Das Z-Dateisystem

Speichern – aber sicher!

Jan Schampera <jan.schampera@unix.net>

November 2007

Inhaltsverzeichnis

1 Einführung

Tagtäglich werden Unmengen an Informationen gesammelt, ob wichtige oder unwichtige, und auf Massenspeichern abgelegt. Durch Dateinamen und in Verzeichnissen geordnet werden sie verwaltet.

Diese Daten sind relativ sicher und normalerweise nur durch unbeabsichtigtes Löschen oder Totalverlust eines kompletten Dateisystems gefährdet. Ein regelmäßiges Backup soll dies verhindern.

Im Folgenden wird kurz dargestellt, was diese Vorgehensweise in der Praxis für die Sicherheit der Daten bedeutet und wie das Dateisystem ZFS einigen der entstehenden Probleme entgegenwirken kann.

2 Klassische Behandlung von Massenspeichern

Traditionell werden zwischen dem physikalischen Datenträger und der Ansicht durch den Benutzer Abstraktionsebenen eingefügt: *Volume-Management* und *Dateisystemschicht*. Diese Vorgehensweise ergibt sich historisch aus der Notwendigkeit, mehrere Festplatten zusammen zu fassen um entweder mehr Kapazität (beispielsweise *RAID-0*), höhere Verfügbarkeit (beispielsweise *RAID-1*) oder auch beides zu erreichen[1].

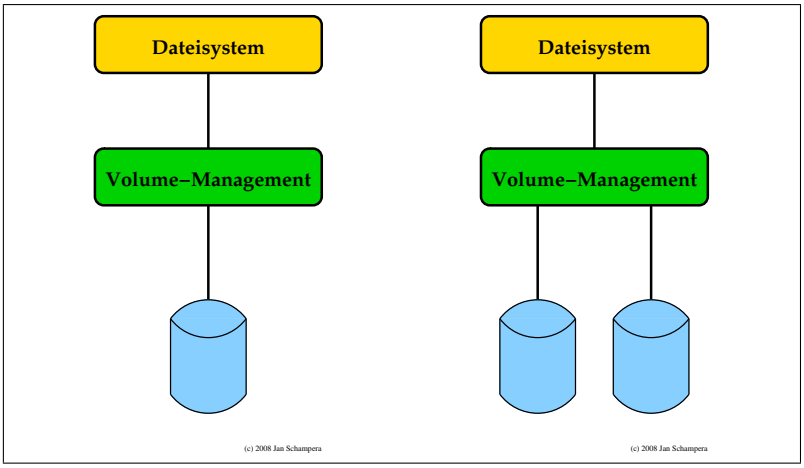

Abb. 2.1: *Klassische Nutzung der Massenspeicher über Volume und Dateisystem*

2.1 Volume-Management

Ein *Volume* ist die logische Betrachtung eines physikalischen Massenspeichers durch das Betriebssystem. Der Umweg wird benötigt, um die Gerätebehandlung an Anforderungen oder Eigenheiten des Betriebssystems auf Blockebene anzupassen. Modernere

[1] RAID – Redundant Array of Inexpensive Disks: Verfahren zur Performacesteigerung, Absicherung oder Kapazitätserhöhung unter Ausnutzung von mehreren Massenspeichergeräten

Volume-Manager können problemlos mehrere physikalische Massenspeicher zu einem einzigen logischen Volume zusammenfassen und gemeinsam als eine Art „virtuellen Datenträger" nutzen.

Auch Dinge wie für das Dateisystem transparente Datenkompression oder Verschlüsselung werden hier implementiert.

2.2 Dateisysteme

Dateisysteme stellen eine der obersten Ebenen bei der Abstraktion der Speichersysteme gegenüber dem Nutzer dar. Sie sind wohl auch die bekannteste Ansicht von Massenspeichersystemen.

