Verteilte Backups

Überblick über Varianten zur Verteilung von Backups

Armin Langhofer und Karl Raab

Seminar aus Informatik
WS2008/2009

Inhaltsverzeichnis

1. Einleitung

Eine Datei bzw. ein ganzes Dateisystem ist gewissen Risiken ausgesetzt die zu erheblichen Datenverlust führen können. Das unbeabsichtigte Löschen oder Überschreiben von Dateien, eine Fehlfunktion oder Ausfall von Hardware wie durch Festplattendefekt, Feuer und höherer Gewalt oder auch durch Überschreitung der Lebensdauer von Speichermedien und ähnliche weitere Gegebenheiten führen zur Notwendigkeit Sicherungskopien anzulegen. Im Bereich der Privatwirtschaft und öffentlichen Betrieben und Verwaltungen sind angemessene Datensicherungen als verpflichtende Maßnahmen durch das Datenschutzgesetz sogar vorgeschrieben [DSG2000].

Definition Backup: „...das teilweise oder gesamte Kopieren der in einem Computersystem vorhandenen Daten auf ein alternatives (häufig transportables) Speichermedium“[BACWIK]

Verteiltes Backup: Als verteiltes Backup kann die Kombination eines *Backups* und das Nutzen von *Verteilten Systemen* gesehen werden.

Medium:

Um eine Datei auf einen Datenträger zu schreiben wird ein Medium benötigt. Kein Hersteller der heute verfügbaren Medien garantiert für die Daten die auf ein Medium geschrieben werden. Im Besten Fall wird Schadenersatz, im Regelfall nur normaler Ersatz des Mediums garantiert bei dem man bei Defekt ein („leeres“) Ersatzmedium erhält. CDs, DVDs, USB-Sticks (Flashspeicher) gelten als Transportmedien, Festplatten als operative Livemedien. Somit ist es nicht möglich Daten auf ein Medium zu schreiben und unter allen Umständen zu einem späteren Zeitpunkt wieder zu lesen. Es ist vielmehr ein (oft nur schwer) messbarer Prozentsatz, der angibt, mit welcher Wahrscheinlichkeit man die Daten wiederherstellen kann.

Die Wahrscheinlichkeit für eine langhaltige Speicherung zu ermitteln ist gerade bei neuen Medien gar nicht möglich, da ja Langzeittests erst nach einer langen Zeit abgeschlossen werden können und nicht vor, bei oder kurz nach der Erstproduktion des Mediums. Eine Lösung für dieses Problem ist, dass man die Daten auf einem Produktionssystem speichert (z.B. Festplatten im RAID-Verbund) und regelmäßig frische Backups anfertigt. Der Clou dabei ist, dass die Backups nie länger als eine bekannte Zeit auf dem Backupmedium gespeichert werden. Somit können auch ohne Langzeittests auf neuen Medien Sicherungskopien abgelegt werden, bei denen vorher nur die Wiederherstellungswahrscheinlichkeit für eine vergleichsweise kurze Zeit (zB 7 Tage) ermittelt wurde.

Abstrahiert man diesen Vorgang weiter, könnte man von Garantien für Wahrscheinlichkeiten sprechen, d.h. wird bei einem Kurzzeit-Test eine 90%ige Wiederherstellungswahrscheinlichkeit gemessen, können die Daten entsprechend x Tage zu y % Wahrscheinlichkeit wieder gelesen werden.

Amazon S3 [AMAZON] macht hier genau dasselbe: hier werden Wahrscheinlichkeiten für gewisse Funktionalitäten und damit Wahrscheinlichkeiten für Verfügbarkeiten garantiert.

Hochverfügbarkeit vs. Backup vs. Archiv:

Oft wird ein RAID-System mit einem Backupsystem oder mit einem Archivsystem verglichen und vertauscht – dabei sind die Anforderungen ganz unterschiedlich: Die Anforderung an ein hochverfügbares System ist, dass es beim Ausfall einer oder mehrerer Komponenten uneingeschränkt weiterarbeiten kann [Hochverfügbarkeit]. Die Anforderungen an ein Archivsystem hingegen ist es, die Daten auf ein Archivmedium zu kopieren/schreiben/brennen/… und das Medium so aufzubewahren dass man nach einem „langen“ Zeitraum diese Daten wieder lesen kann. Eine Papyrusrolle aus der Antike (die älteste wurde im 4. Jh.

v. Chr.) gefunden [PAPYRUS] wäre ein guter Vergleich für ein Archivsystem: Hier wurden „Daten“ (Schrift) auf das Medium geschrieben und können heute noch gelesen werden.

Die Anforderungen an ein Backup hingegen sind weder die langzeitige Speicherung noch die Hochverfügbarkeit eines Systems. Ein Backup soll gewährleisten, dass nach einem Datenverlust (durch manuelles Löschen, durch einen Virus, durch Versagen des Mediums,…) ein älterer Datenstand vorliegt bei dem im Idealfall nur wenig Unterschied zu den verloren Daten vorherrscht [DATENSICHERUNG].

Ein Archiv ist nur so gut wie sein Medium. Durch die Vielzahl und durch ständig neu erscheinende Medien ist es schwierig eine Aussage über die Langlebigkeit zu machen. Eine CD-ROM, wie Sie noch vor ein paar Jahren als aktuelles Medium galt, mag möglicherweise eine gewisse Zeit lang die Daten speichern, gilt aber trotzdem als Transport- und nicht als Archivmedium.

