

Moderne Dateisysteme

Hochleistungs-Ein-/Ausgabe

Michael Kuhn

Wissenschaftliches Rechnen
Fachbereich Informatik
Universität Hamburg

2017-04-28



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG



- 1 Moderne Dateisysteme
 - Orientierung
 - Moderne Dateisysteme
 - ZFS
 - Pools
 - Datensicherheit
 - Datenreduktion
 - Weitere Funktionen
 - Leistungsaspekte
 - Ausblick und Zusammenfassung

2 Quellen

- `stride` gibt die Anzahl der Dateisystemblöcke pro Speichergerät an
- `stripe_width` gibt die Anzahl der Dateisystemblöcke pro Stripe an
 - Stripe bezeichnet hier einen Streifen über die gesamte Breite
 - Üblicherweise $\text{stride} \cdot k$, wobei k die Anzahl der Speichergeräte ist, die Daten enthalten (ohne Parität)

- Unterdateisysteme können wie normale Verzeichnisse benutzt aber auch separat gemountet werden
 - In btrfs Subvolumes genannt

- `async_destroy`: Dateisysteme werden im Hintergrund zerstört
- `lz4_compress`: lz4 steht als Kompressionsalgorithmus zur Verfügung
- `embedded_data`: Dateien, die (nach der Kompression) nicht größer als 112 Bytes sind, können im Blockzeiger gespeichert werden
- `large_blocks`: Blöcke können größer als 128 KiB werden
- Mehr Informationen in der `zpool-features`-Manpage



Pools...

- Pools bestehen aus virtuellen Geräten (vdevs)
 - Daten werden dynamisch darüber verteilt
- Virtuelle Geräte können reale Geräte oder eine Zusammenfassung solcher sein
 - Mirror (RAID 1)
 - RAID-Z (RAID 5 ohne „Write Hole“)
 - RAID-Z2 (RAID 6 ohne „Write Hole“)
 - RAID-Z3 (ohne „Write Hole“)
- Erinnerung: „Write Hole“ ist die Zeit zwischen Schreiben der Daten und der Parität

Transaktionen

- Operationen werden in Transaktionen durchgeführt
 - Auf Dateisystemebene alle Änderungen an Objekten
 - Auf Speicherebene alle Transaktionsgruppen (Daten und zugehörige Metadaten)
- ZFS befindet sich immer in einem konsistenten Zustand
 - Dadurch kein Journaling notwendig
 - Keine Dateisystemüberprüfung mehr notwendig

Copy on Write...

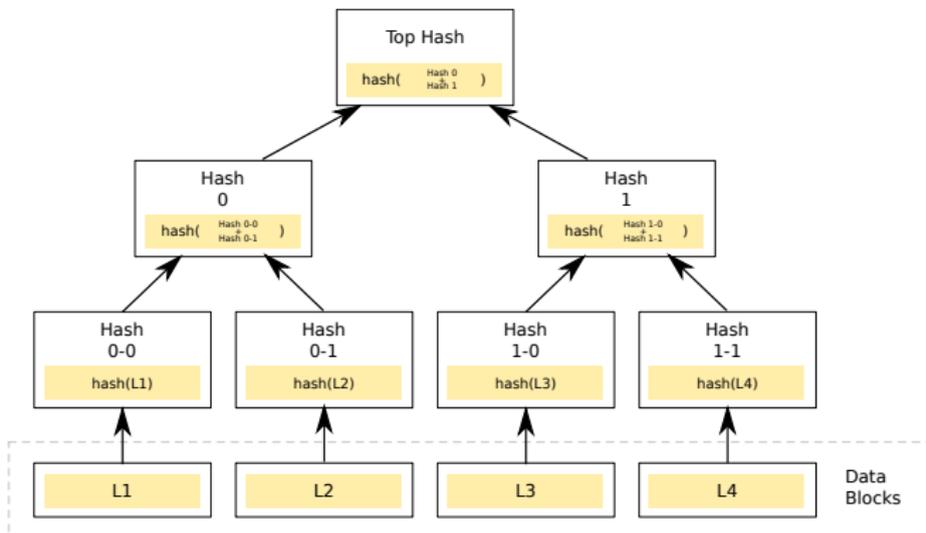


Abbildung: Copy on Write [2]

1 Ausgangszustand

Copy on Write...

