

Inhaltsangabe

RAID-Z.....	1
1. Traditionelle RAID Systeme.....	1
1.1 Allgemeines.....	1
1.2 Hardware RAID und Software RAID.....	2
1.3 RAID Level.....	3
1.3.1 RAID 0.....	3
1.3.2 RAID 1.....	4
1.3.3 Die Kombinationen RAID 0 +1 und RAID 10.....	5
1.3.4 RAID 5.....	7
1.3.5 RAID 6.....	8
2. RAID-Z.....	9
2.1 ZFS.....	9
2.2 RAID-Z.....	11
2.3 Zusammenfassung: Vorteile von RAID-Z gegenüber herkömmlichen RAID Systemen.....	14
3. Quellenverzeichnis.....	17

RAID-Z

1. Traditionelle RAID Systeme

1.1 Allgemeines

RAID steht ursprünglich für “Redundant Array of Inexpensive Disks” - also frei übersetzt “redundante Anordnung billiger Festplatten”. Später änderte sich der Begriff in “Redundant Array of Independent Disks” - also “redundante Anordnung unabhängiger Festplatten”, weil es dann möglich war, eine einzelne Festplatte im Verbund im laufenden Betrieb auszuwechseln.

1987 führten D. A. Patterson, G. Gibson und R. H. Katz von der Universität Kalifornien in Berkeley den Begriff RAID ein.¹ Mit dem RAID System wurde es möglich, grosse, damals auch teure Platten, durch einen Verbund vieler kostengünstiger, kleine Platten zu ersetzen und diese als ein logisches Laufwerk zusammenzufassen. Das höhere Ausfallrisiko einer dieser Festplatten wurde durch redundante Speicherung der Daten verteilt auf den jeweiligen Festplatten des Verbundes begegnet.

Das RAID-System sollte vor allem den Vorteil bieten, kostengünstig schnellen, zuverlässigen und grossen Speicher bereitzustellen.²

- kostengünstig durch Einsatz von preiswerten Platten anstelle der grossen und teuren Platten
- zuverlässig durch redundante Speicherung der Daten bzw. Bilden der sogenannten Parity-Informationen (Checksummen) über die gespeicherten Daten
- schnell aufgrund des parallelen Zugriffs auf mehrere Platten des Verbunds

¹ <http://de.wikipedia.org/wiki/RAID>, http://www.raid.com/04_00.html, <http://www.baarf.com/>

² ebenda

Wenn man von Zuverlässigkeit spricht, dann kommt man auf den Begriff MTBF (Mean Time Between Failure - mittlere Betriebsdauer zwischen Ausfällen) zu sprechen.

MTBF ist also das Mass für die Zuverlässigkeit von Geräten, die instandgesetzt werden.³

Ein Standard- MTBF-Wert für Festplatten ist 500.000 Stunden = 57 Jahre.

Mit Hilfe des MTBF kann aber auch die stationäre Verfügbarkeit (A = Availability) eines Systems berechnet werden:

$$A = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}^4$$

Die Zuverlässigkeit von RAID Level lässt sich mit Hilfe mathematischer Formeln berechnen.⁵

Ebenso lässt sich mathematisch darstellen, wann der Verlust der Daten erfolgt.

Dazu werden neben dem MTBF noch folgende Werte benötigt, denen wir folgende Variablen zuordnen:⁶

F Mean time between failure (MTBF) – mittlere Betriebsdauer zwischen Ausfällen

R Mean time to repair (MTTR) - mittlere Zeit zur Wiederherstellung

N Number of data disks – Anzahl der Festplatten des RAIDs

L Mean time to data loss (MTDL) – mittlere Zeit bis zum Datenverlust (Lifetime)

Im folgenden wird der MTDL = L für jedes RAID Level berechnet.

Bei einer einzelnen Festplatte gilt :

Die Platte fällt aus.

$$L = F$$

1.2 Hardware RAID und Software RAID

Man unterscheidet zwischen Hardware RAID und Software RAID.

Ein Hardware RAID Controller übernimmt die Organisation der Festplatten beim Hardware RAID.

Typischerweise befindet sich der Hardware RAID Controller im Disk Array. Sollten sich die Platten im Server befinden oder direkt an den Server angeschlossen sein, so befindet sich der Hardware RAID Controller im Server als zusätzliche Karte oder auch direkt integriert auf der Hauptplatine. Im letzten Fall beschränkt sich der RAID Solaris Support bisher auf RAID 0 oder RAID 1.⁷

³ Siehe: http://de.wikipedia.org/wiki/Mean_Time_Between_Failures

⁴ ebd.

⁵ Siehe auch: <http://www.digit-life.com/articles/storagereliability/>

⁶ Variablen und Formeln sind Peter's Solaris Zone entnommen: <http://www.petertribble.co.uk/Solaris/>, <http://www.petertribble.co.uk/Solaris/raid.html>

⁷ So zum Beispiel in der X4500; Beschreibung der RAID Level folgt im Abschnitt 1.3 RAID Level

Beim Software RAID kontrolliert eine in das Betriebssystem integrierte spezielle Software die Organisation der Festplatten. Die Festplatten können sich sowohl im Rechner selbst befinden als auch als externer Storage (disk array) direkt am Rechner angeschlossen sein. Letzteres wird oft als JBOD (just a bunch of disks) bezeichnet.

Der Vorteil des Hardware RAIDs gegenüber dem Software RAID gründet sich vor allem darauf, dass die Hardware RAID Controller mit einem batterie-gepufferten Cache ausgestattet sind, so dass im Falle eines Stromausfalls der Schreibvorgang noch vollständig ausgeführt werden kann und somit Datenverlust vermieden wird.

Umgekehrt kann aber der Hardware RAID Controller durch einen Defekt auch die Daten korrumpieren. In diesem Fall spricht man von silent data corruption (schleichende Datenkorruption), da dies solange unbemerkt bleibt, bis man auf die korrumpierten Daten zugreift und einen Fehler erhält.

1.3 RAID Level

In diesem Abschnitt wird nur auf die gängigsten RAID Level eingegangen.

Diese sind RAID 0, RAID 1, RAID 5 und RAID 6 und die Kombination RAID 0+1 bzw 10.⁸

1.3.1 RAID 0

RAID 0 bezeichnet das "Striping", welches man sinngemäss mit "in Streifen anordnen" übersetzen kann.

Beim Striping werden die logischen Laufwerke oder auch logische Volumes in Blöcke gleicher Grösse sequentiell über die physikalischen Laufwerke hinweg angeordnet. Abbildung 1 verdeutlicht das.