Mit Hilfe von Hochverfügbaren und Backupsystemen kann aber ein Archivsystem gebaut werden. Das ist beispielsweise der Weg wie ihn die österreichischen Landesarchive gewählt haben: Diese speichern ihre Daten auf einer NAS (Netzwerkspeicher, network attached storage) die regelmäßig gesichert wird. Fällt das NAS aus, wird auf das Backup zurückgegriffen um das NAS zu reparieren. So kann man selbst mit kurzlebigen Medien die Daten über einen großen Zeitraum verfügbar halten.

Verteiltes System:

Ein verteiltes (Datenverarbeitungs-)System besteht aus mehreren autonomen Prozessor-Speicher-Systemen, die mittels Datenaustausch kooperieren [Mühlhauser 2002; Informatik Handbuch, Hanser]

Im Zusammenhang mit verteiltem Backup sollte erwähnt sein, dass die hier referenzierte Definition eines verteilten Systems vom Grad der Verteilung abhängt. Ein Backup wäre nach dieser Definition schon verteilt wenn 2 Sicherungen auf 2 unabhängigen Prozessor-Speichersystemen abgelegt werden. Die tatsächliche Speicherung könnte aber laut dieser Definition auch auf einem geteilten Storage erfolgen, wodurch das System aber nicht mehr unabhängig arbeiten könnte da bei Ausfall des Storage keines der beiden Prozessor-Speicher Systeme in der Lage wäre eine Sicherung zurückzulesen.

Aus diesem Grund möchte man hier noch weiter greifen und die Abstufung anhand von Beispielen besser hervorheben:

- Mehrere Frontends, ein Backend
  Bei dieser Architektur werden die Sicherungen auf einem (zentralen) Speichersystem abgelegt und durch mehrere (unabhängige) Dienste angeboten.
- Unabhängige Storagesysteme
  Jeder der Backupserver verfügt über die Möglichkeit mind. eine Sicherung auf seinem eigenen Speichersystem abzulegen. Die Verteilung der Daten geschieht hier entweder von den zu sichernden Maschinen auf die Sicherungsserver oder innerhalb der Sicherungsserver. Im Bereich der Datenbanken könnte man hier zum Beispiel von Master/Slave oder Master/Master-Replikation unter den Backupmaschinen sprechen.
- Verteiltes Dateisystem
  Die Art und Weise der Synchronisierung der Datenstände zwischen den Backupservern ist nicht relevant für die Speicherung. Ein Schreibzugriff wird jedenfalls erst dann ohne Fehler bestätigt wenn die Daten so geschrieben wurden, dass eine Verteilung garantiert werden kann

Backup als zwei Komponenten:

C.Batten [Batten02] sieht ein Backup als zwei Komponenten. Einem persistenten Storage/Massenspeicher bzw. einem Dateisystem und einem Management/Versioning Framework wie auch z.B. CVS. Zum Einen müssen die zu sichernden Daten in irgendeiner Form abgelegt werden. Weiters ist es nötig für die gespeicherten Daten eine Methode zu verwenden, die es ermöglicht diese in einer Chronologie wieder zu finden und schließlich daraus eine Restauration durchzuführen.

Ein Massenspeicher ist typischerweise in gleich große Blöcke zur Ablage unterteilt. Ein Dateisystem ordnet eine Datei lediglich in Blöcken ein. Darüber hinaus werden Verzeichnishierarchien zur logisch strukturierten Ablage verwendet. Ein Dateisystem ist also eine Verwaltung von Dateien. Wo und auf welche Weise die Daten tatsächlich abgelegt werden, ist auf der Anwenderebene belanglos. Das heißt also, das die Daten auch verteilt auf verschiedenen Computersystemen vorhanden sein können. Dazu gibt es mittlerweile mehrere verteilte kernelbasierte Dateisysteme. Am bekanntesten sind das NFS [RFC1094], DFS [MSDFS] und AFS [AFS]. Es gibt ebenso die Möglichkeit, dass ein Dateisystem nicht im Kernel eines Betriebssystems vorhanden sein muss, sondern als Software im Userspace realisiert ist und auf bestehende Netzwerkprotokolle aufbaut [FUSE]. Somit können Dateien oder Teile davon ebensogut auf mehreren entfernten Computersystemen (Knoten) in einem Cloud in eine für den Benutzer unbestimmte Art persistent abgelegt werden. Es wird dazu lediglich eine geeignete Schnittstelle benötigt welche eine Abbildung auf die herkömmliche Sichtweise ermöglicht (siehe Illustration 1). Es ist dadurch ebenso möglich ein verteiltes Backup durch die Wahl eines geeigneten Dateisystems zu erreichen.