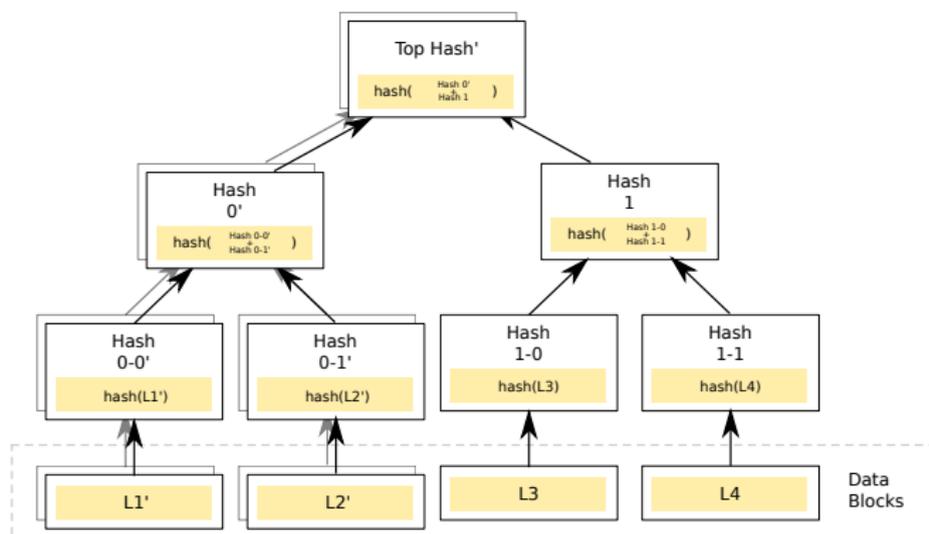


Abbildung: Copy on Write [2]

- 3 Neue Zeigerblöcke werden allokiert und gesetzt

Copy on Write...

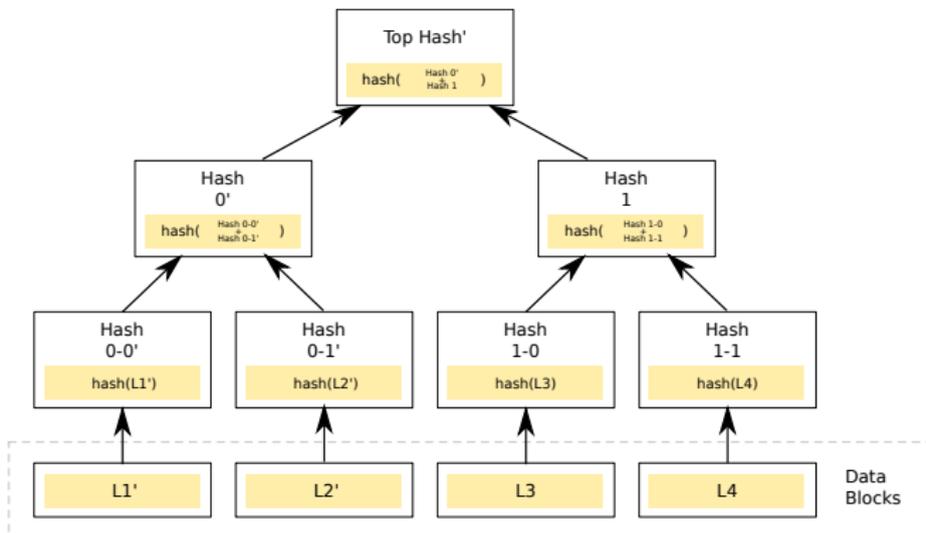


Abbildung: Copy on Write [2]

4 Der „Überblock“ wird aktualisiert



Copy on Write...

- Überblock-Aktualisierung muss atomar geschehen
- ZFS hält ein Überblock-Array mit 128 Einträgen vor
 - Replikate des Arrays werden zur Sicherheit an mehreren Stellen im Pool gespeichert
 - Array wird im Round-Robin-Verfahren benutzt
- Überblöcke enthalten Transaktionsnummer und Prüfsumme
 - Beim Mounten wird der Überblock mit der höchsten Transaktionsnummer benutzt
 - Integrität wird anhand der Prüfsumme überprüft

Schnappschüsse und Klone

- Schnappschüsse vereinfachen einige Anwendungsfälle
 - Vorhalten älterer Daten (z. B. tägliche Schnappschüsse)
 - Mehrere Checkpoints innerhalb einer Datei
 - Momentane Schnappschüsse sind dafür zu grobgranular
- Durch Copy on Write sehr einfach Schnappschüsse möglich
 - Alte Dateisystemwurzel als Schnappschuss speichern
 - Alte Zeiger und Blöcke nicht löschen
- Schnappschüsse können nur gelesen werden
 - Schnappschüsse sind im `.zfs`-Verzeichnis zu finden
 - Benutzer können so selbst auf Backups zugreifen

Schnappschüsse und Klone...

- Schnappschüsse können einfach zurückgerollt werden
 - Überblock ersetzen, ähnlich wie Transaktionen
 - Macht alle Änderungen seit dem Schnappschuss rückgängig
- Veränderbare Schnappschüsse nennen sich Klone
 - Unveränderte Blöcke werden geteilt
 - Änderungen an einem Klon durch neue Blöcke
- Klone sind ebenso einfach zu realisieren
 - Durch Copy on Write nur zusätzlicher Speicherverbrauch für neue und geänderte Daten

Backups

- Backups großer Speichersysteme sind problematisch
 - Traditionell Überprüfung des gesamten Namensraumes
 - Effizientere Abwicklung durch Schnappschüsse
- Vollständiges Backup
 - Als Grundlage dient ein beliebiger Schnappschuss
- Inkrementelles Backup
 - Grundlage sind Änderungen zwischen zwei Schnappschüssen
 - Aufwand hängt von den geänderten Daten ab
- Damit ist auch Replikation realisierbar
 - Erstelle minütlich Schnappschüsse und transferiere die inkrementellen Backups per SSH auf den anderen Server