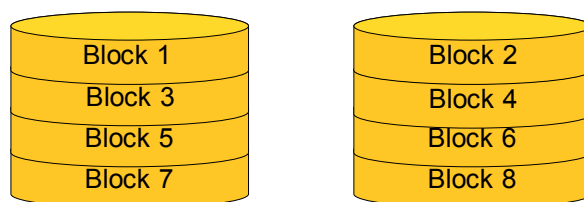


Abb. 1
Raid 0 Striping ueber 2 Festplatten

Das logische Laufwerk oder auch logische Volume setzt sich aus den Blöcken 1-8 zusammen, welche sich auf zwei physikalischen Platten wie oben dargestellt verteilen.

Selbstverständlich ist es möglich ein Striping über mehr als 2 Platten zu organisieren.

Genau genommen stellt RAID 0 kein richtiges RAID dar, da bei diesem Verfahren keine Redundanz der Daten erzielt wird und der Ausfall einer einzigen Festplatte des Arrays auch den Ausfall der Datenverfügbarkeit bewirkt.

RAID 0 beschleunigt den Schreib- und Lesezugriff auf die Daten, da beim Zugriff auf die Daten nicht

⁸ Eine komplette Auflistung der RAID Level ist beispielsweise hier: <http://de.wikipedia.org/wiki/RAID>, Abschnitt 3 und 4

sequentiell auf die Blöcke einer Festplatte zugegriffen wird, sondern stets auf den Block der nächsten Festplatte zugegriffen wird, so dass beim Schreiben und Lesen der Schreib-/Lesekopf der nächsten Festplatte bereits positioniert werden kann und somit parallel geschrieben bzw. gelesen werden kann.



Striping verbessert also die Schreib- und Lesegeschwindigkeit, nicht aber die Ausfallsicherheit

Auch hier die Formel zur Berechnung der Zuverlässigkeit. Die Lebenszeit reduziert sich mit der Anzahl der Platten, denn der Verlust der Daten passiert, sobald eine der Platten ausfällt. Daher gilt:

$$L = F/N$$

1.3.2 RAID 1

Mit RAID 1 wird das "Mirroring" - Spiegelung der Daten – beschrieben.

Es werden mindestens 2 physikalische Festplatten benötigt.

Die Daten der einen Festplatte werden auf eine zweite Festplatte 1:1 gespiegelt. Beide Platten stellen also je eine Spiegelhälfte ("submirror") dar und bilden zusammen den Spiegel ("mirror").

Abbildung 2 verdeutlicht: 1 logisches Laufwerk oder logisches Volume setzt sich zusammen aus den Blöcken 1- 4 auf der ersten Platte (1. Spiegelhälfte) und dieses wird 1:1 auf die zweite Platte (2. Spiegelhälfte) gespiegelt.

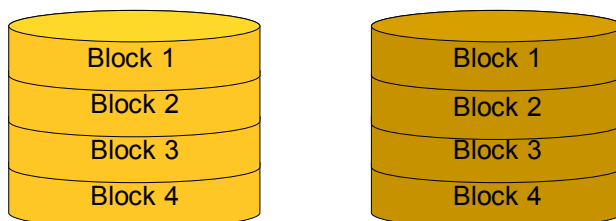


Abb. 2
Raid 1 Spiegelung 2er Festplatten

Bei diesem Verfahren werden die Daten stets auf beide Spiegelhälften geschrieben und liegen demnach redundant vor.

Nachteile für die Schreibgeschwindigkeit entstehen beim Hardware RAID 1 nicht, jedoch beim Software RAID 1.

Die Lesegeschwindigkeit aber wird deutlich verbessert, da von der einen oder der anderen Spiegelhälfte gelesen wird, je nachdem welche gerade besser positioniert ist.

Es kann eine Platte durch einen Defekt ausfallen. In dem Fall werden die Daten auf der verbliebenen Spiegelhälfte geschrieben bzw. von ihr gelesen. In letztem Falle verringert sich allerdings dann die Lesegeschwindigkeit.

Ist die zweite defekte Spiegelhälfte durch eine neue Festplatte ersetzt worden, so wird die neue Spiegelhälfte mit der verbliebenen Spiegelhälfte, meistens manuell angestoßen, synchronisiert.



Übrigens, Spiegelung ersetzt nicht die Datensicherung! Eine fehlerhafte Schreiboperation, bspw. verursacht durch Stromausfall, wirkt sich sofort auf den gesamten Spiegel aus, da die geänderten Daten sofort auf beide Spiegelhälften geschrieben werden, so dass beide Spiegelhälften zwar synchronisiert sind aber korrupte oder inkonsistente Daten enthalten. Zur Wiederherstellung der Daten benötigt man dann die Datensicherung.

Schliesslich, auch hier wieder die Formel für die Zuverlässigkeit:

$$L = (F/2)/(R/F) = F \cdot F / 2R$$

Der Wert berechnet sich aus der erwarteten Lebenszeit einer Festplatte geteilt durch 2 multipliziert mit der Wahrscheinlichkeit, dass die zweite Spiegelhälfte ebenfalls ausfallen wird, bevor die erste Spiegelhälfte ersetzt wurde.

1.3.3 Die Kombinationen RAID 0 + 1 und RAID 10

Die Kombinationen von RAID 0 und 1 bzw. RAID 10 werden sehr häufig verwendet. Die Verfahren kombinieren die Vorteile eines RAID 0 (Verbesserung der Schreib- und Lesegeschwindigkeit) mit denen des RAID 1 (Ausfallsicherheit und Verfügbarkeit durch Spiegelung zweier Laufwerke).

RAID 0 + 1

Die Abbildung 3 zeigt, wie die Festplatten bei RAID 0 + 1 organisiert werden.

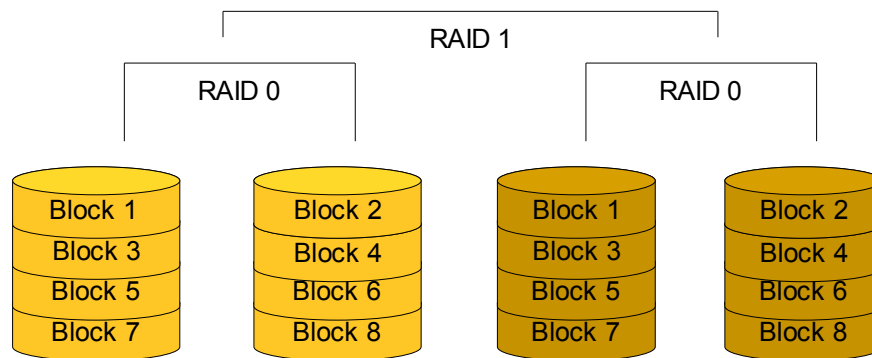


Abb. 3
Raid 0+1 Spiegelung 2er
Stripesets

Zunächst werden 2 oder mehr Platten zu einem logischen Laufwerk zusammengefasst, wobei die Festplatten in gleichgrosse Streifen aufgeteilt werden und ein logisches Laufwerk bilden. Ein zweites logisches Laufwerk wird mit 2 oder mehr Festplatten erstellt und bildet ebenfalls ein logisches Laufwerk.