Klassische Dateisysteme gehen aber von einer Reihe von Annahmen aus, die zum Teil auf veralteten oder unrealistischen Gegebenheiten beruhen. So wird in aller Regel unterstellt, dass der Datenpfad zwischen dem Computersystem und dem Massenspeicher zuverlässig und ungefährdet sei. Doch in den Zeiten von Speichernetzwerken (SAN) und verteilten Massenspeichern, Vielwegekommunikation und anderen modernen Verfahrensweisen ist dies definitiv nicht immer der Fall. Auch erhöhen diese Strukturen die Gefahr der unbemerkten, gewollten Datenmanipulation.

Nicht zuletzt die breite Verwendung von billigen Massenspeichern birgt Risiken – für die Sicherheit und Konsistenz der Daten – denen klassische Dateisysteme schutzlos gegenüberstehen.

3 Gängige Sicherheitsmaßnahmen

Normalerweise versucht man, mit einer Reihe von gängigen Maßnahmen einem Teil- oder Totalverlust der Daten vorzubeugen oder ihn bei Eintritt zu kompensieren.

Natürlich sollte mit einer Reihe von Maßnahmen einem Teil- oder Totalverlust der Daten vorgebeugt werden oder dieser bei Eintritt kompensiert werden können.

3.1 Mehrfachspeicherung der Daten

Eine einfache, aber auch risikoreiche Methode, Daten auf Massenspeichern zu sichern ist die Benutzung einer einzigen, nicht abgesicherten Disk (z.B. Festplatte).

Um den Risiken beim *Single-Disk-Setup* zu entgehen, verwendet man Verfahren, die die Nutzdaten mehrfach auf verschiedenen Datenträgern vorhalten, um im Falle des Versagens einer Disk

- für die Dauer des Ausfalls korrekte Daten zu liefern
- ein eingefügtes Ersatzmedium sofort wieder mit einer Kopie der Daten zu versorgen

Diese Funktionen sind in aller Regel für das Betriebssystem transparent auf Hardwarebasis realisiert. Natürlich gibt es aber auch Softwarelösungen im Volume-Management oder darunter (z.B. im Gerätetreiber).

3.2 Blockprüfsummen

Damit die Festplattenelektronik, oder auch das Betriebssystem selbst, sofort fehlerhafte Daten erkennen können, werden für diese manchmal Prüfsummen berechnet und direkt zusammen mit den Daten abgespeichert. Sollte die beim Lesen errechnete Prüfsumme von der gespeicherten abweichen, so ist der betreffende Datenblock defekt.

Im Bereich der RAID-Technik werden solche Verfahren genutzt, um im Falle verschiedener Datenstände auf den redundanten Festplatten festzustellen, welche Datenblöcke unversehrt sind. Dies kann wichtig sein, um bei Bedarf festzustellen, *welche* Daten wiederhergestellt werden sollen.

3.3 Backups

Die gängigste Sicherheitsmaßnahme gegen Ausfall oder andere Desaster-Szenarien ist wahrscheinlich das einfache *Backup*. Backups sind *unbedingt notwendig* und kaum durch andere Absicherungsmaßnahmen zu ersetzen. Allerdings ergeben sich in der Praxis auch einige kleine Nachteile:

- Zentralbackups sind sehr zeitaufwendig und verbrauchen teilweise enorme Ressourcen
- Volles Recovery aller Daten ist oft zu langwierig (veraltete Datenstände)
- Recovery einzelner Dateien ist in der Regel unverhältnismäßig aufwendig

4 Gefährdung der Daten

Im Folgenden werden einige Punkte genannt, die die Sicherheit und Konsistenz von gespeicherten Daten gefährden können und es im Alltag auch tun. Die schon beschriebenen Absicherungsmaßnahmen decken einen Teil dieser Risiken ab.

4.1 Teilausfall einer nicht-redundanten Disk

Durch Materialfehler oder Störungen bei der Magnetisierung kommt es nicht selten zu einer Teilbeschädigung der Daten auf einer Festplatte [2]. Auch die vom Hersteller veranschlagte Lebensdauer spielt hier natürlich eine Rolle.