RAID

- Fehlerszenario auf einem Mirror aus zwei Festplatten
 - 1 Programm liest Daten
 - 2 ZFS liest einen Block von der ersten Festplatte
 - 3 Es wird erkannt, dass der Block fehlerhaft ist
 - 4 ZFS liest die Blockkopie von der zweiten Festplatte
 - 5 Es wird erkannt, dass die Blockkopie korrekt ist
 - 6 Der fehlerhafte Block wird mit dem korrekten überschrieben
 - 7 Die Daten werden an die Anwendung weitergegeben
- Bei traditionellen Dateisystemen fallen die Schritte 3–6 weg
- ZFS kann erkennen, wenn beide Kopien fehlerhaft sind



RAID...

- Traditionelle RAID-Systeme können Fehler nur erkennen
 - Dafür müssten bei jedem Zugriff die zugehörigen Paritätsdaten gelesen und verglichen werden
 - Im Fehlerfall ist nicht klar ob Daten oder Parität korrekt sind
- Erkennung und Korrektur defekter Daten ist sehr wichtig
 - Teilweise sehr teuer zu berechnen
 - Aufgrund der Größe manchmal keine Backups

Resilvering

- Rekonstruktion der Daten in einem RAID-Verbund
- Traditionell musste alles rekonstruiert werden
 - Strenge Trennung zwischen Verbund und Dateisystem
 - Simples XOR über die noch vorhandenen Daten
 - Keine Möglichkeit die Korrektheit zu verifizieren
- Nur noch tatsächlich vorhandene Blöcke rekonstruieren
 - Bei temporärem Ausfall nur zwischenzeitlich veränderte Daten
 - Mehr Datensicherheit durch Top-Down-Rekonstruktion
 - Verlust von Blöcken auf den oberen Ebenen fatal

- Die oberen Ebenen beziehen sich hierbei auf den Hash-Baum
 - Ein fehlerhafter Knoten macht den ganzen zugehörigen Teilbaum unzugänglich

Kompression...

- zle eliminiert Null-Sequenzen
 - Zero-Length Encoding
 - Üblicherweise geringe Kompressionsrate
 - Wird immer angewendet sobald Kompression aktiviert ist
- gzip komprimiert gut aber langsam
 - Unterstützt mehrere Komprimierungslevel (1-9)
 - Selbst schnelle Level recht langsam (≈ 50 MB/s)
 - Dekompression schneller (≈ 300 MB/s)

Volumes

- ZFS unterstützt auch sogenannte Volumes
 - Exportiert als Blockgerät
 - Nützlich um andere Dateisystem auf Pools zu nutzen
- Alle Poolfunktionen können genutzt werden
 - Schnappschüsse, Komprimierung etc.

```
1 $ zfs create -V 4G tank/swap
2 $ zfs create -V 75G tank/home
3 $ mkswap /dev/zvol/tank/swap
4 $ mkfs.ext4 /dev/zvol/tank/home
```

Listing 1: ZFS-Volume

Diverses

- ZFS arbeitet mit eingeschaltetem Festplatten-Cache korrekt
 - Einzige Voraussetzung: Die Festplatten müssen die Flush-Kommandos befolgen
 - Bei einigen Dateisystemen kann Datenverlust auftreten
- ZFS nutzt dynamische Blockgrößen von bis zu 128 KiB
 - Kann auf 1 MiB erhöht werden
 - Die maximale Blockgröße kann manuell angepasst werden

Caching

- ZFS nutzt einen Pipeline-Scheduler
 - Jede Operation hat eine Priorität und eine Deadline
 - Höhere Priorität resultiert in kürzerer Deadline
 - Leseoperationen erhalten höhere Priorität als Schreiboperationen
- Operationen können zusammengefasst und umsortiert werden
 - Sonst bei jeder Änderung komplette Unterbaumkopie
 - Macht effizientes Copy on Write möglich

Caching...

- ZFS nutzt zwei Cache-Level: ARC und L2ARC
 - Der Adaptive Replacement Cache befindet sich im RAM
 - Der L2ARC befindet sich üblicherweise auf SSDs
- Alle Zugriffe werden durch den ARC beschleunigt
 - Dafür ist ausreichend viel RAM notwendig
 - Dort werden auch die Deduplikationstabellen gespeichert
- Befinden sich Daten nicht im ARC, wird auf den Pool zugegriffen



- 1 Moderne Dateisysteme
 - Orientierung
 - Moderne Dateisysteme
 - ZFS
 - Pools
 - Datensicherheit
 - Datenreduktion
 - Weitere Funktionen
 - Leistungsaspekte
 - Ausblick und Zusammenfassung

2 Quellen



Quellen I

- [1] Seagate. Desktop HDD. http://www.seagate.com/www-content/datasheets/pdfs/desktop-hdd-8tbDS1770-9-1603DE-de_DE.pdf.
- [2] Wikipedia. Hash-Baum. <http://de.wikipedia.org/wiki/Hash-Baum>.
- [3] Wikipedia. Standard RAID levels. http://en.wikipedia.org/wiki/Standard_RAID_levels.