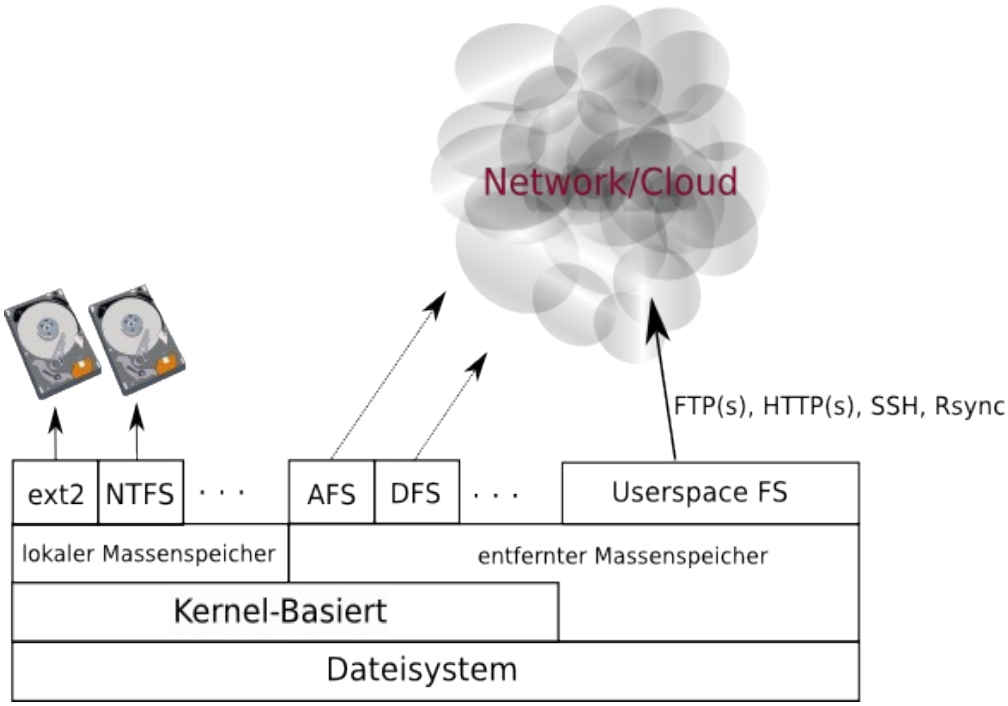


Illustration 1: Ebenen von Dateisystemen

Anforderungen:

An ein verteiltes Backup werden mehrere Anforderungen gestellt:

- Verfügbarkeit: Der Ausfall eines Knotens im System darf die Verfügbarkeit eines Backups nicht beeinträchtigen sowie das erstellen Neuer nicht behindern.
- Integrität: Alle Daten müssen vollständig und richtig Abgespeichert werden.
- Privacy/Confidentiality: Durch die Verteilung, auch auf möglicherweise nicht vertrauenswürdigen Knoten, müssen Vorkehrungen getroffen werden um eine missbräuchliche Verwendung zu verhindern. Es darf also nur derjenige auf Daten Zugriff bekommen, der dazu berechtigt ist.

2. Grundlagen zur Erstellung von Backups

Bei jedem Backup muss zuerst ein organisierter Zugriff auf die zu sichernden Daten vorgenommen werden. Dahingehend gibt es mehrere Methoden die sich auf verschiedenen Ebenen unterscheiden können. Zuerst stellt sich die Frage ob Device-basierende oder Datei-basierende Techniken zum Einsatz kommen sollen. Weiters, ob und welche Historien verwaltet werden. D.h. vollständige Backups (full backups) oder inkrementelle Backups (incremental backup). Ob das zu sichernde Dateisystem offline sein muss oder ob dies während des Betriebes möglich ist. Zuletzt ist es entscheidend welches Speichermedium zur Ablage der Backups benutzt werden soll. Durch die Verbreitung des Internets haben sich hier einige neue Möglichkeiten entwickelt die zur einer Erhöhung der Redundanz und Integrität sowie Beschleunigung der Wiederherstellung (recovery) genutzt werden können [vgl. Chervenak].

2.1.Device-basierende und Datei-basierende Methoden

Festplatten sind in Blöcke unterteilt in denen Dateien abgelegt werden. Dabei ist es möglich, das vor allem größere Dateien nicht kontinuierlich sondern zerstückelt gespeichert sind. Dies ist bei der Device-basierenden Methode zu berücksichtigen. Hier werden ohne Rücksicht auf Dateistrukturen einfach diese Blöcke kopiert. Da dadurch Seek-Operationen entfallen, ist dies eine äußerst performante Methode Daten zu sichern. Jedoch ist eine Wiederherstellung von einzelnen Dateien äußerst aufwändig und langsam. Es muss ein Mechanismus zur Indizierung von Dateien auf dem Sicherungsmedium verwendet werden. Dies ist jedoch sehr stark Systemabhängig und kaum portabel.

Datei-basierte Backups hingegen sind nicht von dem physischen Medium abhängig. Dateien werden hier kontinuierlich, also nicht wie tatsächlich auf einer Festplatte o.ä. gespeichert, auf das Sicherungsmedium kopiert. Diese Methode ist im Vergleich zwar langsamer, hat aber den Vorteil das einzelne Dateien für eine Sicherung ausgewählt werden können. So ist es möglich nur Dateien zu sichern, die seit dem letzten Backup verändert wurden [vgl. Chervenak].

2.2.vollständiges, differenzielles und inkrementelles Backup

Beim vollständigen Backup wird schlichtweg das komplette Dateisystem kopiert. Kommt es zu einem Defekt, kann dieses Backup einfach wieder aufgespielt werden und das System ist wieder einsatzbereit. Diese einfache Möglichkeit ermöglicht zwar eine schnelle Wiederherstellung, benötigt jedoch sehr viel Speicherplatz und verursacht bei wiederholten Backups extrem viele Duplikate und die Speicherung dauert lange.

Eine Alternative sind differentielle Backups bei dem immer der Stand seit dem letzten Vollbackup gesichert werden. Man benötigt zum Wiederherstellen das letzte volle Backup und **die letzte** differenzielle Sicherung.

