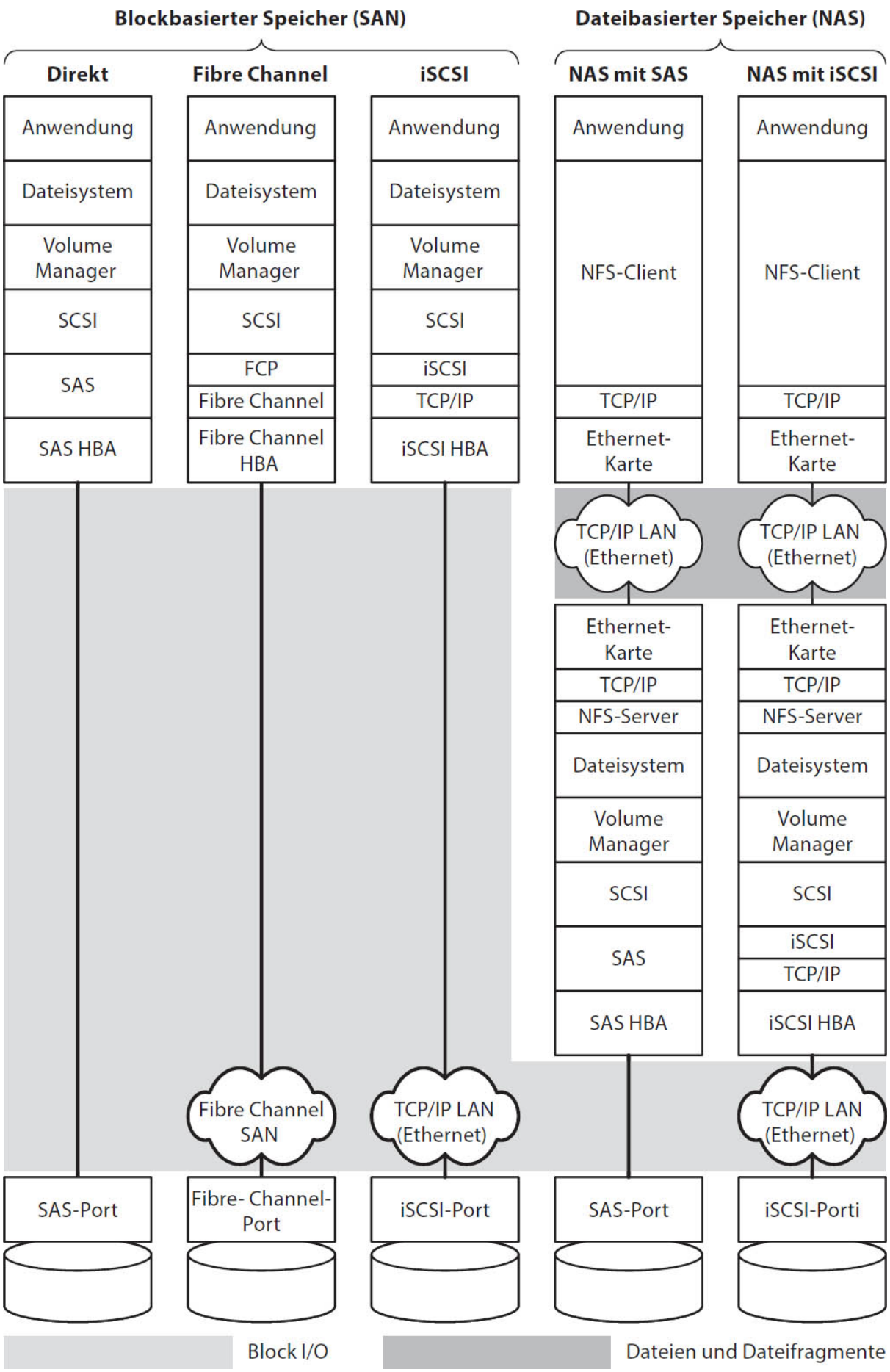
Blockbasierter Speicher (SAN)
Dateibasierter Speicher (NAS)
Direkt
Fibre Channel
iSCSI
NAS mit SAS
NAS mit iSCSI
Anwendung
Dateisystem
Volume Manager
SCSI
SAS
SAS HBA
FCP
Fibre Channel
Fibre Channel HBA
iSCSI
TCP/IP
iSCSI HBA
NFS-Client
TCP/IP
Ethernet-Karte
TCP/IP LAN (Ethernet)
NFS-Server
Dateisystem
Volume Manager
SCSI
SAS
SAS HBA
iSCSI
TCP/IP
iSCSI HBA
Fibre Channel SAN
TCP/IP LAN (Ethernet)
SAS-Port
Fibre- Channel-Port
iSCSI-Port
SAS-Port
iSCSI-Porti
Block I/O
Dateien und Dateifragmente

Abb. 4–18 Vergleich der I/O-Pfade: SCSI, Fibre Channel, iSCSI, NAS
Die Abbildung stellt noch einmal die unterschiedlichen I/O-Pfade von SAS, iSCSI, Fibre Channel und NAS gegenüber.

