HAMMER Filesystem

von

Timm Müller

vorgetragen am

13.07.2012

Betreuer

Michael Kuhn

Inhaltsverzeichnis

1. Allgemein

1.1. Was ist HAMMER überhaupt?

HAMMER ist ein 64-Bit Cluster-Dateisystem.
Dabei hat „Cluster“ nicht die übliche Bedeutung, das heißt es ist kein Dateisystem, das einen zentralen Speicher für einen Rechnercluster bildet, stattdessen ist gemeint, dass das Dateisystem selbst in ein Cluster zerteilt werden kann.
HAMMER wurde als Dateisystem für DragonFlyBSD entwickelt und seit Version 2.0 des Betriebssystems als Standard eingesetzt. Vorher war das Standarddateisystem von DragonFlyBSD UFS, welches noch immer neben HAMMER während der Installation zur Auswahl steht.
Das Dateisystem wurde gezielt für heutige Festplattengrößen ausgelegt und benutzt viele Features, die sehr viel Speicherplatz benötigen. Daher empfehlen die Entwickler eine Mindestgröße von 50GB für eine HAMMER Partition.

1.2 DragonFlyBSD

DragonFlyBSD ist ein Unix-Derivat aus der Familie der BSDs. Es wurde in den Jahren 2003 bis 2004 von Matthew Dillon als Fork von FreeBSD geschaffen.
Dillon wollte mit seinem neuen System einen besonderen Fokus auf Multiprozessorsysteme und Cluster legen.
Zu dem Fork führten diverse Meinungsverschiedenheiten zwischen Matthew Dillon und den restlichen Entwicklern von FreeBSD.
Dillon war zuvor stark an der Entwicklung von FreeBSD beteiligt und hatte die Möglichkeit direkt Änderungen an der Codebase beizusteuern. Jedoch waren die meisten FreeBSD Entwickler nicht einverstanden mit einigen seiner Änderungen, sodass ihm diese Rechte schließlich entzogen wurden. Besonders mit der Umsetzung von Threading und Multiprocessing war Dillon unzufrieden, jedoch wurden seine Ideen nicht angenommen, woraufhin für Dillon nur noch die Möglichkeit eines Forks blieb.

1.3 Was kann HAMMER denn so?

Als Hauptfeatures für HAMMER wird unter anderem die „instant crash recovery“ genannt, die garantieren soll, dass nach einem Absturz das Dateisystem sofort wieder verfügbar sein soll. Außerdem wird die Historie angepriesen, die auf zwei Arten angeboten wird. Einmal bietet die „fine-grained“ Historie einen Zugriff auf die Daten und Veränderungen der letzten 30-60 Sekunden an und außerdem werden Snapshots gespeichert, die bis zu 60 Tage zurückreichen.
Auch interessant sind die PFSs, die „pseudo-file-systems“, die genau das sind, was der Name vermuten lässt. So lässt sich eine HAMMER Partition in mehrere PFSs aufteilen, sodass eine HAMMER Partition aus einem Cluster aus Tausenden PFSs bestehen kann.

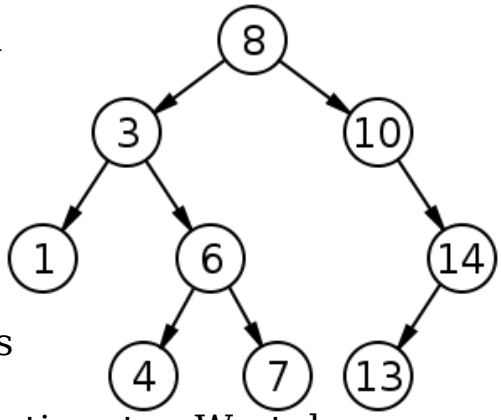
2. Features genauer betrachtet

2.1 Spezifikationen

HAMMER ist ein 64-bit Dateisystem, dass bis zu einem Exbibyte Speicherkapazität bietet. Es benutzt B+ Trees für die Verzeichnisstruktur; B+ Trees werden im nächsten Teil genauer erklärt. Zur Speicherplatzersparnis unterstützt das Dateisystem on-demand data deduplication, welche ich später auch genauer behandeln werde. Zur Sicherung der Datenintegrität werden bei HAMMER sowohl gespeicherte Daten als auch Metadaten mittels CRC-checks auf Integrität geprüft.