Eine weitere Alternative sind inkrementelle Backups bei denen nur Dateien gesichert werden, die seit dem letzten Backup verändert wurden. Zur vollständigen Wiederherstellung benötigt man das letzte volle Backup und **alle** inkrementellen Sicherungen.

Inkrementell oder differentiell sichern spart Zeit bei der Sicherung und Speicherplatz am Sicherungsmedium. Ein klassisches inkrementelles Szenario beginnt mit der Erzeugung eines vollständigen Backups. Anschließend werden lediglich die neu hinzugekommenen und veränderten Dateien gesichert. Da bei einer kompletten Wiederherstellung jedes inkrementelle Backup benötigt wird, ist es ratsam Zwischenschritte einzufügen was zu einer Art logarithmischen Sicherung führt. Beispielsweise kann folgende inkrementelle Sicherungsstrategie benutzt werden:

1.inkrementelle Sicherung jeden Tag

2.vollständige Sicherung der inkrementellen Änderungen einer Woche

3.vollständige Sicherung der inkrementellen Änderung eines Monats

4.vollständige Sicherung jedes Jahr

Wie bereits erwähnt dauert beim inkrementellen Backup eine Wiederherstellung länger, da mehrere Backups benötigt werden. Die Verwendung einer logarithmischen Strategie schafft hier jedoch Abhilfe [vgl. Chervenak].

2.3.Mischform

Es ist möglich eine vollständige Sicherung und eine inkrementelle Sicherung zu kombinieren. Ein gutes Beispiel mit einfachen Mitteln ist zum Beispiel rsnaphshot [Rsnapshot]. Hier wird geschickt eine Basisfunktionalität von vielen Dateisystemen wie ext3, reiserfs, xfs, jfs etc. ausgenutzt: Das Hardlinkkonzept. Ein Hardlink ist ein Verweis auf eine Datei, die im Dateisystem zwar öfter aufscheint, physisch aber nur einmal verspeichert ist. Somit ist es möglich auf dem physischen Speicherplatz für eine Datei viele Verlinkungen an andere Stellen im Dateisystem zu erzeugen.

Rsnapshot erzeugt bei der Initialsicherung ein 1:1 Abbild der Originaldaten im Sicherungsdateisystem. Werden die Daten nun nicht verändert und ein zweiter Sicherungslauf gestartet, werden alle Dateien per Hardlink „dupliziert“, d.h. wechselt man mit einem Filebrowser in das Verzeichnis der ersten Sicherung sieht man exakt dieselben Daten wie im Verzeichnis der zweiten Sicherung. Wird nun vor dem dritten Sicherungslauf eine neue Datei im Quellverzeichnis angelegt werden im Zuge der Sicherung alle Dateien wie gehabt verlinkt und die neue Datei zusätzlich ins Verzeichnis kopiert. Es wurde also nach der 3. Sicherung der Speicherplatz der 1. Sicherung und dem Speicherplatz der neuen Datei verbraucht.

Aus Sicht des Speicherplatzverbrauches würde man hier wohl von einem inkrementellen Backup sprechen. Sieht man jedoch aus Sicht eines Benutzers aufs Dateisystem würde dieser die Backups vermutlich als Fullbackups beschreiben.

2.4.On-Line Backup, Snapshot

Im einfachsten Fall wird für ein Backup das Stillstehen des gesamten Dateisystems verlangt. In dieser Zeit dürfen keinerlei Dateioperationen stattfinden. Das ist aber meist, vor allem bei hochverfügbaren Systemen, nicht erreichbar. Es wirft jedoch einige Probleme auf, wenn auf zu sichernde Daten zugegriffen wird. Am stärksten ist der Effekt wenn die Verzeichnisstruktur verändert wird. Wird während des Backup-Vorgangs ein Verzeichnis verschoben, kann es zu Inkonsistenzen kommen. Hat der Sicherungsvorgang ein Verzeichnis zur Sicherung registriert bzw. befindet dieser sich bereits innerhalb der Traversierung des Verzeichnisses, kann es passieren, dass das Verzeichnis „übersehen“ wird, da es zum

Zeitpunkt des tatsächlichen Kopiervorgangs nicht mehr vorhanden ist. Wird das Verzeichnis verschoben nachdem der tatsächliche Kopiervorgang durchgeführt wurde, ist es möglich dass dieses nochmals, also doppelt, gesichert wird. Es kann, abhängig vom verwendeten Dateisystem, auch vorkommen dass eine Mixtour aus der alten und neuen Verzeichnisstruktur entsteht.

Problematisch sind auch Transformierungen, Änderungen oder Löschungen von Dateien. Wird eine Datei komprimiert und an einem anderen Ort gespeichert und das Original gelöscht kann es passieren, dass die neue Datei ausgelassen wird, da diese zum Zeitpunkt der Sicherung noch nicht existierte, sowie, dass das Original auch nicht gesichert wird, da es zum Zeitpunkt der Sicherung nicht mehr vorhanden ist.

Wird eine Datei während des Kopiervorgangs auf das Sicherungsmedium verändert, entsteht womöglich eine inkonsistente Kombination aus der alten und neuen Version.

Um diese Probleme zu umgehen gibt es zwei Möglichkeiten. Entweder die zu sichernden Daten werden auf Änderungen gesperrt oder es wird nach der Sicherung ein Vergleich mit dem On-Line System und dem Backup durchgeführt. Nicht gesicherte Verzeichnisse und Dateien werden dann für das nächste Backup vorgemerkt.