[2] Natürlich trifft das auf alle Arten von Magnetspeichern zu (z.B. Disketten oder Magnetband)

Die beschädigten Bereiche können dann nicht mehr oder nur fehlerhaft gelesen bzw. beschrieben werden. Der Inhalt einer Datei, die einen solch beschädigten Bereich nutzt, ist somit teilweise verstümmelt. Auch wird sich bei solchen Fehlern die Organisationsstruktur des Dateisystems langsam und schleichend zersetzen.

In der Regel werden solche Fehler viel zu spät erkannt, um noch Gegenmaßnahmen einzuleiten die alle Daten zu 100% erhalten würden.

4.2 Totalausfall einer nicht-redundanten Disk

Ein eher selten vorkommender Fehler ist der sofortige totale Ausfall einer kompletten Festplatte. Dies kann durch grobe mechanische oder thermische Beanspruchung, öfter aber durch elektrische Zerstörung (Überspannung) geschehen.

Ein solcher Ausfall wird normalerweise sofort erkannt, da mit der Festplatte auch sämtliche auf ihr gespeicherten Daten verloren gehen.

4.3 Fehlererkennung nur bedingt möglich

Häufig werden aus Redundanzgründen wichtige Daten einfach automatisch gespiegelt und doppelt vorgehalten. Dadurch wird erreicht, dass die Daten beim Totalausfall einer der beiden Disks (Spiegelhälften) zu 100% vorhanden sind.

Eine gewisse Problematik ergibt sich nur dann, wenn beide Spiegelhälften unterschiedliche Daten aufweisen – welcher Datenbestand ist der richtige? Welche Spiegelhälfte enthält die Daten, mit denen die jeweils andere Spiegelhälfte korrigiert werden muss?

Ein Ausweg aus dieser Situation ist nur möglich, wenn der RAID-Controller, der für die Spiegelung verantwortlich ist, Prüfsummen transparent mit auf den Disks abspeichert und beim Lesen gegenprüft. Da aber diese Prüfungen ohne Zutun und Wissen vom Betriebssystem durchgeführt werden, kann dieses im Endeffekt nicht *selbst* entscheiden, ob die Daten wirklich in Ordnung sind.

4.4 Ungesicherte Datenwege

Dieser Bereich ist eine Problematik an die man nicht in erster Linie denkt, wenn es um Sicherheit der abzulegenden Daten geht. Trotzdem kommt es relativ häufig vor, dass durch fehlerhafte Hardwarekomponenten, oder gar absichtlich, Daten verfälscht oder unbrauchbar gemacht werden:

- Fremdlicht bzw. Fremdspannung im Übertragungsmedium
- Fehlerhafte Firmware
- Fremdmagnetisierung von magnetischen Speichern
- Beabsichtigte Verfälschung von transportierten Daten

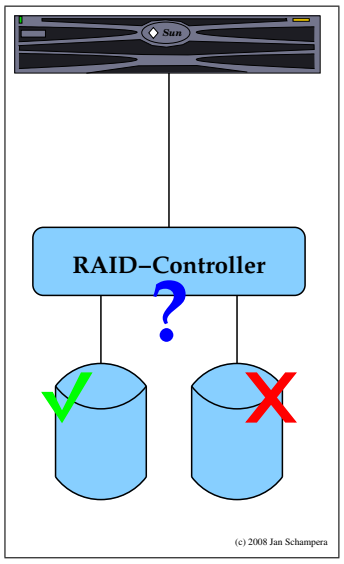

Abb. 4.1: *RAID-1 mit inkonsistenten Spiegelhälften*

Eine solche Verfälschung kann unter Umständen *überhaupt nicht* erkannt werden, da die prüfende Hardware selbst betroffen sein kann. Die Zerstörung der Daten ist hier im Normalfall die unausweichliche Folge.

4.5 Schadcode

Viele Computerviren verstecken sich innerhalb normaler ausführbarer Dateien, und kommen dort bei Ausführung der entsprechenden Programme zum Einsatz.

