VISOKA ŠKOLA ZA PRIMIJENJENO RAČUNARSTVO

ZAGREB

SEMINARSKI RAD

Predmet: Operativni sustavi

ZFS (zettabyte file system)

Vladimir Škorić

Zagreb, lipanj, 2009

ZFS

Iako razvijen od strane Sun Microsystems za operativni sustav Solaris nije namjenjen isključivo za Solaris.

Osim na Solarisu ZFS je dostupan na sljedećim platformama:

- BSD
- Mac OS X
- Linux

Za pripremu rada teksta korišten je ZFS na Linuxu počevši od instalacije.

Trenutno Linux driver za ZFS nije smješten u kernel zbog nekompatibilne licence pa se ZFS izvodi u tzv. user-spaceu poput korisničkog programa što može biti uzrok nešto slabijim perfomansama u odnosu na one čiji su driveri u kernelu ali bez obzira na to može nam ozbiljno poslužiti jer perfomanse nisu jedino što ZFS obećava niti su one loše.

Da eventualnom čitatelju olakšam isprobavanje ZFS-a odabrao sam Ubuntu na koji ću instalirati ZFS, napraviti neke testove i pokušati naći primjenu, odnosno vidjeti donosi li ZFS nešto "običnom" korisniku laptopa ili je njegova primjena vezana samo uz servere i vrlo velike file sisteme.

Vrlo velike file sisteme zato što :

ZFS koristi 128-bitnu adresnu shemu koja može sadržavati 256 kvadriliona zetabajta. Zetabajt je 2^{70} bajtova ili 1 milijarda terabajta iz čega je nastao naziv ZFS – Zettabyte File System. Svi smo već više nego zadovoljni sa brojkama koje pruža 64-bit shema no 128-bitna je daleko veća a obje su nezamislivo veće od onoga što trenutno možemo iskoristiti.

Krenimo s instalacijom ZFS drivera, Filip Brčić je napravio paket za ubuntu pa je dovoljno konfigurirati apt tj. dodati repozitorij zfs paketa čime je svedena na vrlo jednostavnu proceduru:

```
vskoric@inf0011l:~$ sudo su -
[sudo] password for vskoric:
root@inf0011l:~# vi /etc/apt/sources.list.d/zfs-fuse.list
```

pomoću vi ili nekog drugog editora dodamo sljedeća 2 retka u novu datoteku zfs.fuse.list

```
deb http://ppa.launchpad.net/brcha/ubuntu jaunty main
deb-src http://ppa.launchpad.net/brcha/ubuntu jaunty main

root@inf0011l:~#aptitude update
root@inf0011l:~#aptitude install zfs-fuse
```

```
Setting up zfs-fuse (0.5.1-1ubuntu2) ...
 * Starting zfs-fuse zfs-fuse
[ OK ]
 * Mounting ZFS filesystems...
[ OK ]
```

i ZFS je spreman za korištenje na disku ili particiji.

Kratak opis okoline na kojoj testiram ZFS:

na svom računalu imam samo jednu neiskorištenu particiju veličine ~10Gb koju ću pomoću LVM-a (Logical Volume Manager) sustavu prikazati kao više manjih blok uređaja.

Odlučio sam se na LVM jer time neću mjenjati tablicu particija na samom hard disku što bi mi poremetilo redosljed i vjerovatno bi zahtjevalo restart. Ukratko LVM mi omogućuje elegantno brisanje dodavanje novih particija (logičkih volumena), te logičke volumene ću ZFS-u prikazati kao hard diskove koji će sačinjavati ZFS pool, istina mogao sam preskočiti LVM i ZFS-u dodjeliti cijelu slobodnu logičku particiju sda8 kao hard disk no onda bi to bio jedini zfs disk, s LVM ću od ovih 10GB napraviti nekoliko npr. 5 particija od kojih ću napraviti npr. raid5, raid1,raid0 ZFS polja tj. zfs datasetove.
Naravno u praksi ovako nešto nema previše smisla no za potrebe demonstracije i testiranja mogućnosti zfs može poslužiti.