Eine weitere Methode sind sogenannte Snapshots – ein konsistentes Abbild vom Dateisystem. Hierbei wird das Dateisystem für einen kurzen Moment angehalten (frozen) und die Verzeichnisstruktur kopiert. Dann wird mit der Erstellung des Snapshots begonnen. Alle in dieser Zeit stattfindenden Dateioperationen werden an anderer Stelle auf dem Speichermedium durchgeführt. D.h. das Original wird kopiert und mit der Kopie wird weitergearbeitet. Die Verzeichnisstruktur des Snapshots bleibt somit erhalten. Dieser Vorgang wird Copy-on-write genannt [vgl. Chervenak].

3. Backup zu (verteilten) Netzwerken

Für die Verteilung eines Backups in einem Kommunikationsnetzwerk betrachten wir 2 grundlegende Modelle. Das Client/Server Modell und ein generelles Peer-To-Peer (P2P) Modell (siehe Illustration 2 und 3). Die Darstellung eines Servers entspricht dabei ebenso einer Objektablage auf die die Clients zugreifen.

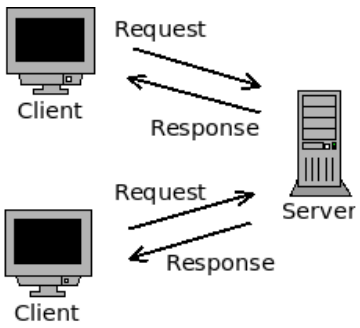


Illustration 2: Client/Server Modell

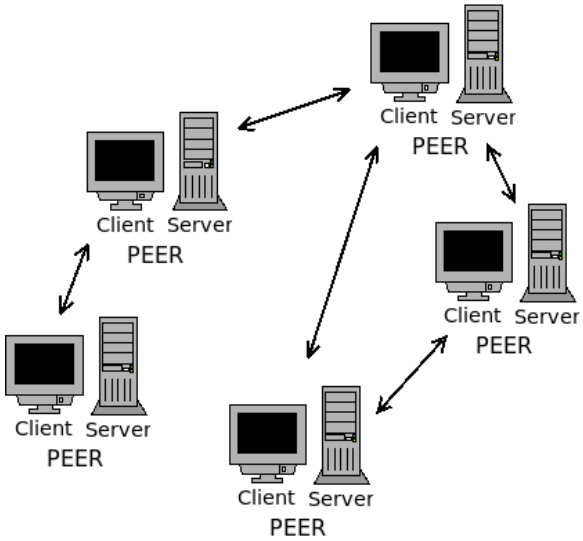


Illustration 3: P2P Modell

3.1. Client/Server Modell

Das besondere beim Client/Server Modell im Bezug auf Sicherungen ist, dass der Client immer eine Verbindung zum Server voraussetzt und der Client nur mit dem Server „verbunden“ ist. Es können dabei mehrere Server, mehrere Clients und auch Clients dabei sein, die selbst Serverfunktionalität bieten aber letztlich verbindet sich ein Client immer zu einem (logischen) Server. Im weiter unter beschriebenen P2P hingegen ist jeder mit jedem verbunden, auch wenn vielleicht zu einem bestimmten Zeitpunkt nicht immer alle „Peers“ (Partner) online und dadurch verfügbar sind.

Bei der Sicherung eines Datenstandes von einem Client auf den Server muss mindestens ein Knoten zur Verfügung stehen auf den geschrieben werden können. Es ist nicht zwingendermaßen erforderlich dass alle Serverknoten schreiben können. Das Servernetzwerk verteilt die Daten innerhalb des Netzwerkes auf eine unbekannte Art und Weise sodass die anderen Serverknoten zumindest das Backup ausliefern können. Um dem Client eine Sicherung wieder einspielen zu können ist mind. ein R/O-Knoten und um das Backup anfertigen zu können wird mindestens ein R/W Knoten benötigt.

3.1.1.Amazon S3

Als klassisches Beispiel für einen modernen Netzwerkspeicher dient Amazon S3 [AMAZON]. Nach Anlegen eines Buckets (geteilter Namespace, limitiert in Anzahl pro Benutzer) kann man beliebig viele selbst vergebene Schlüssel erzeugen um darin Daten - sogenannte Objects - zu speichern.

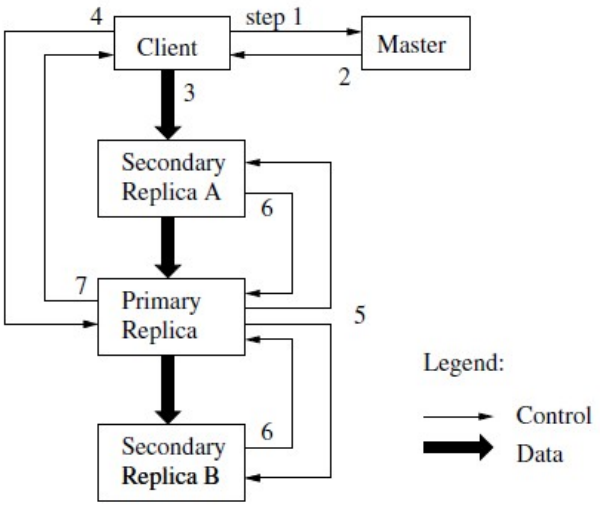
Der Zugriff geschieht dann entweder lesend oder schreibend wobei beide Zugriffsrechte pro Schlüssel definiert werden können. Ein Benutzer kann also maximal folgende Zugriffsrechte auf einen Schlüssel erhalten: READ, WRITE, READ_ACP, WRITE_ACP und FULL_CONTROL. Bei den {READ,WRITE}_ACP Rechten darf der Benutzer selbst wieder für andere Bentuzer die Zugriffsrechte vergeben.