Nun werden diese beiden logischen Laufwerke mit RAID 1 gespiegelt, so dass jedes RAID 0 Set eine Spiegelhälfte darstellt.

Und schliesslich wieder die Formel für alle Interessierten:

$$L = (F/2N)/(R/F) = F * F / (2N * R)$$

2N ist gleich die Anzahl der Festplatten x 2, so dass die Zeit für den Ausfall F/2N ist. R/F bezeichnet die Wahrscheinlichkeit, wann ein Ausfall der Daten geschieht.

RAID 10

In dieser Kombination wird erst gespiegelt und dann "gestriped". Der Unterschied zu RAID 0+1 wird deutlich, wenn mehr als insgesamt 4 Platten verwendet werden.

Angenommen es stehen 6 Platten zur Verfügung.

Bei RAID 0+1 würden 3 Platten als logisches Laufwerk ein RAID 0 Array bilden. 2 x 3 Platten bilden so die beiden Spiegelhälften (siehe Abbildung 4).

Im RAID 10 Verfahren würden 2 Platten zunächst gespiegelt werden, daraus resultieren 3 logische Laufwerke, die dann "gestriped" werden, also 3x2 (siehe Abbildung 5).

Damit verändert sich die Ausfallrate erheblich.

Im ersten Fall RAID 0+1, 2x3 Platten kann genau eine Platte ausfallen. Es bleibt eine intakte Spiegelhälfte. Das bedeutet, die Hälfte der eingesetzten Platten (hier 3) sind unbrauchbar. Ein Ausfall einer weiteren Platte in der verbliebenen Spiegelhälfte während der Wiederherstellung führt zum totalen Datenverlust, die Wahrscheinlichkeit, dass das passiert, ist hier 1:5; aber die Wahrscheinlichkeit, dass es die intakte Spiegelhälfte trifft ist schon 1:2.

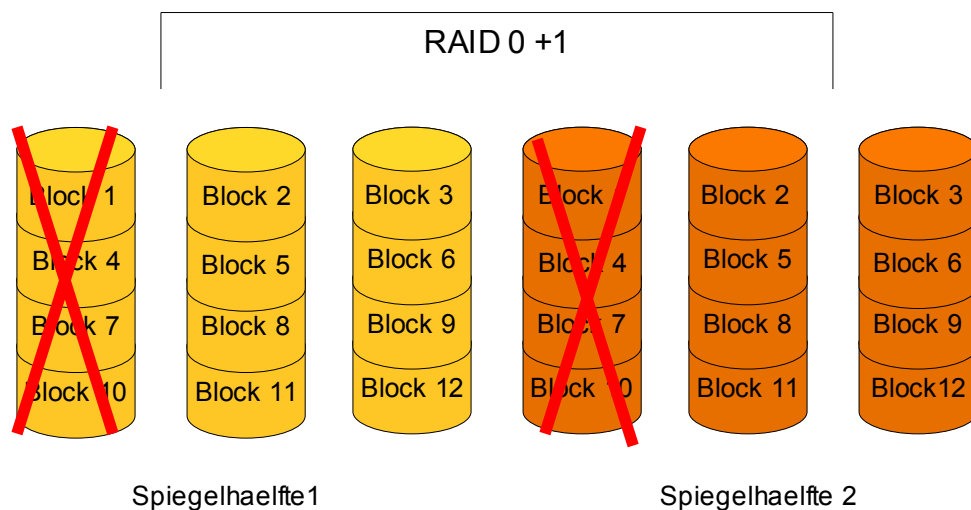


Abb. 4
RAID 0 + 1

Im zweiten Szenario, RAID 10, könnten 2 Platten gleichzeitig ausfallen. Entweder wäre eine komplette Spiegelhälfte von insgesamt drei unbrauchbar, oder aber 2 Spiegelhälften wären ausgefallen; eine intakte verbleibt aber in jedem Fall.

Geht man aber von dem Fall aus, es wäre zunächst nur eine Platte defekt, verbleiben 2 intakte Spiegelhälften. Fällt nun eine zweite Platte aus - die Wahrscheinlichkeit beträgt 1:5 - so ist die Wahrscheinlichkeit, dass es eine Platte eines intakten Spiegels betrifft immer noch 1:3. Es bleibt aber in jedem Fall mindestens eine Spiegelhälfte intakt. Erst wenn eine dritte Platte während der Wiederherstellungszeit ausfällt - die Wahrscheinlichkeit beträgt 1:4, erhöht sich die Wahrscheinlichkeit, dass es den intakten Spiegel trifft um 50 %, nämlich 1:2.

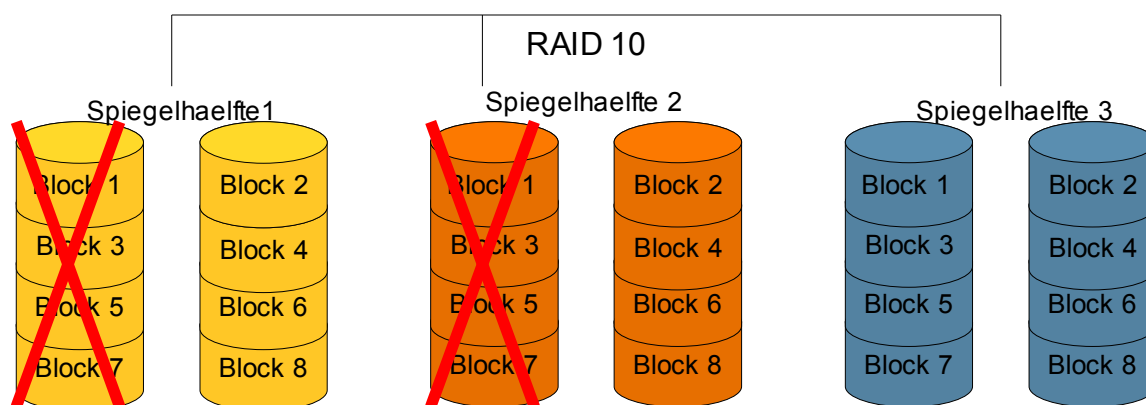


Abb. 5
RAID 10



Fazit: beim RAID 10 ist ein Datenverlust unwahrscheinlicher, als bei RAID 0 + 1, und das Zeitfenster zur Wiederherstellung ist grösser.

1.3.4 RAID 5

RAID 5 erhöht die Zuverlässigkeit der Daten und die Schreib-/Lesegeschwindigkeit durch parallele Zugriffe.

Die Redundanz der gespeicherten Nutzdaten wird dadurch gewährleistet, dass Paritätsinformationen (Parity) ebenfalls verteilt über die Platten des RAID 5 abgelegt werden.

Die Parität erlaubt den Ausfall einer Festplatte innerhalb des RAID 5 Verbundes.