Konfiguracija LVM-a – (sda8 možete zamjeniti vašom particijom sve ostalo može ostati isto)

```
pvcreate /dev/sda8 -v
vgcreate -A y  4ZFS /dev/sda8 -v

lvm> pvdisplay
  --- Physical volume ---
  PV Name               /dev/sda8
  VG Name               4ZFS
  PV Size               10.36 GB / not usable 1.38 MB
  Allocatable           yes
  PE Size (KByte)       4096
  Total PE              2651
  Free PE               2651
  Allocated PE          0
  PV UUID               BbtFky-9DYS-Saky-mRBL-8OBv-hatk-2OmfQz


 lvm> vgdisplay
  --- Volume group ---
  VG Name               4ZFS
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
```

```
  VG Size               10.36 GB
  PE Size               4.00 MB
  Total PE              2651
  Alloc PE / Size       0 / 0
  Free  PE / Size       2651 / 10.36 GB
  VG UUID               zf3C50-lCyj-DkSX-I0Cr-CGmC-1kgA-MC3jsm

lvm> lvcreate -C y   -p rw -L 1G -n zfsdisk1 4ZFS /dev/sda8
  Logical volume "zfsdisk1" created
lvm> lvcreate -C y   -p rw -L 1G -n zfsdisk2 4ZFS /dev/sda8
  Logical volume "zfsdisk2" created
lvm> lvcreate -C y   -p rw -L 1G -n zfsdisk3 4ZFS /dev/sda8
  Logical volume "zfsdisk3" created
lvm> lvcreate -C y   -p rw -L 1G -n zfsdisk4 4ZFS /dev/sda8
  Logical volume "zfsdisk4" created
lvm> lvcreate -C y   -p rw -L 1G -n zfsdisk5 4ZFS /dev/sda8
  Logical volume "zfsdisk5" created
lvm> lvdisplay -C
  LV      VG   Attr   LSize Origin Snap%  Move Log Copy%  Convert
  zfsdisk1 4ZFS -wc-a- 1.00G
  zfsdisk2 4ZFS -wc-a- 1.00G
  zfsdisk3 4ZFS -wc-a- 1.00G
  zfsdisk4 4ZFS -wc-a- 1.00G
  zfsdisk5 4ZFS -wc-a- 1.00G
```

Sada imamo 5 volumena koji su što se tiče operativnog sustava “obični” blok uređaji koje možemo koristiti kako želimo, npr. možemo na njima napraviti file systeme i priključiti ih na sustav (mount) ili ih koristiti kao diskove unutar linux softverskog raid polja ili u našem slučaju kao neovisne hard diskove za ZFS.
Umjesto “diskom” bolje bi služiti se izrazom blok uređaj, ovim primjerom u kojem koristim lvm između ostalog vidi se da zfs možemo iskoristiti sa bilo kojom tehnologijom za spremanje podataka.
Trenutno je korištenje ZFS prednosti na windows operativnim sustavima svedeno na pristupanje windows (smb/cifs) shareovima koji su posluženi sa solaris ili linux (samba) servera, no ako ZFS licenca bude open source postojati će mogućnost da se razvije driver za windows kao što recimo postoji za ext2/3 file system. No mislim da je najbolja opcija open solaris samba server sa zfs-om kao član windows domene ili linux.

ZFS
Osnovna struktura u ZFS-u je disk pool koji se sastoji od fizičkih uređaja npr. hard diskova ili njihovih particija, u ovom slučaju koristimo LVM volumene koje sam prethodno napravio.
Dakle konačno ZFS:

kreiramo ZFS pool naziva zfs_mirror a sastoji se od 2 “diska” zfsdisk1 i zfsdisk2 koji su u mirror polju.

```
root@inf0011l:~# zpool create zfs_mirror  mirror /dev/4ZFS/zfsdisk1
/dev/4ZFS/zfsdisk2
```

Provjera:

```
root@inf0011l:~# zfs list
NAME         USED  AVAIL  REFER  MOUNTPOINT
zfs_mirror    69K   984M    18K  /zfs_mirror
```

Direktorij /zfs_mirror nam je već dostupan za spremanje datoteka u našem / filesystemu.

ZFS struktura “ispod” zfs poola je dataset, kreiramo npr .dva dataseta u zfs poolu koji smo maloprije napravili.

```
root@inf0011l:~# zfs create zfs_mirror/dataset1
root@inf0011l:~# zfs create zfs_mirror/dataset2
```

Naredbom df na linuxu ispisujemo podatke o datotečnim sistemima na sustavu.

```
root@inf0011l:/zfs_mirror/dataset1# df -h |grep zfs

zfs_mirror            984M   21K  984M   1% /zfs_mirror
zfs_mirror/dataset1   984M   18K  984M   1% /zfs_mirror/dataset1
zfs_mirror/dataset2   984M   18K  984M   1% /zfs_mirror/dataset2
```

Zanimljivo je da OS vidi sve 3 točke (pool i 2 dataseta) kao 3 sistema i svaki veličine ~1 Gb što bi se moglo pogrešno protumačiti kao 3 x 1Gb iako smo definirali zfs pool tipa mirror a svaki dodjeljeni disk je bio veličine 1Gb.