Dieser Befall findet in der Regel auf eine Weise statt, der die Funktion des Programms nicht beeinträchtigt. Manche Befallmechanismen wirken sich auch nicht auf die Größe der Datei im Dateisystem aus, da sie sich entweder in gut nutzbare Bereiche der Datei oder in ungenutzte Bereiche des Dateisystems selbst (Slack-Area-Viren) schreiben. Normalerweise wird beim Infizieren von Dateisystemen der Weg über die Dateisystemtreiber umgangen.

Eine gut durchgeführte Infektion kann nur durch regelmäßige Prüfungen mit Anti-Viren-Software oder durch Erstellung und Vergleich von Prüfsummen über alle Daten erkannt werden.

4.6 Ein Beispiel aus der Praxis

Ein mittlerer UNIX®-Mailserver mit ca. 600 Gigabytes an Nutzdaten stürzte aufgrund fehlerhafter und unsinniger Informationen in den Inode-Tabellen (Dateizuordnungstabellen) seines UFS[3]-Dateisystems ab. Nach einem Neustart dauerte die angestoßene

[3]UFS – UNIX®-filesystem; ein Dateisystemtyp

Dateisystemüberprüfung ca. 10-12 Stunden – ohne nennenswerten Erfolg. Im darunterliegenden RAID-1-System enthielten beide Spiegelhälften unterschiedliche Informationen, was zu weiteren Inkonsistenzen im Dateisystem und damit zu Abstürzen führte.

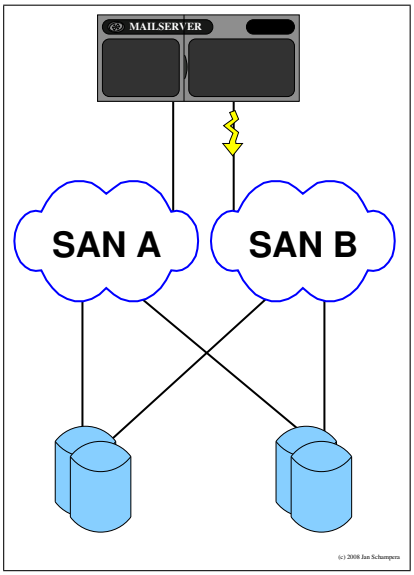

Abb. 4.2: *Beispiel aus der Praxis: UNIX®Mailserver mit defektem HBA*

Die Ursache für dieses zunächst unerklärliche Verhalten war ein defekter HBA (Fiber-Channel-Adapter), der bei der Übertragung von Daten ins SAN Fehler in den Datenstrom injizierte, je nach momentanem Zugriffspfad im Speichernetzwerk, entweder auf den einen oder auf den anderen RAID-Spiegel (Abb. 4.2).

5 Eine Lösung: ZFS

Um den genannten Problemen für die Datensicherheit zu begegnen, wurde durch ein Entwicklerteam bei Sun Microsystens Inc. das *ZFS* ins Leben gerufen. Die Fähigkeiten dieses „Z-Dateisystems" gehen weit über die reine Zusatzabsicherung der Daten hinaus, allerdings werden hier natürlich nur die für die Datensicherheit relevanten Eigenschaften behandelt.

Die Bezeichnung *ZFS* stammt ursprünglich von *„Zettabyte Filesystem"*, was klarstellen sollte, dass eine unvorstellbar grosse Menge an Speicherplatz verwaltet werden kann[4], indem durchgehend 128bit-Adressierung verwendet wird. Mittlerweile wird aber nur noch „ZFS" oder „Z-Dateisystem" ohne besondere Bedeutung als reiner Name verwendet.

[4]Der Größenfaktor „zetta" ist der Nächste nach „exa", welcher in Binärsystemen auch die 64bit-Grenze beschreibt (DIN1301)

5.1 Grundaufbau

ZFS vereinigt die beiden historisch getrennten Funktionen des Dateisystems und des Volume-Managements in einem einzigen Organisationssystem. Hier bilden „Datei" und „Festplatte" sozusagen eine Einheit.