Es werden mindestens 3 physikalische Festplatten benötigt.

Abbildung 6 stellt das Verfahren dar:

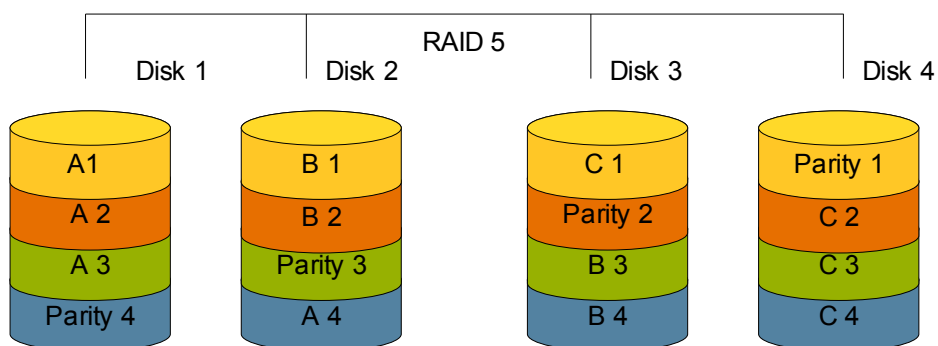


Abb. 6
RAID 5

Wie bei RAID 0 bilden gleichgrosse, sequentielle Streifen über die Festplatten ein logisches Laufwerk oder auch ein logisches Volume. In der Abbildung bilden die Streifen A1, B1, C1 und Parity 1 der Disk 1, Disk 2 und der Disk 3 das 1. logische Laufwerk (gelb). Ebenso bilden A2, B2, C2 und Parity 2 (orange), und A3, B3, C3 und Parity 3 (grün) und die A4, B4, C4 und Parity 4 (blau) eigene logische Laufwerke 2, 3 und 4. Die Paritätsinformationen werden in dem Parityblock 1 für das 1. logische Laufwerk abgelegt, für das 2. logische Laufwerk im Parityblock 2 und so weiter.

Die Parität wird mittels einer logischen XOR Operation berechnet. So lautet die Operation für Parity 1 $A1 \text{ XOR } A2 \text{ XOR } A3$ und für Parity 2 $A2 \text{ XOR } B2 \text{ XOR } C2$ und so weiter.

Durch die Parität wird die Gesamtkapazität für die Nutzdaten auf (Anzahl der Platten $N - 1$) verringert. Angenommen die Festplatten Disk 1 bis Disk 4 besitzen eine Gesamtspeichergrossse von 1 TB. Im obigen Beispiel wird nun jede Platte in 4 gleich grosse Streifen zu je 250 GB aufgeteilt (vertikal), die dann die logischen Laufwerke bilden (horizontal nach Farben sortiert). Die logischen Laufwerke 1-4, zusammengesetzt aus den A1, B1 und C1, sowie Parity 1 bzw. den Streifen A2, B2, C2 und Parity 2 usw. besitzen dann wieder eine Gesamtkapazität von 1 TB, nämlich $4 \times 250 \text{ GB}$.

Für die Nutzdaten können aber nur 3 Blockeinheiten verwendet werden, da ein Streifen, nämlich die Parity, für die Speicherung der Paritätsinformationen genutzt wird. Daraus ergibt sich eine Speicherkapazität für die Nutzdaten von:

$(N \text{ Festplatten} - 1) \times \text{Blockgrösse} = \text{Speicherkapazität der Nutzdaten}$

Also im obigen Beispiel:

$(4 - 1) \times 250 \text{ GB} = 750 \text{ GB}$

Dies gilt natürlich für alle logischen Laufwerke im obigen Beispiel und daher gilt die Formel:

$N \text{ Festplatten} - 1 = \text{Gesamtkapazität für Nutzdaten}$

Bei einem Ausfall höchstens einer Platte ist zwar die Datensicherheit gewährleistet, jedoch lässt die Schreib- und Lesegeschwindigkeit während der Rekonstruktion der Daten auf eine Hot Spare Platte bzw. auf eine neue Platte deutlich nach. Der Rekonstruktionsprozess kann je nach Grösse der logischen Laufwerke lange dauern.

1.3.5 RAID 6

Das RAID 6 Verfahren ist dem RAID 5 Verfahren sehr ähnlich. Es werden bei RAID 6 2

Paritätsinformationen (P und Q siehe Abbildung 7) pro logisches Laufwerk berechnet und so über die Platten blockweise verteilt, dass 2 Platten ausfallen können.

RAID 6 benötigt mindestens 4 Platten.

Die Gesamtkapazität des Plattenspeichers berechnet sich dementsprechend: $n + 2$, wobei n die Anzahl der Platten für die Nutzdaten darstellt plus 2 Platten für die Paritäten.

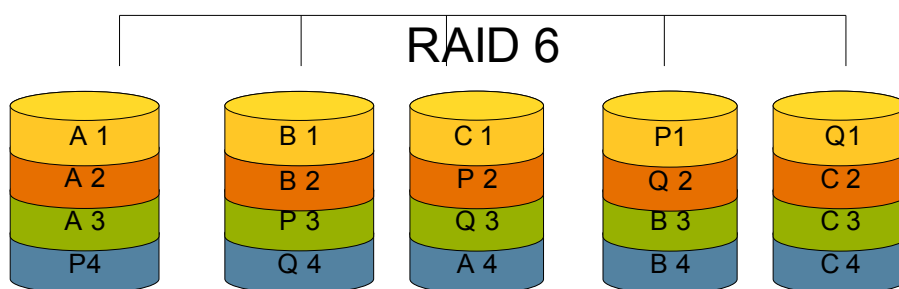


Abb. 7
RAID 6

2. RAID-Z

2.1 ZFS

Mit Solaris 10 06/06 wurde das Dateisystem ZFS offiziell als Teil des Betriebssystems herausgegeben.

Wie in den vorangegangenen Kapiteln beschrieben bieten die üblichen Dateisysteme keinen Schutz gegen schleichende Datenkorruption (silent data corruption). Jeder Defekt auf dem Datenpfad kann die Daten unbemerkt korrumpieren.

Die Implementierung und auch das Management von logischen Volumes gestaltet sich aufwendig und umständlich (Erstellen von Labels, Partitionen und logischen Volumes; Vergrößern/Verkleinern von Volumes, Verwalten von `/etc/vfstab` und `/etc/dfs/dfstab` etc.) und sie unterliegen einigen Begrenzungen (festgelegte Blockgröße, Dateisystemgröße, Anzahl Verzeichnis- und Dateieinträge, Dateigröße, Anzahl Snapshots etc.). Ausserdem gibt es keine Möglichkeit Dateisysteme zwischen Plattformen zu portieren.