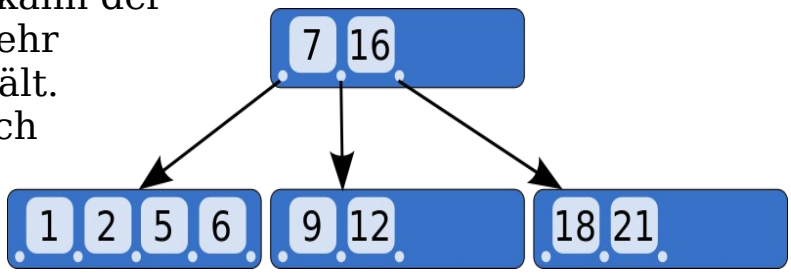
2.2 B+ Tree

Der B+ Baum ist ein Suchbaum, der in HAMMER für die Verzeichnisstruktur benutzt wird.
Der Vorteil eines Suchbaumes ist, dass man gesuchte Daten sehr schnell finden kann. Bereits beim einfachen Binären Suchbaum kann man dies feststellen. Ein Knoten in einem binären Suchbaum hat bis zu 2 Kindern, also folgende Knoten. Dabei hat, wie man im Beispiel rechts erkennen kann das linke Kind einen kleineren und das rechte Kind einen größeren Wert.

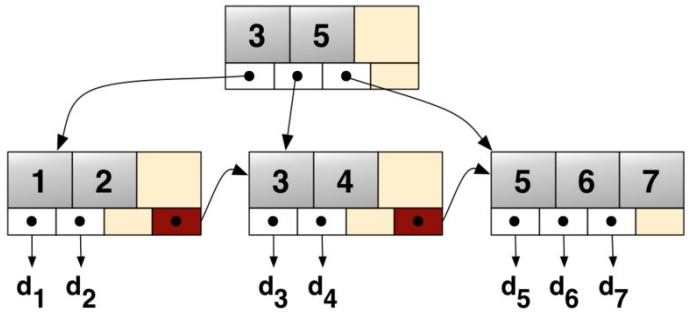


Geht man nun auf der Suche nach einem bestimmten Wert den Baum entlang, wird man den gesuchten Wert im Durchschnitt deutlich schneller finden, als wenn man eine einfach Liste durchsuchen müssen.

Der B-Tree ist dabei sogar noch effektiver, da er die Tiefe des Baumes stark reduziert und damit im Durchschnitt schneller zum Ergebnis führt. Das B beim B-Tree kommt dabei aber nicht, wie sich vermuten ließe, von Binär, denn der B-Tree muss nicht zwangsweise ein Binärer Baum sein. Es ist nicht genau bekannt, was das B Ursprünglich bedeuten sollte.
Die Knoten im B-Tree sind in Schlüssel und Verweise aufgeteilt. Dabei kann der Knoten immer einen Verweis mehr enthalten, als er Schlüssel enthält. Die Schlüssen können dabei auch den gesuchten Datenwert darstellen. Im Beispiel rechts kann man sehen, dass man eine ähnliche Struktur und Logik hat, wie auch im Binären Suchbaum. So kann man den Verweis links vom Schlüssel nutzen um zu kleineren Werten zu gelangen. Dadurch, dass man nun mehrere Kinder haben kann, wird der gesamte Baum gestaucht und somit eine schnellere Suche ermöglicht.



Der B+ Tree baut auf dem B-Tree auf. Bei ihm sind die Werte ausschliesslich in den Blättern, also den Knoten ohne Kinder, gespeichert. Schlüssen sind tatsächlich nur dafür da um den gesuchten Wert zu finden.
Wie man im rechten Beispiel sehen kann, ist der Baum ähnlich aufgebaut, wie der normale B-Tree, nur dass die Werte in den Blättern gespeichert sind und die Knoten zusätzlich Querverweise enhalten, wodurch man auch Seitwärts von Knoten zu Knoten gelangen kann, anstatt nur zu Kindern. Der B+ Tree wird wird zum Beispiel von diversen bekannten Dateisystemen und Datenbanken verwendet, wie NTFS, ReiserFS, XFS, Microsoft SQL sowie Oracle 8.



