

ZFS

Tim Langens
Jef Neefs

1 Inhoudstafel

1	Inhoudstafel	2
2	Geschiedenis	3
3	Capaciteit	3
3.1	128-bit.....	3
3.2	De theoretische opslag limiet.....	3
4	Opslagpools	4
5	Copy-on-write transactie model	5
6	Snapshots en klonen	5
7	Variabele blok grootte	5
8	Beperkingen	5
9	Platforms	6
10	Gedistribueerde systemen	6
11	Concurrentie	6
12	Bronnen	7
A	Websites.....	7
B	Boeken	7

2 Geschiedenis

ZFS is ontworpen en geïmplementeerd door een team van Sun geleid door Jeff Bonwick. Op 31 oktober 2005 werd de broncode van het uiteindelijke product geïntegreerd in de main trunk van Solaris en op 16 november 2005 werd het uitgebracht geïmplementeerd in OpenSolaris build 27. In juni 2006 kondigde Sun aan dat ZFS geïntegreerd werd in update 6/06 van Solaris 10, één jaar na de start van de OpenSolaris community.

De naam verwijst naar “Zettabyte File System”, maar is nu gekend als een letterwoord. Dit omdat de datalimiet een zettabyte is.

3 Capaciteit

ZFS is een 128-bit bestandssysteem, dit wil zeggen dat het 18 miljard miljard (18 triljoen) keer meer data kan opslaan dan de huidige 64-bit systemen.

3.1 128-bit

De wet van Moore zegt dat de opslagcapaciteit elk anderhalf jaar verdubbeld. Er bestaan nu al datasets in de orde van petabytes (2^{50} byte), dit wil zeggen dat de limiet van de huidige 64-bit systemen nog 14 verdubbelingen verwijderd is, wat overeenkomt met ongeveer 20 jaar. Omdat bestandssystemen vaak enkele decennia blijven bestaan, zou het dom zijn om een nieuw bestandssysteem te creëren dat binnen enkele jaren niet meer aan de noden voldoet en heeft men dus voor een 128-bit bestandssysteem gekozen.

Zoals Jeff Bonwick zelf zegt:

“If 64 bits isn't enough, the next logical step is 128 bits. That's enough to survive Moore's Law until I'm dead, and after that, it's not my problem.”

3.2 De theoretische opslag limiet

Kwantum mechanica zorgt voor enkele fundamentele limieten voor de *computation rate* en de informatie capaciteit van een fysische schijf. Hierdoor zal de wet van Moore ooit doorbroken moeten worden. Als we dit nader bekijken blijkt dat een volledig beschreven 128-bit opslag pool 2128 blocks nodig heeft. 2^{128} blocks = 2^{137} bytes = 2^{140} bits. In het boek van Seth Lloyd, “Ultimate physical limits to computation.” Vinden we dat 1kg materie beperkt tot 1 liter ruimte 1051 operaties per seconde kan doen met maximum 1031 bits aan informatie. Dit wil dus zeggen dat we minstens 136 miljard kg aan materie nodig hebben om dit op te slaan. Om deze hoeveelheid aan data te creëren, heb je meer energie nodig dan de energie nodig om de oceanen te doen koken.

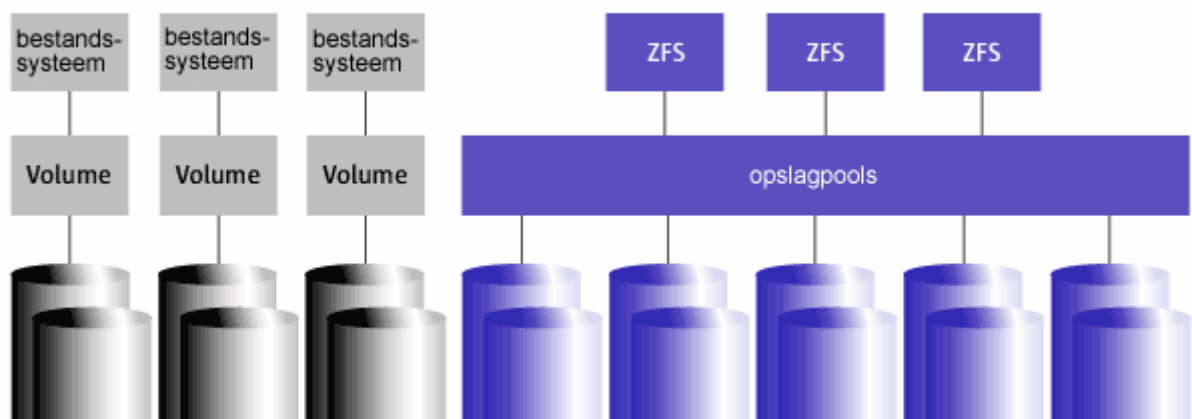
Enkele theoretische limieten van ZFS:

2^{48}	Aantal snapshots in elk bestandssysteem (2×10^{14})
2^{48}	Aantal bestanden in elk individueel bestandssysteem (2×10^{14})
16 exabytes	Maximum grootte van het bestandssysteem
16 exabytes	Maximum grootte van één enkel bestand
16 exabytes	Maximum grootte van elk attribuut
3×10^{23} petabytes	Maximum grootte van elke zpool
2^{56}	Aantal attributen van een bestand (beperkt tot 2^{48} aangezien dit het aantal bestanden is per bestandssysteem)
2^{56}	Aantal bestanden in een map (beperkt tot 2^{48} aangezien dit het aantal bestanden is per bestandssysteem)
2^{64}	Aantal apparaten in elke zpool
2^{64}	Aantal zpoolen per systeem
2^{64}	Aantal bestandssystemen in een zpool

Om een idee te geven van de grootte van deze getallen, als er elke seconde 1000 bestanden worden gecreëerd duurt het nog 9000 jaar voor deze limieten worden bereikt.

4 Opslagpools

Met traditionele volumes wordt data gefragmenteerd opgeslagen. Je hebt dan ook een volumemanager nodig om met verschillende apparaten te communiceren. Met ZFS' opslagpools moeten er geen partitie's beheerd worden. De gecombineerde I/O bandbreedte van alle apparaten in de opslagpools zijn altijd bereikbaar voor elk bestandssysteem.



Een pool bestaat uit virtuele apparaten (virtual devices – vdevs). Deze apparaten kunnen een vaste schijf zijn, een mirror (RAID 1) van één of meer apparaten of een RAID Z groep van twee of meer apparaten. ZFS kan gebruik maken van al deze virtuele apparaten door middel van de zpool.

Een pool kan gereserveerd worden, zodat een bepaald bestandssysteem opslagruimte heeft. Er kunnen ook quota's opgelegd worden zodat de pool niet teveel opslagruimte in beslag neemt.

5 Copy-on-write transactie model

Alle bewerkingen zijn copy-on-write transacties. Hierdoor moet het ZFS bestandssysteem nooit nagekeken worden op fouten. Elk blok wordt nagekeken om stille data corruptie te vermijden, de data hersteld zichzelf indien er gebruik gemaakt wordt van een replica configuratie (RAID). Als een kopie beschadigd is zal ZFS dit detecteren en een andere kopie gebruiken om de fout te herstellen.

6 Snapshots en klonen

ZFS voorziet ongelimiteerd gebruik van snapshots en klonen. Een snapshot is een moment-opname van een bestandssysteem dat enkel te lezen is. Een kloon is een snapshot die aangepast kan worden. Door het gebruik van klonen ontstaat er een plaatsbesparende manier om vele kopieën van veel gedeelde data op te slaan.

Snapshots helpen in het back-up en herstel proces van ZFS. Elke snapshot kan een volledige back-up genereren. De mogelijk bestaat ook om een incrementele back-up te genereren door verschillende snapshots samen te gebruiken. Incrementele back-ups zijn maken het gebruik van voor herstel op afstand zeer efficiënt.

7 Variabele blok grootte

De huidige bestandssystemen hebben meestal een vaste blok grootte, hierdoor gaat er veel schijfruimte verloren aan overhead. Het voordeel van vaste blok grootte is dat het gemakkelijk te implementeren is.