Performanceeinbußen sind durch das Schreiben auf den nächsten freien Block (random-writes), feste Blockgrößen, Feststellen und Loggen nicht geschriebener Daten (dirty region logging bzw. journaling), und ineffizientes Prefetching verursacht.

Mit ZFS sollten diese Schwierigkeiten behoben werden.

Daher wurde ZFS ein völlig neues Design zugrundegelegt und mit neuen Funktionalitäten entwickelt.

ZFS ist ein 128-bit Dateisystem und setzt nach dem heutigen Stand keine Limits.

Der vorhandene Datenspeicher wird als Pool allokiert (zpool). Ein zpool besteht aus sogenannten vdevs (virtual devices). Vdevs können einzelne Festplatten, Spiegel, RAID Gruppen, Block devices und sogar Dateien sein. Das zugeordnete ZFS Dateisystem bedient sich aus dem zpool.

Ein Volume Manager wird nicht mehr benötigt, da das Filesystem nicht einem physikalischen Device zugeordnet ist, sondern sich den Plattenspeicher des Pools mit allen anderen Filesystem teilt. Wird dem Pool zusätzlicher Plattenspeicher hinzugefügt, so steht dieser ZFS sofort zur Verfügung.

Auf eine Art ähnelt der ZFS Storage Pool dem virtuellen Memory-System, denn fügt man diesem zusätzliche DIMMs hinzu, nutzen die Prozesse diesen auch sofort, ohne dass der Administrator das System konfigurieren müsste.

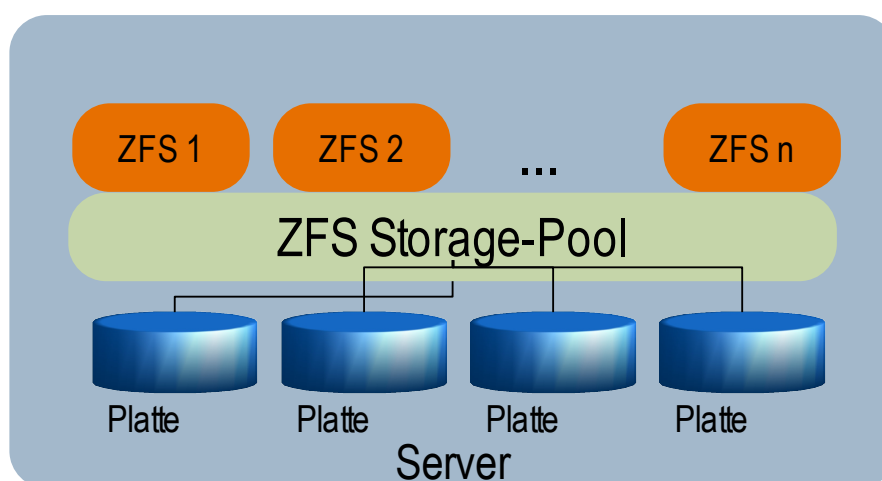


Abb. 8 ZFS Storage Pool

Die Datenintegrität wird gewährleistet mit Hilfe von 256-bit Prüfsummen, die über alle Daten, User-Daten UND Metadaten, gebildet werden. Diese werden hierarchisch in übergeordnete Metadatenblöcke abgelegt und enden im allen übergeordneten Überblock (siehe Abbildung 9).

ZFS unterstützt Pools, die verschiedenen Datenredundanzen: Spiegel (mirror), RAID-Z (entspricht RAID5) und seit Solaris 10 11/06 auch RAID-Z2 (entspricht RAID6).

Zusätzlich bietet ZFS einen selbst-heilenden Mechanismus (self-healing).

ZFS erkennt Datenkorruption und kann die defekten Daten reparieren, indem es auf eine intakte Kopie der Daten zurückgreift und die defekten Daten durch die intakten Daten ersetzt.

ZFS ist ein transaktionelles Dateisystem.

Wie oben festgestellt wurde, werden Daten bei den traditionellen RAID Systemen live modifiziert (read-modify-write). Bei einem Stromausfall während dieses Vorgangs kann es zu Dateninkonsistenzen kommen. Um die Daten wieder konsistent herzustellen, musste man bisher ein "fsck" ausführen. Mit ZFS werden die modifizierten Daten stets im Copy-on-write-Verfahren geschrieben (COW). Das heisst, eine COW Transaktion ist entweder komplett beendet oder aber sie wird ignoriert, beispielsweise im Falle eines Stromausfalls während des Vorgangs. Somit ist die Datenintegrität stets gegeben⁹.

⁹ Eine detaillierte Beschreibung gibt auch Constantin Gonzalez in seinem Whitepaper zu ZFS http://blogs.sun.com/constantin/entry/new_zfs_white_paper_in

Abbildung 9 verdeutlicht das.

Die Originaldaten bleiben erhalten (grau). Zum Schreiben der neuen Daten werden neue Blöcke allokiert, die Daten geschrieben, neue Metadatenblöcke und schliesslich ein neuer Überblock erzeugt (orange).

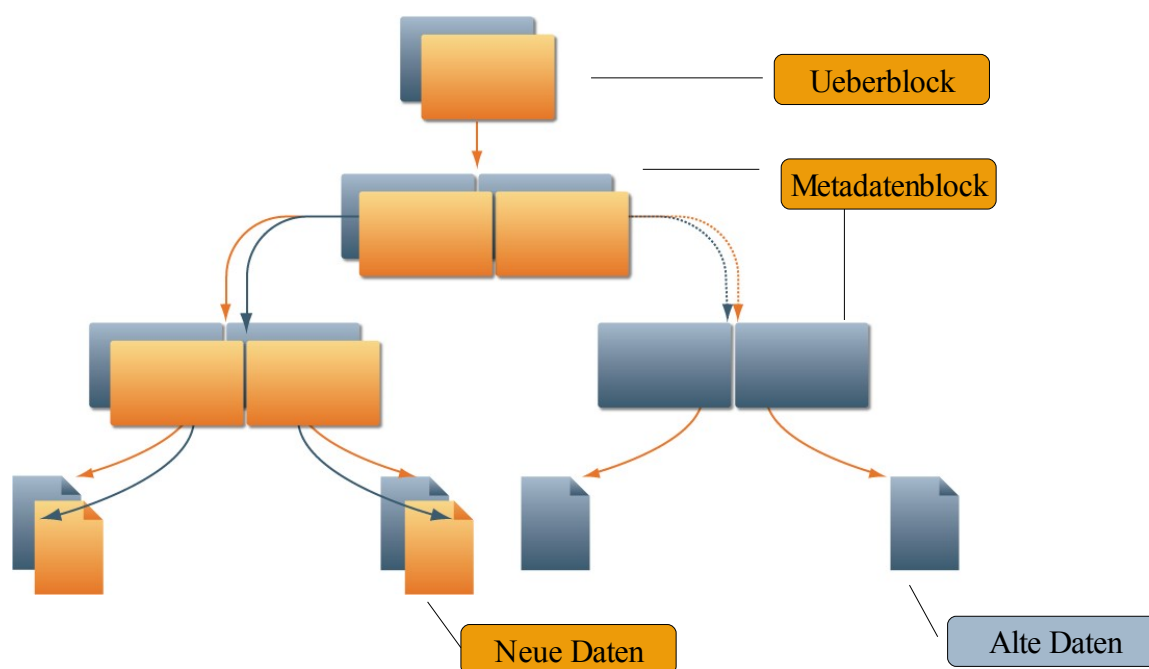


Abb. 9 Copy-on-Write Transaktion

2.2 RAID-Z

RAID-Z ist ein vdev eines ZFS Storage Pools (zpool). RAID-Z vereint die Datenredundanz eines RAID5 mit der Datenintegrität von ZFS¹⁰.