Auf Ein Bucket in Amazon S3 greift man per Client/Server-Modell zu. Ein Client verbindet sich mit Amazon S3, schreibt oder liest seine Daten und trennt die Verbindung wieder. Dahinter verbirgt sich aber natürlich ein großes Netz aus einer Vielzahl von Knoten um so viele Clients weltweit bedienen zu können. Dem Client wird verborgen, dass hinter seiner Gegenstelle (dem „Server) ein großes verteiltes System steckt.

3.1.2.Google FS

Das nicht öffentlich verfügbare Google Dateisystem [GFS] wird von Google für die Speicherung der Daten verwendet. Dabei sind die Anforderungen bei Google wohl folgende: große Dateien (mehrere Gigabytes) und hauptsächlich wachsende Dateien (es werden selten Daten gelöscht und es werden selten Dateien verkleinert). Für diese Anforderungen wurde eine große Blockgröße von 64MB gewählt, die unüblich im Vergleich zu anderen verteilten Dateisystemen sind.

Das Basiskonzept sieht **einen** (im Thema der Hochverfügbarkeit und Skalierbarkeit ist es nie gut wenn nur eine Komponente nur einmal vorkommen darf) Masterserver und viele Chunkserver vor. Die Masterserver instruieren den Client wo er seine Daten ablegen soll und verwaltet dann die Chunks lokal.



[GFS]

Der Client spielt dabei seine Daten bei Secondary Replica A und Primary Replica ein. Primary Replica kann so konfiguriert sein, dass er die Daten zusätzlich noch auf Secondary Replica B kopiert ohne dass der Client dies merkt.

3.2.Peer-To-Peer Modell

3.2.1.Einführung in P2P Protokolle

Wie bereits erwähnt haben sich durch die zunehmende Verbreitung des Internets neue Netzwerktechnologien entwickelt. Bei P2P Protokollen übernimmt ein Client ebenso die Aufgabe eines Servers. Die Kombination eines Clients und Servers zu einer Instanz nennen wir hier Peer. Die Ablage von Objekten (z.B. Dateien) unterliegt nicht mehr ausschließlich einem Server, sondern zu gleichen Teilen auch den Peers. Allgemein bekannt wurden P2P Netzwerke durch Filesharing von Musikdateien durch Napster [NAPST], Gnutella [GNUT] und ähnliche. Bei diesen Netzwerken werden an einem Peer vorhandene (Musik-)Dateien für alle weiteren Nutzer im System/Cloud zum Download zur Verfügung gestellt. Will ein Peer auf eine Datei eines anderen Peers zugreifen, dann geschieht dieser Zugriff direkt auf den entsprechenden Knoten.

Es besteht nun die Hauptschwierigkeit bei P2P Protokollen in der entsprechenden Informationsfindung, auf welchem Peer sich welche Dateien befinden (lookup). Es haben sich dabei verschiedene Objektabfragemechanismen entwickelt, welche sich in erster Linie durch den Grad der Dezentralisierung unterscheiden lassen. Napster und Gnutella entsprechen den zwei Gegensätzen bei der Strukturierung von P2P Protokollen:

Napster benötigt für die Suche von Dateien eine *zentrale Datenbank*, einen sogenannten Verzeichnisdienst. Kommt ein Peer im System hinzu, sendet dieser eine Liste seiner bereitgestellten Dateien an die Datenbank. Die Dateien verbleiben jedoch am Peer. Durch die Kontaktaufnahme mit dem Verzeichnisdienst geschieht somit auch gleichzeitig eine Anmeldung im P2P System. Eine Suchabfrage eines beliebigen Peers wird nur über den zentralen Verzeichnisdienst durchgeführt. Dieser gibt als Antwort eine Liste aller passenden verfügbaren Dateien im System an den Peer zurück. Ist nun eine Datei zum Download gefunden, wird der Peer an dem die Datei liegt direkt kontaktiert und diese übertragen.

Diese zentralisierte Struktur hat zwar den Vorteil, das Suchabfragen von anderen Peers durch die zentrale Datenbank nicht weitergeleitet werden müssen (query routing), jedoch kann diese Struktur nicht vollständig autonom aufrecht erhalten werden. Durch Ausfall des Verzeichnisdienstes wird diese Art von P2P-Protokollen vollständig lahmgelegt. Es gibt also einen „single point of failure“.

Gnutella geht hier einen vollständig umgekehrten Weg. Es gibt *keine Zentralisierung*. Die Informationsverbreitung wird vollständig von Peer zu Peer durchgeführt. Um nun in diesem Netzwerk teilzunehmen, muss mindestens ein Peer zum „Einstieg“ bekannt sein. Bei Kontaktaufnahme retourniert dieser eine Liste weiterer Peers auf welche wiederum zugegriffen werden kann. Die Durchführung einer Suchabfrage um eine Datei zu finden, wird über die direkten Nachbarn durchgeführt. Diese senden empfangene Abfragen wiederum an ihre Nachbarn weiter und geben deren Resultat, inklusive ihren Eigenen, zurück. Dieses „Flooding“ wird durch eine TTL auf eine gewisse Tiefe beschränkt.