	Blockbasierter Speicher (SAN)	Dateibasierter Speicher (NAS)
Protokoll	FCP, iSCSI, FCoE (SCSI)	NFS, SMB, HTTPS
Netz	Fibre Channel, TCP/IP, Ethernet	TCP/IP
Quelle/Ziel	Server/Speichergerät	Client/NAS-Server, Anwendungsserver/NAS-Server
Transferobjekte	Geräteblöcke	Dateien, Dateifragmente
Zugriff über das Speichergerät	direkt über Fibre Channel, iSCSI oder FCoE	indirekt über den NAS-internen Rechner
Eingebettetes Dateisystem	nein	ja
Prefetch Hit Rate	50%	nahe 100%
Konfiguration	durch Endanwender (flexibel)	vorkonfiguriert durch NAS-Hersteller (Plug&Play)
Eignung für Datenbanksysteme	ja	bedingt
Produktionsreife	ja	ja

Tab. 4–19 Vergleich: blockbasierter und dateibasierter Speicher

SAN: blockbasiert

Bei SAN findet im Gegensatz zu NAS der Datenaustausch zwischen Servern und Speichergeräten blockbasiert statt. Diese Speichernetze sind schwieriger zu konfigurieren. Dafür liefert zumindest Fibre Channel optimale Leistung für den Datenaustausch zwischen Server und Speichergerät.

NAS: dateibasiert

NAS-Systeme dagegen sind schlüsselfertige Fileserver und man kann sie ausschließlich als Fileserver einsetzen. Als Datenspeicher für I/O-intensive Datenbanksysteme sind sie mangels Leistung nur bedingt geeignet. Mit NAS-Systemen lassen sich Speichernetze realisieren, indem man zwischen NAS-System und den Anwendungsservern ein zusätzliches LAN installiert (Abb. 4–20). Im Gegensatz zu SAN überträgt dieses Speichernetz Dateien oder Dateifragmente.

Prefetch Hit Rate für NAS

Ein Vorteil von NAS besteht darin, dass NAS-Systeme eine höhere Prefetch Hit Rate für Lesezugriffe erzielen können als Disk- und Flashsysteme, die über Fibre Channel oder iSCSI (oder nur über SAS) angeschlossen sind. NAS-Systeme arbeiten auf Dateisystemebene, Flash- und Disksysteme nur auf Blockebene. Ein NAS-System weiß, welche Blöcke zu einer Datei gehören, und kann diese von den Laufwerken in den Hauptspeicher einlagern und so nachfolgende Dateizugriffe schneller aus dem Hauptspeicher bedienen. NAS-Systeme und Fileserver können dadurch eine Prefetch Hit Rate von nahe 100% erreichen.

Prefetch Hit Rate für SAN

Ein blockbasiertes Speichersystem weiß hingegen nicht, wo die Blöcke, die zu einer Datei gehören, liegen, zumal Dateien auf einem blockbasierten Speicher nicht notwendigerweise benachbarte Blöcke belegen. Es kann nur versuchen, zu erraten, welcher Block der nächste ist. Flash- und Disksysteme kommen dadurch lediglich auf eine Prefetch Hit Rate von circa 50%, weil sie nur Blöcke kennen. Sie wissen nicht, wie die Daten (beispielsweise ein Dateisystem oder ein Datenbanksystem) in den Blöcken organisiert sind. Darüber liegende Anwendungen haben in der Regel ihre eigenen Caching-Strategien. So erreicht beispielsweise ein Fileserver oder ein NAS-System mit internen Laufwerken die gleiche Prefetch Hit Rate wie ein Fileserver beziehungsweise ein NAS-System, das Laufwerke von einem Disk- oder Flashsystem bezieht.

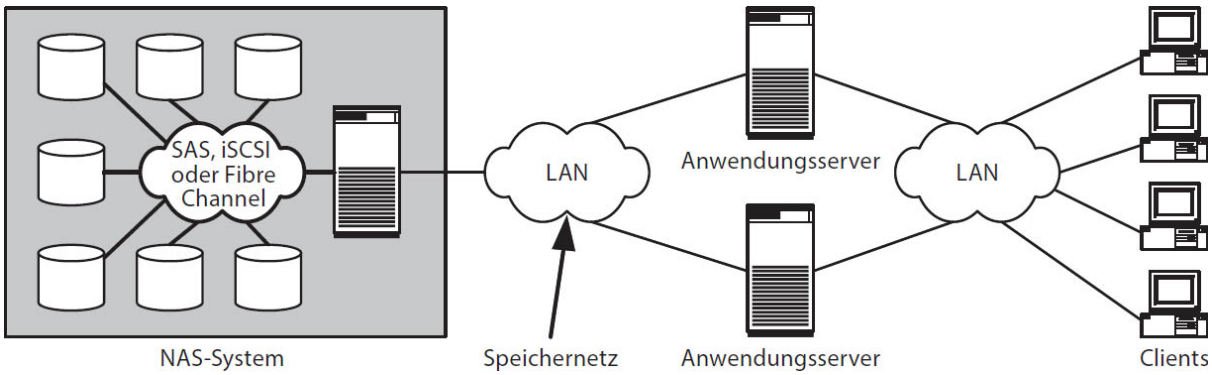

Abb. 4–20 Speichernetz mit NAS-Systemen
Aus Leistungsgründen ist hier zwischen Anwendungsservern und dem NAS-System ein separates LAN installiert, das als Speichernetz dient.