2.3 Instant crash recovery

Die instant crash recovery ist ein angepriesenes Feature des HAMMER Dateisystems. Bekannte und verbreitete Dateisysteme, wie NTFS oder ext4, benötigen nach einem Systemabsturz eine Überprüfung um Datenkonsistenz zu gewährleisten. Bei ext4 wird im Falle eines Absturzes fsck ausgeführt um diese Überprüfung durchzuführen. Dies ist bei HAMMER nicht nötig, da es die Historie nutzt um den letzten Stimmigen Zustand wieder herzustellen. Laut Angaben von Matthew Dillon soll dies innerhalb einer Sekunde während dem Einhängen des Dateisystems geschehen.

2.4 Historie

HAMMER speichert eine genaue Historie ab, auf die man freien Zugriff hat, wodurch man Änderungen kurzfristig problemlos rückgängig machen kann. Diese Historie wird fine-grained (dt. feinkörnig) genannt und deckt standardmäßig die letzten 30-60 sekunden ab. Diese Zeit kommt daher, dass etwa alle 30-60 Sekunden eine sogenannte prune Operation ausgeführt wird, die unter anderem diese Historie bis zum Zeitpunkt des letzten Snapshots löscht. Da diese Snapshots alle 30 Sekunden gemacht werden enthält die fine-grained Historie mindestens 30 Sekunden, da aber nach spätestens 60 Sekunden wieder ein prune ausgeführt wird, sind immer maximal 60 Sekunden gespeichert.

Die eben erwähnten Snapshots bilden die so genannte coarse-grained (dt. grobkörnig) Historie. Standardmäßig besteht diese aus 30 Sekündigen Snapshots, die einen Tag lang gespeichert bleiben sowie Tägliche Snapshots, die 60 Tage lang gespeichert bleiben. Auf diese Snapshots ist jederzeit live zugriff möglich, man kann also eine Version einer Datei oder eines Ordners abrufen, die bis zu 60 Tage alt ist. Außerdem bietet HAMMER den UNDO Befehl an, mit dem einfach die Wiederherstellung einer beliebigen Version ermöglicht wird. Zusätzlich kann man mit diesem Befehl auch diff Output erzeugen, der die Unterschiede zur aktuellen Version enthält.
Die täglichen Snapshots werden im Laufe eines nächtlichen Cronjobs erstellt, der neben diesen Snapshots auch die rebalance (das Ausbalancieren des B+ Trees für schnellere Suchergebnisse), dedup (die data deduplication) und reblock (neu Anordnen der Blocks um Fragmentierung zu verhindern) Befehle ausführt.

2.5 Data deduplication

Data deduplication bedeutet, dass doppelt vorkommende Daten nur einmal gespeichert und dafür zwei Referenzen auf den gleichen Datensatz abgelegt werden. Diese Funktion ist optional in HAMMER, wird also nicht erzwungen, stattdessen wird durch einen Befehl das Dateisystem überprüft und doppelte Daten durch Referenzen auf das erste Vorkommen ersetzt.
Durch diese Funktion lassen sich zwischen 10 bis 40 Prozent Speicherplatz sparen.
Die Überprüfung findet auf Block Level statt, also wird nicht nach doppelten Dateien gesucht, sondern nach identische Blöcken.
Der Vergleich findet durch CRC-32 checks statt, wodurch kein unnötiger zusätzlicher Overhead verursacht wird, da CRC-checks ohnehin schon für Datenintegrität durchgeführt werden.
Momentan wird data deduplication nur per PFS ausgeführt, es ist aber angedacht dies irgendwann auch übergreifend zu ermöglichen, da man dadurch noch mehr Platz sparen könnte.