Pokušajmo snimiti datoteku veličine ~500 M u dataset1

```
dd if=/dev/urandom of=/zfs_mirror/dataset1/random500M.file bs=500M count=1
1+0 records in
1+0 records out
524288000 bytes (524 MB) copied, 135.572 s, 3.9 MB/s
```

Ispis podataka o datotečnim sistema na sustavu:

```
root@inf0011l:/zfs_mirror/dataset1# df -h |grep zfs
zfs_mirror            494M   21K  494M   1% /zfs_mirror
zfs_mirror/dataset1   984M  491M  494M  50% /zfs_mirror/dataset1
zfs_mirror/dataset2   494M   18K  494M   1% /zfs_mirror/dataset2
```

Nije uobičajeno da se veličina sistema mjenja dinamički, ali to se upravo dogodilo, nakon što smo snimili file u dataset1 veličina ostala 2 file systema u tom zfs poolu se smanjila za veličinu datoteke koju smo upravili snimili što je logično jer trošimo prostor iz poola.

Snimimo nešto manje datoteke u /zfs_mirror/dataset2 i u /zfs_mirror.

```
dd if=/dev/urandom of=/zfs_mirror/dataset2/random50M.file bs=50M count=1
zfs_mirror            434M   21K  434M   1% /zfs_mirror
zfs_mirror/dataset1   934M  501M  434M  54% /zfs_mirror/dataset1
zfs_mirror/dataset2   484M   51M  434M  11% /zfs_mirror/dataset2

dd if=/dev/urandom of=/zfs_mirror/random50M.file bs=50M count=1
zfs_mirror            434M   51M  384M  12% /zfs_mirror
zfs_mirror/dataset1   884M  501M  384M  57% /zfs_mirror/dataset1
zfs_mirror/dataset2   434M   51M  384M  12% /zfs_mirror/dataset2
```

Dakle pomoću standardnih alate za prikazivanje zauzetosti file systema ne možemo saznati koliko imamo ukupno prostora, poznato nam je jedino koliko još možemo zauzeti prostora u

file systemu ali samo ako ne pišemo u druge za to vrijeme. Ukupnu veličinu možemo dobiti tako da veličini najvećeg file systema pridodamo zauzetost ostalih, tj. u ovom slučaju 884 + 51 +51 = 986, ukupna veličina je 984 ali zbog zaokruživanja na megabajt kod zbrajanje više zaokruženih vrijednosti dolazi do greške.

ZFS naredbama dobivamo jasniji rezultat, jer one za razliku od sistemske df "znaju" za hijerarhiju pool-dataset

```
root@inf00111:~# zpool list
NAME         SIZE   USED  AVAIL    CAP  HEALTH  ALTROOT
zfs_mirror  1016M   601M   415M    59%  ONLINE  -
```

Razlika između 984 do 1016 odlazi na podatke o filesystemu i rezervirani prostor.

```
root@inf00111:~# zfs list
NAME                  USED  AVAIL  REFER  MOUNTPOINT
zfs_mirror            601M   383M  50.1M  /zfs_mirror
zfs_mirror/dataset1   500M   383M   500M  /zfs_mirror/dataset1
zfs_mirror/dataset2  50.1M   383M  50.1M  /zfs_mirror/dataset2
```

Sve 3 datoteke su dakle smještene unutar zfs pool-a koji se sastoji od 2 hard diska, zfs se pobrinuo da su svi podaci snimljeni na oba diska (mirror), neki bi ovo nazvali softverskim raidom. Danas su sve manje razlike u perfomansama između softveskog raida (raid softver je unutar os-a) i hardverskog (raid softver je na raid kontroleru).

```
root@inf00111:~# zpool status
  pool: zfs_mirror
 state: ONLINE
 scrub: none requested
config:

      NAME                STATE     READ WRITE CKSUM
      zfs_mirror          ONLINE       0     0     0
        mirror            ONLINE       0     0     0
          4ZFS/zfsdisk1   ONLINE       0     0     0
          4ZFS/zfsdisk2   ONLINE       0     0     0

errors: No known data errors
```

Pogledajmo dalje neke mogućnosti zfs-a i što smo dobili sa datasetovima, odnosno zašto ih uopće koristimo jer datoteke možemo spremati u /zfs_mirror i prije nego što smo kreirali dataset. Datasetovi nam omogućuju da grupiramo datotečne sustave u poolove.