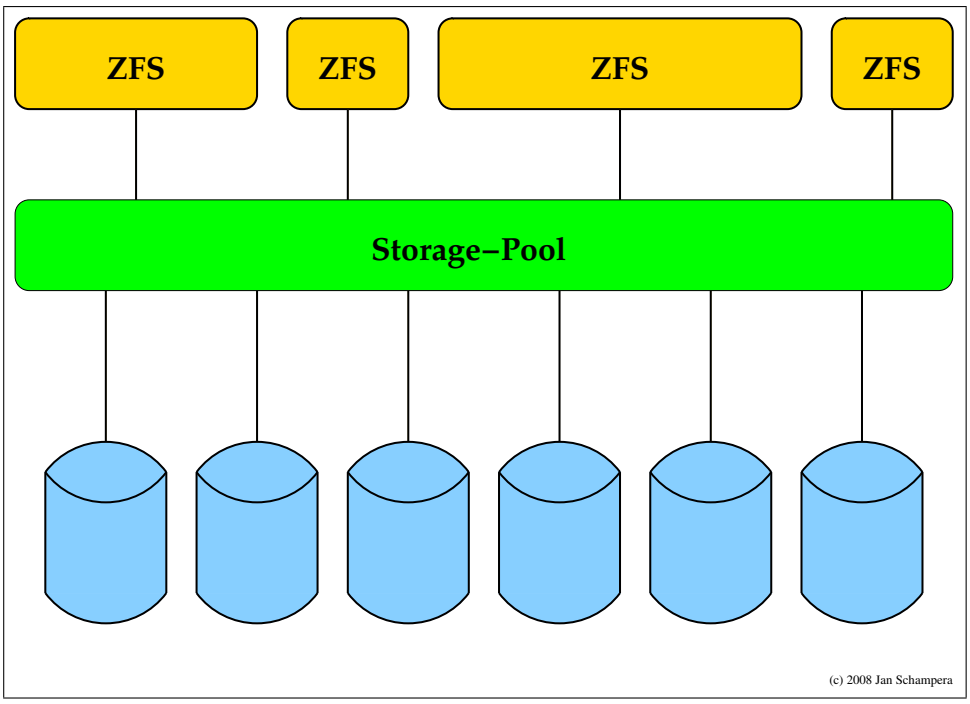

Abb. 5.1: *Grundaufbau des ZFS und seines Storage-Pools*

Alle zugewiesenen Massenspeicher werden in einem sogenannten *Storage-Pool* (Z-Pool) zusammengefasst und verwaltet. In diesem Pool wird auf Blockebene auch die Redundanz der Daten sichergestellt – zum Beispiel durch eine spezielle Methode, die vom Prinzip des RAID-5 abgeleitet ist. So wird sichergestellt, dass jeder Speicherblock im Z-Pool mehrfach abgesichert ist. Storage-Pools sind also für die Verwaltung und Absicherung der Speicherressourcen auf Hardware-Ebene verantwortlich.

Im nächsten Schritt bedienen sich ZFS-Dateisysteme bei Bedarf aus den von diesem Pool bereitgestellten Speicherblöcken zur Nutzung sowohl als Daten- und auch Metadatenblöcke. Es werden keine weiteren Schritte zur Absicherung nötig, denn ein vom Dateisystem genutzter Speicherblock ist bereits im Z-Pool abgesichert (Abb. 5.1).

Diese Zugriffstechnik kann man am Besten mit der RAM-Zuweisung an Applikationen vergleichen. Der RAM wird nicht partitioniert und in festen, großen unveränderlichen Stücken an die Applikation gegeben, sondern häppchenweise aus einem Gesamtpool genommen und nur bei Bedarf bereitgestellt.