2.6 PFSs

PFS steht für „Pseudo File System“, was genau das bedeutet, was der Name vermuten lässt. Ein HAMMER Dateisystem kann in bis zu 65536 PFSs unterteilt werden, die im Prinzip virtuelle Dateisysteme darstellen. Sie teilen sich den verfügbaren Speicherplatz dynamisch, was bedeutet, dass das PFS immer so groß ist, wie die Daten, die darin gespeichert sind.
Es gibt so genannte Master und Slave PFSs, wobei die Slave PFSs für den Nutzer immer read-only sind, HAMMER eigene Operationen können natürlich auch schreibend zugreifen. Eine sinnvolle Möglichkeit der Nutzung eines Slaves ist z.B. Mirroring.

2.7 Mirroring

Mirroring findet beim HAMMER Dateisystem über PFSs statt, wobei ein Master PFS oder auch ein Slave PFS auf ein anderes Slave PFS gespiegelt wird. Dafür bietet HAMMER befehle an, die eine einmalige Kopie eines PFS auf einen Slave abbilden, oder auch einen Stream anbieten, mit dem man alle Änderungen am Ursprungs PFS an den Spiegel weiterleiten kann.
Dabei kann man mehrere Slave PFSs als Mirror verwenden, aber natürlich kann man nicht mehre PFSs auf einen Mirror abbilden.

2.8 Multi-Volume

Ein HAMMER Dateisystem kann sich auf mehrere Physikalische Volumes erstrecken, dabei sind bis zu 256 verschiedene Volumes möglich. Dabei kann ein Dateisystem bis zu 4 Petabytes Kapazität haben, aufgeteilt auf jeweilt 4096 Terabytes pro Volume.

3 Vergleich

HAMMER schneidet im Vergleich nicht allzu schlecht ab. So ist es schneller als der vorherige Standard bei DragonFlyBSD UFS. Dabei ist es aber dennoch langsamer als ZFS, welches auch oft als die Zukunft angesehen wird. Diese Vergleiche sind aber nicht als eindeutig anzusehen, da jedes Dateisystem Vor- und Nachteile hat und sich unter verschiedenen Situationen unterschiedlich verhält. Jedoch sind diese Tendenzen gerade im Vergleich mit UFS teilweise sehr deutlich zu sehen.
Ein Vorteil von HAMMER ist, dass es eine DragonFlyBSD eigene Entwicklung ist, während ZFS von Solaris geportet werden musste. Da UFS nur sehr wenig herausstechende Features hat, liegt HAMMER in dem Bereicht sehr viel weiter vorne, jedoch kann ZFS noch mehr Features aufweisen. Allerdings ist HAMMER auch noch deutlich jünger als ZFS, wodurch dies nicht verwunderlich ist.

4 Fazit

HAMMER ist ein neues und junges Dateisystem, dass mit vielen guten und interessanten Ideen überzeugt. Für sein junges Alter hat es bereits ein großes Featureset und gute Performance. Es sind noch viele Verbesserungen und Erweiterungen der vorhandenen Features geplant und auch bei der Performance wird man Verbesserungen erwarten können.
Dadurch ich HAMMER ein sehr interessantes Dateisystem, auf das man durchaus ein Auge werfen sollte, sofern man sich für die Welt der BSDs interessiert.

5 Quellen

- http://www.dragonflybsd.org/hammer/
- http://leaf.dragonflybsd.org/cgi/web-man?command=hammer&section=5
- http://leaf.dragonflybsd.org/cgi/web-man?command=hammer&section=8
- http://leaf.dragonflybsd.org/cgi/web-man?command=newfs_hammer&section=8
- https://en.wikipedia.org/wiki/DragonFly_BSD
- https://en.wikipedia.org/wiki/HAMMER
- https://en.wikipedia.org/wiki/Unix_File_System
- https://en.wikipedia.org/wiki/ZFS
- https://en.wikipedia.org/wiki/Binary_Search_Tree
- https://en.wikipedia.org/wiki/B-tree
- https://en.wikipedia.org/wiki/B%2B_tree
- http://kerneltrap.org/DragonFlyBSD/HAMMER_Crash_Recovery
- http://www.phoronix.com/scan.php?page=article&item=dragonfly_hammer