Beim Einrichten des zpools entscheidet man ob man seinen zpool als Spiegel (mirror), als RAID-Z Gruppe (entspricht RAID5) oder -seit Solaris 10 11/06 – als RAID-Z2 (entspricht RAID6 = double Parity) anlegt.



¹⁰http://competitive.uk.sun.com:9364/competitive/index.php/Sun/_Operating_Systems/_Solaris/_ZFS,
<http://www.opensolaris.org/os/community/zfs/>, <http://blogs.sun.com/bonwick/category/ZFS>,
<http://blogs.sun.com/roch/category/ZFS>, http://www.sun.com/bigadmin/features/articles/zfs_part2_ease.html,
<http://www.opensolaris.org/os/community/zfs/demos/>, http://www.sun.com/software/solaris/zfs_learning_center.jsp

Die Empfehlung, wann man besser einen Spiegel oder eine RAID-Gruppe anlegt, ist diese:¹¹

Das Einrichten von Spiegeln verbraucht mehr Plattenkapazitäten, ist aber bei vielen kleinen Blöcken für die Lesegeschwindigkeit von Vorteil.

Die RAID-Z-Konfiguration maximiert die Speicherkapazität. RAID-Z eignet sich gut beim Lesen und Schreiben grosser Datenblöcke (128k und grösser).

RAID-Z2 bietet die höchste Verfügbarkeit und beste MTDL (Mean Time To Data Loss) bei ungefähr gleicher Performance wie RAID-Z.

Die Wahl richtet sich also danach, ob man schnelle Lese- und Schreibgeschwindigkeiten oder hohe Datenverfügbarkeit erreichen möchte.

Eine RAID-Z Konfiguration entspricht zunächst einer RAID5 Gruppe, d.h. eine RAID-Z Gruppe besteht aus einer Anzahl Platten N der Grösse X, einer Parität P und damit ergibt sich eine Speicherkapazität von $(N-P) \times X$.

Es kann eine Festplatte ausfallen ohne Datenverlust.

RAID-Z2 arbeitet mit der sogenannten "double parity", das heisst eine RAID-Z2 Gruppe besteht aus einer Anzahl Platten N der Grösse X- 2 Platten (2P) für die Parität. Daraus ergibt sich auch die Speicherkapazität $(N-2P) \times X$.

Es können also 2 Platten ausfallen und die Datenverfügbarkeit ist noch gewährleistet.

RAID-Z verwendet dynamische Streifengrössen zwischen 512 Bytes und 128 Kilobytes. Jeder logische Block ist sein eigener Streifen. Auf diese Weise sind alle Schreibvorgänge full-strip-writes, das verbessert auch die Performance.

Abbildung 10 verdeutlicht das:

Die Y-Achse stellt die LBAs (logical block address = Festplattensektoren) 0 – 12 dar, die X-Achse die Festplatten A – E.

Jeder LBA A0, B0 etc. (also jedes Kästchen in der Abbildung 10) hat eine Grösse von 512 Bytes.

Wenn nun Daten geschrieben werden, dann werden die Daten auf genau die Anzahl von LBAs, die die Daten benötigen, und über die Anzahl Platten verteilt, die nötig sind um die Datenintegrität zu gewährleisten. Somit sind die logischen Blockgrössen und damit auch die Streifengrössen variabel und dynamisch angepasst (das heisst ein Streifen verteilt sich nicht stets auf alle Platten der RAID-Z Gruppe).

Das erste Beispiel in der Abbildung 10 in der Farbe orange (die ersten beiden Zeilen) zeigt Daten, die 8 physikalische Blöcke (D0 – D7) plus 2 Paritäten (P0 und P1) belegen.

In diesem Beispiel ist die Datenmenge so gross, dass 2 LBAs pro Platte benötigt werden, sie bilden einen sogenannten chunk (A0,A1 und B0,B1 etc.) und ausserdem über alle 4 Platten plus die Parität (Platte A) reicht, das ist der Streifen (stripe). Die chunks und die Verteilung über die Platten bilden einen logischen Dateisystem-Block bzw. einen Streifen.

Die Parität wird übrigens, wie man sieht, stets am Anfang der Datenblöcke geschrieben.

¹¹ <http://blogs.sun.com/roch/category/ZFS>, http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide, <http://blogs.sun.com/relling/category/ZFS>

LBA	Disk				
	A	B	C	D	E
0	P ₀	D ₀	D ₂	D ₄	D ₆
1	P ₁	D ₁	D ₃	D ₅	D ₇
2	P ₀	D ₀	D ₁	D ₂	P ₀
3	D ₀	D ₁	D ₂	P ₀	D ₀
4	P ₀	D ₀	D ₄	D ₈	D ₁₁
5	P ₁	D ₁	D ₅	D ₉	D ₁₂
6	P ₂	D ₂	D ₆	D ₁₀	D ₁₃
7	P ₃	D ₃	D ₇	P ₀	D ₀
8	D ₁	D ₂	D ₃	X	P ₀
9	D ₀	D ₁	X	P ₀	D ₀
10	D ₃	D ₆	D ₉	P ₁	D ₁
11	D ₄	D ₇	D ₁₀	P ₂	D ₂
12	D ₅	D ₈	•	•	•

Abb.10 RAID-Z¹²

Wenn wir uns das zweite Beispiel anschauen (gelb; Kästchen A2- D2), dann ist die zu schreibende Datenmenge kleiner, denn es werden nur 3 LBAs für die Daten (Kästchen B2 – D2 =logischer Block) plus 1 Paritätsblock (Kästchen A2) benötigt . Jeder physikalische Block ist für sich ein chunk, der Streifen ist 4 LBAs gross und reicht nur über 4 der 5 Festplatten der RAID-Z-Gruppe. Der logische Block setzt sich also aus 3 LBAs verteilt auf 3 Platten zusammen plus 1 Parität, insgesamt 4 Platten.