ZFS echter gebruikt variabele blok grootte tot 128 KB. Het voordeel hiervan is dat je grote blokken voor grote bestanden kunt gebruiken en deze blokken kunt opdelen in kleine blokken voor kleine bestanden. Indien een klein bestand dan toch groot blijkt te worden kun je die gemakkelijk nog even uit een klein blok halen en in een groot blok plaatsen. Doordat het tot dan toe een klein bestand was geeft dat (eenmalig) verplaatsen ook niet echt veel overhead.

8 Beperkingen

Aangezien de huidige opstartsystemen ZFS nog niet ondersteunen, kan je ZFS niet als opstartbestandssysteem instellen. Het "ZFS boot project" probeert dit probleem op te lossen.

ZFS heeft geen transparante encryptie, zoals NTFS. Momenteel kan je enkel n+1 redundantie toepassen, n+2 redundantie (RAID 6) is nog in ontwikkeling.

Om de 64-bit checksums, gebruikt bij copy-on-write, te berekenen gaat de processorbelasting wel wat omhoog, maar in de praktijk blijkt dat verwaarloosbaar. Zeker met de huidige processoren.

ZFS-gebruikers moeten er van tevoren voor zorgen dat ze op hun harde schijven uitsluitend EFI-labels gebruiken. Alleen met behulp van deze labels kan je namelijk bestandssystemen die op sparc-hardware zijn aangemaakt lezen op x86/x64-systemen. Met het oude labelformaat gaat dat niet. Deze beperking geldt enkel voor harde schijven en niet voor andere vdevs.

9 Platforms

ZFS wordt uiteraard geïmplementeerd in Sun's Solaris en OpenSolaris. Hierdoor is het bestandssysteem bruikbaar op SPARC en x86 systemen. ZFS is open-source en dus kan men zonder tussenkomst van Sun, ZFS implementeren in andere besturingssystemen.

Apple toont interesse om ZFS te implementeren in Mac OS X.

ZFS implementeren in Linux is moeilijker omdat ZFS onder de CDDL licentie is uitgegeven en de Linuxkernel onder GPL wordt verdeeld.

Ook de BSD-wereld ziet een implementatie van ZFS wel zitten.

10 Gedistribueerde systemen

ZFS leent zich enorm voor gedistribueerde systemen, omdat alle schijven van alle systemen in één Zpool kunnen gezet worden. Zo kan het lijken dat je één grote schijf hebt die bestaat uit alle schijven van elke aparte computer. Het nadeel hiervan is wel dat je netwerk een flessenhals gaat vormen omdat normale schijftoegang via een IDE of SATA kabel gaat, die vele sneller zijn dan de meeste huidig gebruikte ethernet snelheden.

Wat ZFS zo goed maakt om te draaien op gedistribueerde systemen is de aanpassende endian of in het engels "adaptive endiannes" de ZFS block pointer laat toe dat de metadata in big, middle of little endian kan opgeslagen worden. De metadata wordt weggeschreven in de oorspronkelijke endiannes en als er gelezen wordt van een andere endiannes wordt de metadata op een correcte manier omgevormd in het geheugen naar de correcte endiannes waardoor het leesbaar en schrijfbaar is voor ELK systeem.

11 Concurrentie

Voorlopig heeft ZFS één grote concurrent, VxFS. VxFS is speciaal ontworpen voor gedistribueerde systemen en heeft als hoofdvoordeel tov andere file systemen dat de defragmentatie miniem tot onbestaande is. Bij ZFS is dit ook zo dmv de variabele blok grootte. ZFS heeft daarbij een praktisch onbereikbare datalimiet wat het een enorm voordeel geeft tov VxFS.

12 Bronnen

A Websites

blogs.sun.com/bonwick/date/20040925#128_bit_storage_are_you

www.haifa.il.ibm.com/projects/storage/zFS/public.html

www.opensolaris.org/os/community/zfs/demos/selfheal/

www.opensolaris.org/os/community/zfs/demos/basics/

www.opensolaris.org/os/community/zfs/

<http://en.wikipedia.org/wiki/Zfs>

<http://www.sun.com/2004-0914/feature/>

B Boeken

CT magazine oktober 2006 artikel ZFS door Thomas Nau