Das ZFS selbst ist eine baumartige Struktur von Metadatenblöcken und Nutzdatenblöcken aus dem Z-Pool. Die Wurzel dieses Baumes bildet der sogenannte *Uberblock*[5]. Er

[5] vom dt. „über", ins Englische übernommen

enthält Zeiger auf weitere Organisationsblöcke, die ihrerseits auf Daten oder weitere Metadaten verzweigen (Abb. 5.2).

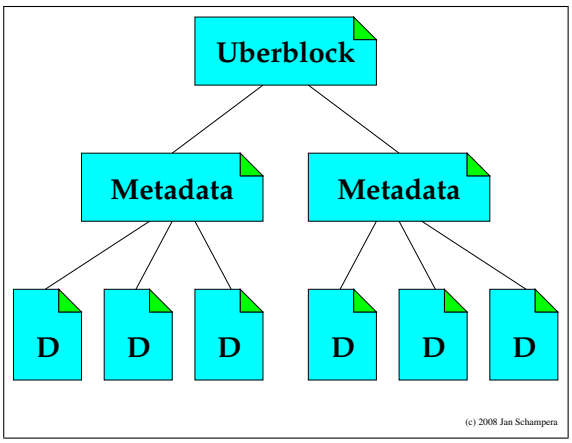

Abb. 5.2: *Datenblockorganisation im ZFS*

5.2 Absicherung der Daten

ZFS versucht, mit einer Reihe von Sicherheitsvorkehrungen und Features vielen Gefährdungspotentialen entgegenzuwirken, oder im Falle des Falles Schaden zu beheben oder zumindest zu begrenzen.

5.2.1 Transaktionen

Das Z-Dateisystem arbeitet, ähnlich wie jede gängige Datenbanksoftware, durchgehend *transaktionsorientiert*. Jede atomare Schreiboperation durch die Applikation im Dateisystem erzeugt dabei eine Kette von Schreibereignissen in ZFS und Z-Pool, die alle zu einer gesamten Transaktion zusammengefasst werden. Sollte diese Transaktion aus irgendeinem Grund nicht abgeschlossen werden, so ist der Zustand des Dateisystems, als hätte sie nie stattgefunden. Wird zum Schluß der *Uberblock* selbst überschrieben, so gilt die Transaktion als erfolgreich abgeschlossen.

Dieses Verfahren verhindert natürlich keine logische Verfälschung der Daten (z. B. falsche Datensätze weil die Anwendung nach dem Schreiben nicht geprüft hat), es ist aber sichergestellt, dass durch unterbrochene Schreiboperationen keine Gefahr für die Konsistenz des Dateisystems selbst entsteht. In jedem Fall muß logische Inkonsistenz durch die Applikationen selbst abgefangen werden.

5.2.2 Copy-on-Write

ZFS überschreibt niemals Datenblöcke im Datenträger-Pool. Bei Schreiboperationen werden stets neue Blöcke angefordert, nach Abschluß die alten freigegeben (Copy-on-Write – *CoW*) und natürlich der *Uberblock* angepasst. Dies sorgt unter Anderem dafür,

dass die „Originaldaten” noch allermindestens bis zum Abschluß der Schreibtransaktion vorhanden sind und als Sicherheit dienen (Abb. 5.3).