Der größte Nachteil bei dieser Art der Informationsfindung ist das Entstehen großer Lasten ausschließlich für die Übermittlung der Abfragen und Ergebnisse. Jedoch ist diese Struktur unempfindlich gegenüber Ausfällen von Knoten.

In weiteren Generationen von P2P-Protokollen haben sich schließlich Mischformen bewährt. Durch Strukturierung des Netzwerkes mittels Meshes, Ringe, Torus und weitere, wird erreicht, das eine zentraler Servers bzw. ein simples Flooding in der oben vorgestellten Form nicht mehr notwendig ist. Als Beispiel sei hier Chord [CHORD] aufgeführt. Das in der Literatur häufig vorzufindende Modell, verwendet eine sogenannte distributed hash table [DHT]. Hierbei werden zu den Datenobjekten globale Schlüssel vergeben. Diese Value/Key Paare ergeben in Summe eine globale Hashtabelle. Tatsächlich verwaltet jeder Peer nur einen Teil dieser Tabelle. Es wird also in Intervalle aufgeteilt. Bei Chord wird nun ein Ring aufgebaut bei dem jeder Peer nur für einen Teil weiß, welche Daten vorhanden sind. Bei der Suche wird nicht der gesamte Ring durchlaufen, sondern benutzt Verweise auf entfernte Peers – die sogenannte Fingertabelle. Aufgrund dieser lassen sich Suchabfragen äußerst effizient mit einem Aufwand von O(log n) durchführen (siehe Illustration 4)

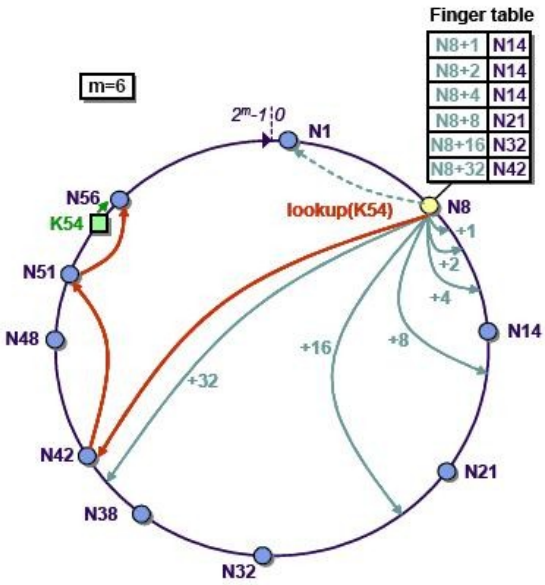


Illustration 4: Chord Ring - Dieses Bild verdeutlicht den Lookup Mechanismus von K54 [WPChord]

3.2.2. Backup mittels Peer-to-Peer Techniken

Es ist wie oben vorgestellt klarerweise nicht nur möglich die durch Filesharing-Programmen allgemein bekannte Verbreitung von Dateien durchzuführen, sondern auch P2P Topologien zur Bereitstellung von Speicherplatz zu verwenden. Es lassen sich dadurch sogar verteilte Dateisysteme bewerkstelligen [FARSITE].

Anhand eines Beispieles betrachten wird die Nutzung von P2P Technologien für Backup Zwecke. C. Batten führte eine Untersuchung mit dem eigens entwickelten Projekt pStore durch [Batten02]. Will man in diesem System z.B. 10MB Backup-Speicherplatz zur Verfügung gestellt bekommen, muss dafür 10MB Speicherplatz für andere Benutzer freigegeben werden. Es ist also der nutzbare Speicherplatz für Backups im pStore System proportional zu dem Speicherplatz der lokal für andere Nutzer zur Verfügung gestellt wird. Als Basis für die P2P Kommunikation wird das vorhin erwähnte Chord verwendet.

Bei pStore werden die zu sichernden Dateien in File-Blocks (FB) unterteilt und in einer File-Block Liste (FBL) Referenziert. Die FBL beinhaltet einen Identifier, einen Hash der Daten, die Länge des Blocks und einen Offset zum Dateibeginn. Die FBL verbleibt auf dem Peer und die verschlüsselten FBs werden auf die anderen Peers verteilt. Das Wiederfinden von FBs geschieht schließlich über ein Lookup der Identifier. Die Übertragung der verschlüsselten FB zwischen den Peers wird über rsync [RSYNC] bewerkstelligt. In Anlehnung an CVS werden bei Änderungen von Dateien lediglich diese übertragen. Dies spart Speicherplatz und Bandbreite. In Illustration 5 ist eine FBL mit den FBs schematisch Dargestellt.

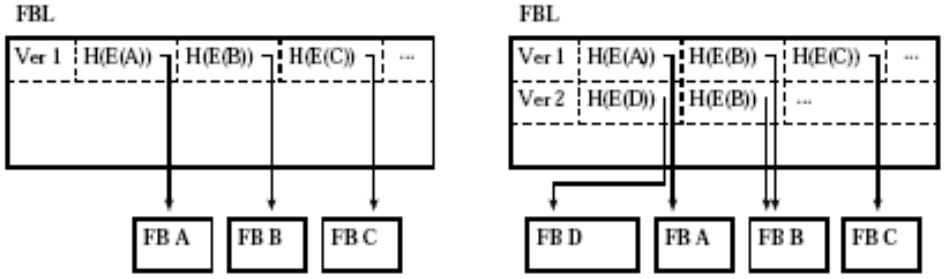


Illustration 5: Versioned File Block List [Batten02]

Ein zu bewerkstelligendes Problem sind Ausfälle von Peers, da somit Teile einer Datei nicht mehr verfügbar würden. Daher werden FB mehrmals auf unterschiedlichen Peers repliziert. Die Untersuchungen haben ergeben, dass die Wahrscheinlichkeit das eine Datei mit k Blöcken welche r mal repliziert wird nicht verfügbar ist, bei $1-(1-f')^k$ liegt, wobei f die Anzahl der ausgefallenen Peers ist.