Einordnung

SAN und NAS werden seit vielen Jahren erfolgreich in Produktionsumgebungen eingesetzt. Fibre Channel eignet sich am besten für hohe Leistungsanforderungen I/O-intensiver Anwendungen. Insgesamt ist Stand 2018 Fibre Channel das vorherrschende Protokoll für blockbasierte Speichernetze. iSCSI und FCoE werden eher in Nischen eingesetzt, dies aber durchaus erfolgreich. NAS eignet sich hervorragend für

Webserver und für das File Sharing von Arbeitsgruppen. Mit NFS und SMB über RDMA könnte sich NAS auch als bequemer Datenspeicher für Datenbanksysteme mit hohen Leistungsanforderungen etablieren.

4.6 Zusammenfassung und Ausblick

Zusammenfassung des Kapitels

Laufwerke stellen ihren Speicher als Blöcke zur Verfügung, die über Zylinder, Sektoren und Spuren adressiert werden. Dateisysteme verwalten die Blöcke der Laufwerke und stellen Benutzern deren Speicherkapazität in Form von Verzeichnissen und Dateien zur Verfügung. Moderne Dateisysteme haben über Zusatzfunktionen, die in verschiedenen Situationen die Verfügbarkeit von Daten erhöhen: Journaling sorgt dafür, dass Dateisysteme nach einem Systemabsturz schnell wieder zur Verfügung stehen. Mit Snapshots können Datenbestände in wenigen Sekunden eingefroren und wiederhergestellt werden; der Volume Manager ermöglicht es, auf veränderte Speicheranforderungen ohne Unterbrechung des Betriebs zu reagieren. Daten-Management-Funktionen erlauben es, Kosten durch den sinnvollen Einsatz geeigneter Medientypen zu senken. Netzwerk-Dateisysteme, Shared-Disk-Dateisysteme und Shared-Nothing-Dateisysteme ermöglichen von verschiedenen Servern aus den Zugriff auf einen gemeinsamen Datenbestand. Dazu exportieren Netzwerk-Dateisysteme lokale Dateisysteme über das LAN. Immer wenn mehrere Benutzer oder Anwendungen auf Daten zugreifen wollen, müssen Zugriffe authentisiert und autorisiert werden, um Daten vor unerlaubten Zugriffen zu schützen. Die Authentisierung erfolgt entweder lokal oder über einen zentralen Verzeichnisbeziehungsweise Authentisierungsdienst. Netzwerk-Dateisysteme werden von Fileservern oder NAS-Systemen bereitgestellt. NAS-Systeme sind vorkonfigurierte Fileserver, deren Betriebssystem für die Aufgaben von Fileservern optimiert wurde. Auch mit NAS-Systemen können Speichernetze realisiert werden. Allerdings sind heutige NAS-Systeme nur bedingt dafür geeignet, Speicherplatz für I/O-intensive Anwendungen zur Verfügung zu stellen. Denn die Leistung von Netzwerk-Dateisystemen ist durch zwei Faktoren limitiert: (1.) Alle Datenzugriffe auf Netzwerk-Dateisysteme müssen durch einen einzelnen Fileserver geschleust werden. (2.) Heutige Netzwerk-Dateisysteme wie NFS und SMB sowie die zugrunde liegenden Netzprotokolle eignen sich nur bedingt für einen hohen Durchsatz. Wir haben mehrere Ansätze vorgestellt, diese Leistungsengpässe zu umgehen: auf RDMA aufsetzende Dateisysteme wie das Direct Access File System (DAFS) sowie NFS und SMB über RDMA, Shared-Disk-Dateisysteme wie das General Parallel File System (GPFS) sowie verteilte Dateisysteme wie das Hadoop Distributed File System (HDFS).

Bezug zu den folgenden Kapiteln

In den bisherigen Kapiteln haben wir grundlegende Techniken für Speichernetze vorgestellt. Viele dieser Techniken virtualisieren Speicherressourcen wie zum Beispiel RAID, Instant Copy, Remote Mirroring, Volume Manager, Dateisysteme und Snapshots. All diese Virtualisierungstechniken haben gemeinsam, dass sie den höheren Schichten vorhandene Speicherressourcen in einer Form anbieten, die einfacher zu verwalten und meist auch noch leistungsfähiger und verfügbarer ist als die zugrunde liegenden Ressourcen. Im nächsten Kapitel zeigen wir, wie speicherzentrierte IT-Systeme das Konzept der Speichervirtualisierung verändern und davon profitieren (Kap. 5). Im Anschluss stellen wir mit dem Objektspeicher eine weitere Technologie vor, die sich wie Dateisysteme für die Speicherung unstrukturierter Daten eignet (Kap. 6).

Index

B

C

D

E

F

G

H

I