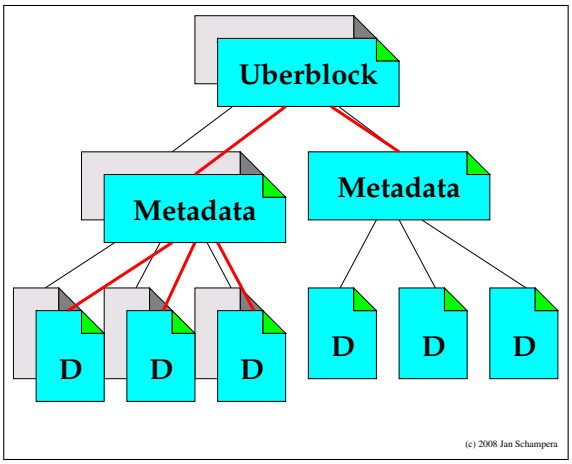

Abb. 5.3: *Funktionsprinzip Copy-on-Write (CoW)*

Diese Technik birgt auch einen kleinen Geschwindigkeitsvorteil: Daten werden zum Überschreiben nicht extra durch die Laufwerksmechanik angefahren, stattdessen werden einfach neue Blöcke vom Z-Pool angefordert, die sich in der Regel bei oder nahe der aktuellen Position der Schreib-/Leseköpfe der Festplatten befinden. Eine Folge von Schreiboperationen im Dateisystem ergibt dann auf der physikalischen Ebene im Idealfall einen *Sequential Write*, eine Sequenz von Schreiboperationen, bei denen die Köpfe nicht umpositioniert werden müssen.

5.2.3 Prüfsummen

Prüfsummen über die Datenblöcke, sowie alle Metadatenblöcke stellen zu jeder Zeit sicher, dass die Daten und deren Organisation in sich konsistent sind.

Diese Checksummen werden durch die gesamte Hierarchie durch gespeichert und bei einem Lesezugriff ausgewertet. Bei jeglichem Schreibzugriff werden die Prüfsummen natürlich für den jeweiligen Bereich neu berechnet.

Wenn man moderne Systemarchitekturen und Prozessoren zugrundelegt, ist der Mehraufwand zur Prüfsummenberechnung relativ gering und geht bei normaler Belastung vollkommen „im Grundrauschen” unter – ein geringer Preis für die durchgehende Absicherung.

5.3 Snapshots

Durch die vorher erwähnte Copy-on-Write-Technik wird eine weitere Eigenschaft des ZFS sehr simpel und effizient implementiert: *Snapshots*

Wenn ein Snapshot erzeugt wird, werden einfach alle Datenblöcke vor erneuter Verwendung geschützt, womit sie als „Originalkopie" nach dem Überschreiben mittels CoW noch vorhanden sind und als Snapshot-Stand dienen.

Blöcke die nicht geändert wurden stellen beides, aktuellen Stand und Snapshot-Stand dar. Snapshots sind sozusagen nur Abbilder von ZFS-Dateisystemen, die auf alten Metadaten und Nutzdaten basieren. Der „alte" *Uberblock* wird auch aufgehoben und dient als Einstiegspunkt in das „eingefrorene" Dateisystem.

Dies ermöglicht speichersparende Momentanzustände des Dateisystems aufzunehmen, um beschädigte oder unabsichtlich gelöschte Daten kurzfristig wiederherzustellen, ohne das Replay eines Backups einzuleiten. Man könnte zum Beispiel nach jedem Arbeitstag oder nach einem wichtigen Arbeitsschritt einen Snapshot erzeugen lassen, um in jedem Fall auf ältere Daten Zugriff zu haben.

5.4 Selbstheilung

Durch die Transaktionsorientierung und die Prüfsummen kann das ZFS *jederzeit* korrupte Datenblöcke erkennen und die Daten aus einer redundanten Quelle bereitstellen. Diese korrekten Daten dienen auch gleichzeitig als Vorlage zur Wiederherstellung des defekten Datenblocks.

Dieser Vorgang ist volltransparent für den Nutzer und erfordert keinerlei administrativen Eingriff. Der „schleichenden Zerstörung" der Daten wird somit wirksam entgegengetreten und etwaiger physikalischer Magnetisierungsverlust wird rechtzeitig erkannt.

6 Fazit

Der Bedarf an Massenspeicherplatz wird sich weiter, wie die letzten Jahrzehnte auch, etwa alle 10 Jahre verdoppeln. Der Bedarf an *sicheren* Speichermöglichkeiten wird sich ähnlich entwickeln. Der Aufbau und die Konzepte des Z-Dateisystems ist ein Schritt in die richtige Richtung, nicht zuletzt weil derzeit Verschlüsselungstechniken und weitere Sicherheitsmechanismen für ZFS entwickelt werden.

Abbildungsverzeichnis