Das reicht ja auch, denn die Daten sind über 3 physikalische Platten verteilt, die Parität liegt auf einer anderen Festplatte und eine Platte kann in dieser Aufteilung ausfallen.

Schaut man sich den nächsten logischen Block an (grün, Kästchen E2 plus A3, B3 und C3), dann verhält es sich hier genauso wie im vorherigen Beispiel.

Nun schaut man aber auf die nächsten Kästchen D3 und E3 (rot), dann wird deutlich, dass ein RAID-Z-Streifen aus nur 2 Platten besteht und auch redundant ist ! Eine Platte kann ausfallen und die Daten können wiederhergestellt werden.

12 Abbildung ist aus http://www.opensolaris.org/os/community/zfs/docs/zfs_last.pdf, slide13

Im darauffolgenden Beispiel ist dargestellt, dass die Datenmenge in “chunks” aus 4 physikalischen Blöcken und aus 3 physikalischen Blöcken geschrieben werden kann. Man könnte sagen, der logische Block besteht aus 4 Streifen (Reihe 4 -7) unterschiedlicher Grösse, tatsächlich stellt dieser logische Block aber ein RAID-Z Streifen dar und ist ein full-stripe-write.

Im nächsten Beispiel (dunkellila, Kästchen D7, E7, A8, B8, C8 und D8) ist das Kästchen D8 mit einem X markiert. Das gleiche finden wir auch im nächsten Beispiel (hellblau, Kästchen E8, A9, B9 und C9 mit einem X markiert). Der Grund dafür sind sogenannte “alignments filler”. Diese sind code-spezifisch.¹³

2.3 Zusammenfassung: Vorteile von RAID-Z gegenüber herkömmlichen RAID Systemen

Die traditionellen RAID Systeme wurden eingeführt, um kostengünstig schnellen und zuverlässigen Datenspeicher bereitzustellen.

Das Hauptaugenmerk sollte dabei auf kostengünstig liegen. Diese Zeiten sind vorbei. RAID Systeme sind heute nicht mehr so kostengünstig wenn sie zuverlässig sein sollen, meint Jeff Bonwick auch in seinem Blog – Eintrag.¹⁴

RAID-5 Systeme, so schreibt Jeff Bonwick in seinem Blog Eintrag¹⁵, konnten ihre Versprechen nicht halten, wegen des sogenannten “write-holes”.

Das “write-hole” tritt auf, wenn nach dem Schreiben geänderter Daten, aber vor dem Berechnen der Parität ein Stromausfall geschieht.

In diesem Falle sind die Daten und die korrespondierende Parität ab diesem Zeitpunkt inkonsistent.

Die Daten können nicht wiederhergestellt werden.

Um genau dieses Risiko des “write-holes” zu vermeiden, werden bei Hardware RAIDs die geänderten Daten im batterie-gepufferten NVRAM gehalten, bis sie – auch nach einem Stromausfall - vollständig geschrieben werden konnten. Nur, NVRAM ist teuer!

Ein weiteres Problem der Hardware und Software RAIDs sind die Performanceprobleme, die sich mit den Schreiboperationen ergeben.

Wenn weniger Datenänderungen geschrieben werden müssen als das RAID-Stripe gross ist (partial-stripe writes im Gegensatz zu full-stripe writes), so müssen zunächst die alten Daten und die Parität synchron gelesen, dann geändert werden, bevor die neuen Daten geschrieben werden können (read-modify-write Operation). Die Originaldaten werden also direkt in drei Operationsschritten überschrieben. Das dauert und der Vorgang erzeugt Last.

Nun versucht man die Verzögerung des Schreibvorgangs vor dem Benutzer zu verbergen, weil die zu schreibenden Daten im NVRAM zwischengespeichert werden, und für den Benutzer der Schreibvorgang aus seiner Sicht damit beendet ist.

Das gilt aber nur für die Hardware RAID Systeme, ein Software RAID hat kein NVRAM.

Übrigens, der Hardware RAID Controller kann Datenkorruption nicht erkennen, denn der Hardware RAID Controller prüft nicht die Dateninhalte. So kann der Hardware Controller nicht erkennen, dass die Paritäten zu den Daten nicht stimmen oder ob Daten korrumpiert wurden (zum Beispiel auch durch

¹³ Diese alignment filler gehen zurück auf den Code und den darin enthaltenen roundup Operationen zur Berechnung der benötigten Datenblöcke.

¹⁴ Siehe Jeff Bonwicks Blog Eintrag: http://blogs.sun.com/bonwick/entry/raid_z

¹⁵ ebenda

sogenannte Bit- Kipper).

Der Hardware Controller kennt die Festplatten und ihre Anordnung in logische Laufwerke.

Er kennt aber nicht die Inhalte der Datenblöcke.

Daher geschieht die Korruption schleichend (silent data corruption) verursacht durch beispielsweise eine defekte Festplatte, oder einen defekten RAID-Controller, oder ein defektes Kabel, oder sogar einen defekten SAN-Switch Port.

Erst wenn die Dateninhalte geprüft werden, wie beim Lesen der Daten und beim Prüfen der Checksummen, kann die Datenkorruption festgestellt werden.

Das Prüfen der Dateninhalte geschieht durch das Filesystem oder durch den Volume Manager.

Im Fehlerfall kann das Filesystem "panicen" und der Administrator muss manuell eingreifen (bspw. ein Filesystem Check durchführen), um die Dateninkonsistenzen wieder zu beheben.

Der Volume Manager würde üblicherweise versuchen, die Datenkonsistenz mit Hilfe der verbliebenen, "guten" Blöcke wiederherzustellen (resyncing). Allerdings könnte es sein, dass diese Blöcke ebenfalls korruptiert wurden, dann könnten die Daten nur noch durch ein Wiederherstellen des Backups gerettet werden. Beim Spiegeln von Platten werden die Daten auf beide Spiegelhälften gleichzeitig geschrieben, so dass sich korruptierte Daten auf beiden Spiegelhälften sofort befinden und die Daten unbrauchbar sind.

Applikationen, die korrupte Daten erhalten, werden die Datei schlichtweg nicht lesen können, was dem Datenverlust gleichkommt.¹⁶

Eine weitere Lösung ist das Transaktionslogging. Alle Transaktionen werden mitgeführt (Journaling). Sollte ein Schreibvorgang nicht beendet werden können, beispielsweise durch einen Crash, so gilt er als unvollständig und wird nach dem Reboot erneut ausgeführt. Aber, so Jeff Bonwick, die Software Lösungen sind zu langsam um mehr als nur von akademischen Interesse zu sein.¹⁷

Mit ZFS wurde ein neues Verfahren zur Datenredundanz vorgestellt: RAID-Z.