Es kann somit durch ausreichende Erhöhung der Replikation eine hohe Ausfallssicherheit erreicht werden. Betrachtet man die Bandbreitennutzung, wird durch die gleichzeitige Aufteilung der Daten auf mehrere Peers eine starke Erhöhung des Durchsatzes möglich. Bei der klassischen Client/Server Kommunikation ist die Geschwindigkeit durch einen einzigen Kanal begrenzt. Im P2P Netzwerk ist der mögliche Durchsatz durch die Summe aller Bandbreiten der Peers gegeben.

Da im Allgemeinen entfernte Peers nicht als vertrauenswürdig angesehen werden können, müssen wie bereits erwähnt die Daten verschlüsselt werden. Es ist für den tatsächlichen Einsatz notwendig im Bereich von Privacy einige Vorkehrungen zu treffen um die missbräuchliche Verwendung von sensiblen Daten zu verhindern.

Ob sich tatsächlich solch ein System jemals bewähren und durchsetzen wird, ist abzuwarten. Die Anschaffungskosten von Massenspeichern (Festplatten) werden immer geringer und durch die Delegation der persistenten Datenablage zu fremden unbekannten nicht vertrauenswürdigen Personen (man hat die Daten nicht mehr selber unter Kontrolle) entsteht ein subjektiver Unsicherheitsfaktor. Jedoch haben sich bereits mehrere kommerzielle Anbieter gefunden, welche auf diesem P2P-Prinzip beruhen [Allmydata, WuaLa].

4. Referenzen

[DSG2000]	Datenschutzgesetz 2000 (DSG 2000), BGBl. I Nr. 165/1999.
[BACWIK]	Wikipedia. http://de.wikipedia.org/wiki/Backup
[Batten02]	C. Batten, K. Barr, A. Saraf and Stanley Trepetin. “pStore: A Secure Peer-to-Peer Backup System.”. *MIT, Technical Memo*, Massachusetts Institure of Technology Laboratory for Computer Science, October 2002.
[RFC1094]	Sun Microsystems. “NFS: Network File System Protocol specification.”. United States. 1989.
[MSDFS]	Microsoft. “Distributed File System”. http://www.microsoft.com/windowsserver2003/technologies/storage/dfs/default.mspx, United States
[AFS]	J.H. Howard, “An Overview of the Andrew File System”. Information Technology Center, Carnegie Mellon University, Technical Report, http://reports-archive.adm.cs.cmu.edu/anon/anon/home/ftp/itc/CMU-ITC-062.pdf, 1988.
[FUSE]	M. Szeredi. “Filesystem in USEr space.”. http://sourceforge.net/projects/avf, 2003.
[vgl. Chervenak]	A.L. Chervenak, V. Vellani and Z. Kurmas. “Protecting File Systems: A Survey of Backup Techniques”. In *Proceedings of the Joint NASA and IEEE Mass Storage Conference*, March 1998.
[NAPSTER]	Napster: http://en.wikipedia.org/wiki/Napster
[GNUT]	Gnutella: http://wiki.limewire.org/index.php?title=GDF
[CHORD]	I.Stoica, R.Morris, D.Karger, F.Kaashoek and H.Balakrishnan, "Chord: A scalable peer-topeer lookup service for Internet applications.". *In Proceedings of ACM SIGCOMM’2001*, August 2001.
[DHT]	H.Balakrishnan, M.F.Kaashoek, D.Karger, R.Morris and I.Stoica. “Looking up data in P2P systems.”. *In Communications of the ACM*, February 2003.
[WPChord]	P2P Wiki: http://ast-deim.urv.cat/wiki/Chord
[FARSITE]	W.J.Bolosky, J.RDouceur and J.Howell. “The Farsite project: a retrospective.”. *In ACM SIGOPS Operating Systems Review*, v.41 n.2, p.17-26, April 2007.
[RSYNC]	rsync: http://samba.anu.edu.au/rsync/
[ALLMYDATA]	Allmydata Inc. http://www.allmydata.com
[WUALA]	Caleido AG. http://wuala.com
[RSNAPSHOT]	Rsnapshot.org http://rsnapshot.org
[Hochverfügbarkeit]	Wikipedia. http://de.wikipedia.org/wiki/Hochverf%C3%BCgbarkeit

[PAPYRUS]	Wikipedia. http://de.wikipedia.org/wiki/Papyrus_(Beschreibstoff)
[DATENSICHERUNG]	Wikipedia. http://de.wikipedia.org/wiki/Datensicherung
[AMAZON]	Amazon, Amazon Simple Storage Service, http://aws.amazon.com/s3/
[GFS]	S.Ghemawat, H.Gobioff and S.-T.Leung, „The Google File System“, Google Inc. USA, 2003, http://labs.google.com/papers/gfs-sosp2003.pdf