Es ist dem RAID5 Verfahren sehr ähnlich, verwendet aber, statt fester Streifengrößen, variable, dynamisch angepasste logische Blockgrößen zwischen 512 Bytes und 128 Kilobytes. Jeder Dateisystemblock ist sein eigenes RAID-Z-Stripe-Set, unabhängig von seiner Blockgröße. Aufgrunddessen tritt das Problem des "write-holes" nicht auf.

Wie oben bereits erwähnt beschreibt das "write-hole" die Datenkorruption wenn zwischen dem Schreiben der Daten und der Berechnung der Parität ein Stromausfall eintritt und die Parität nicht zu den geschriebenen Daten passt.

Jeder Schreibvorgang in RAID-Z ist ein "full-stripe-write". Daher ist RAID-Z auch schneller, denn es gibt nicht die Operation des partial-stripe-writes (synchron lesen, modifizieren und Berechnen der neuen Paritäten, dann schreiben), welche sehr zu Lasten der Geschwindigkeit geht. Deswegen und in Verbindung mit der transaktionellen Semantik des Copy-on-write von ZFS gibt es auch keine Gefahr des Datenverlustes durch das "write-hole" und damit auch nicht die Notwendigkeit den Festplattenspeicher mit teurem NVRAM ausstatten zu müssen.

Eine Schlüsselfunktionalität von RAID-Z ist in Kombination mit ZFS die Fähigkeit Datenkorruption nicht nur zu erkennen, sondern auch zu beheben.

16 Constantin Gonzalez beschreibt die möglichen Fehlerquellen für Datenkorruption in seinem Whitepaper zu ZFS

http://blogs.sun.com/constantin/entry/new_zfs_white_paper_in

17 Zitat Jeff Bonwick, 01 Jun 2005, in einer Mail an zfs-interest@sun.com

"The usual workaround is to throw hardware at the problem. If you put an NVRAM write cache in your RAID controller, you can also use that same NVRAM to remember which writes have completed. That way, after a power loss, you can complete any unfinished work. Software solutions (e.g. write logging to disk) also exist, but are too slow to be of more than academic interest."

Herkömmliche Volumemanager können zwar einen Fehler erkennen, z.B. wenn ein Festplattencontroller korrupte Daten liefert, aber nicht unbedingt beheben.

Im Falle eines Lesefehlers wird der Volume Manager versuchen die Daten unter Verwendung der verbliebenen Blöcke zu rekonstruieren, ohne dabei aber feststellen zu können, ob die verbliebenen Daten tatsächlich in Ordnung (silent data corruption).

Auch mit ZFS kann das Problem auftreten: Während des Lesevorgangs wird ein Lesefehler erkannt und die Daten werden rekonstruiert, aber ZFS kann auch überprüfen ob die Daten tatsächlich korrekt sind, indem es die ZFS Prüfsummen nutzt.

ZFS validiert jeden Block gegen die 256-bit Prüfsumme. Traditionelle RAID-Systeme machen das nicht; sie führen lediglich ein XOR der Daten zueinander blindlings durch.

Mit RAID-Z ist die Datenrekonstruktion noch ein wenig trickreicher.

Da die Streifen alle unterschiedliche Grössen haben, da sie ja variabel angepasst sind, gibt es nicht die einfache Formel der XOR Operation nach Null¹⁸. Es müssen bei RAID-Z alle Metadaten des Dateisystems durchlaufen werden um die RAID-Z Geometrie bestimmen zu können und damit auch die Daten rekonstruieren zu können.

Wann immer ein RAID-Z Block gelesen wird, vergleicht ZFS die Daten mit ihrer Prüfsumme. Wenn ZFS feststellt, dass die Daten nicht zur Prüfsumme passen, wird ZFS die Parität lesen und verschiedene Kombinationen der Parität mit den Datenblöcken probieren, um herauszufinden, welche Festplatte korrupte Daten übermittelt. Die defekten Daten werden repariert und erst dann wird ZFS die korrigierten Daten an den User übermitteln.

Ausserdem wird ZFS diesen Fehler an Solaris FMA senden, so dass der Systemadministratör darüber informiert wird, das hier eine "silent-data-Korruption" aufgetreten ist.

Das wäre bei den bisher üblichen Konfigurationen, Dateisystem und RAID Array sind voneinander getrennte Produkte, nicht möglich.

Mit RAID-Z und ZFS hat das Dateisystem einen logischen als auch physikalischen Blick auf die Datenstruktur.

Daher ist RAID-Z auf dem heutigen Storage Markt auch einzigartig.

Schliesslich bleibt noch zu bemerken, dass mit RAID-Z keine spezielle Hardware oder Hardwarekomponenten benötigt werden.

RAID-Z braucht kein NVRAM, um Schreibvorgänge zu beschleunigen und um das write-hole zu vermeiden.

Um es abschliessend mit Jeff Bonwicks Worten zu sagen:¹⁹

"With RAID-Z, ZFS makes good on the original RAID promise: it provides fast, reliable storage using cheap, commodity disks."

ZFS loves cheap disks – ZFS liebt kostengünstige Festplatten – lautet daher auch der Slogan für ZFS.²⁰

¹⁸ Vgl. Seite 8 dieses Dokuments

¹⁹ http://blogs.sun.com/bonwick/entry/raid_z

²⁰ Zu diesem Thema gibt es ein sehr interessantes Video: CSI:Munich von C.Gonzalez, T.Henzen, R.Kersten, C.Müller <http://www.youtube.com/watch?v=1zw8V8g5eT0> und in Englisch: <http://www.systemhelden.com/?p=52>

3. Quellenverzeichnis

http://blogs.sun.com/bmc/entry/welcome_to_zfs A ZFS Overview
<http://docs.sun.com/app/docs/doc/819-5461> ZFS Administration Guide
http://blogs.sun.com/bonwick/entry/raid_z Jeff Bonwicks Blog
http://blogs.sun.com/bonwick/entry/smokin_mirrors What is Resilvering?
http://blogs.sun.com/bonwick/entry/zfs_end_to_end_data About ZFS Data Integrity
http://blogs.sun.com/bonwick/entry/the_general_purpose_storage_revolution The purpose of RAID-Z
http://blogs.sun.com/ahl/entry/double_parity_raid_z RAID-Z2
http://blogs.sun.com/kashmir/entry/1tb_zfs_raid_z_pool RAID-Z
http://blogs.sun.com/timthomas/entry/couple_of_scripts_to_create Examples to create zpools
http://blogs.sun.com/yglee/entry/enter_raid_z RAID-Z again
http://blogs.sun.com/constantin/entry/7_easy_tips_for_zfs Constantin's 7 Tips
http://blogs.sun.com/relling/entry/raid_recommendations_space_vs_mttld Why RAID?
http://blogs.sun.com/roch/entry/when_to_and_not_to When to use mirrors or RAID
<http://www.opensolaris.org/os/community/zfs/> Opensolaris Community ZFS
http://www.theregister.com/2007/10/10/panasas_tiered_parity_raid/ PANANAS new raid view