Kompresija

```
# zfs set compress=off zfs_mirror/dataset1

# zfs get compress
NAME                 PROPERTY     VALUE                SOURCE
zfs_mirror           compression  off                  local
zfs_mirror/dataset1  compression  off                  local
```

```
zfs_mirror/dataset2  compression  on                    local
```

U dataset 2 sam kopirao datoteku veličine 120 Mb, u datoteci se nalazila sql skripta (dump mysql baze). Omjer koji sam dobio sa defaultnom kompresijom (lzjb algoritam) za tekst je 2.24. No na to možemo utjecati.

```
# zfs get compressratio zfs_mirror/dataset2
NAME                 PROPERTY       VALUE                SOURCE
zfs_mirror/dataset2  compressratio  2.24x                -
```

Promjeniti ću kompresiju sa lzjb koji je optimiziran za brzinu u gzip-9 radi demonstracije i testiranja.

```
# zfs set compression=gzip-9 zfs_mirror/dataset2
# zfs get compressratio zfs_mirror/dataset2
```

Nakon brisanja i ponovnog snimanja datoteke na dataset2 omjer kompresije podignut. Promjena kompresija odnosi se samo na datoteke koje se snime nakon promjene, znači sve što postoji na datasetu neće se promjeniti i ne možemo komprimirati fs nakon što vidimo da nam ponestaje mjesta, no možemo napraviti komprimirani ds i prebaciti datoteke na njega, uključiti kompresiju na izvornom ds-u te vratiti datoteke na njega i to bez potrebe za dodatnim disk prostorom jer koristili bi naredbu move a unutar zfs poola svi datasetovi djele prostor dakle move bi mogao zapeti eventualnu ako imamo veliku datoteku za čiju kopiju nema mjesta unutar zfs poola.

```
NAME                 PROPERTY       VALUE                SOURCE
zfs_mirror/dataset2  compressratio  4.76x                -
```

Pogledajmo kako ZFS koristimo za snapshote file systema.

Trenutno je u dataset2 spremljena samo 120Mb velika sql skripta.
Učinimo snapshot direktorij vidljiv kod ispisa sadržaja direktorija i napravimo snapshot

```
# zfs set snapdir=visible zfs_mirror/dataset2
# zfs snapshot zfs_mirror/dataset2@moj_snapshot
```

Snapshot je gotov istog trenutka, a sam snapshot ne zauzima prostor na disku sve do trenutka dok se u dataset2 ne počne mjenjati odnosno dok u njega ne počnemo spremati nove datoteke, u osnovi snapshot zauzima onoliko više koliko je veća razlika od trenutnog stanja do stanja u trenutku uzimanja snapshota dataseta.
Pogledajmo ispis:

```
# zfs list
NAME                        USED  AVAIL  REFER  MOUNTPOINT
zfs_mirror                  726M   258M    21K  /zfs_mirror
zfs_mirror/dataset1         500M   258M   500M  /zfs_mirror/dataset1
zfs_mirror/dataset2         225M   258M   225M  /zfs_mirror/dataset2
zfs_mirror/dataset2@vlado    17K      -   175M  -
```

Parametar `snapdir=visible` omogućuje da se na vrhu filesystema čiji smo snapshoot napravili pojavi “skriveni” .zfs direktorij kroz koji je moguće doći do datoteka, pristup datoteka u snapshotu je read only. Iz nekog razloga na mom računalu se ne pojavljuje spomenuti skriveni direktorij. No to ne znači da ne mogu imati koristi od snapshot opcije.

Krenimo na mogućnost **kloniranja**.

Klonirati ne možemo dataset direktno, nego njegov snapshot. Obje operaciju su izuzetno kratke (instant) i ne utječu na perfomanse sustava jer svi podaci su već ionako na disku, jedino je bitno razdvojiti nove podatke u originalnom datasetu tj. snaphotu i kloniranom a o tome će zfs voditi računa.

```
# zfs clone zfs_mirror/dataset2@vlado zfs_mirror/dataset2clone
```

Klon je automatski mountan pod zfs poolom, u našem slučaju zfs pool je zfs_mirror.

```
# zfs list
NAME                         USED  AVAIL  REFER  MOUNTPOINT
zfs_mirror                   726M   258M    22K  /zfs_mirror
zfs_mirror/dataset1          500M   258M   500M  /zfs_mirror/dataset1
zfs_mirror/dataset2          225M   258M   225M  /zfs_mirror/dataset2
zfs_mirror/dataset2@vlado     17K      -   175M  -
zfs_mirror/dataset2clone        0   258M   175M  /zfs_mirror/dataset2clone
```

Bitno je primjetiti da je svim datotečnim sistemima slobodan prostor jednak tj. 285M te ću nakon snimanja datoteke od oko 200 Mb ostati bez prostora na svima tj. bez slobodnog prostora u zfs poolu.
S obzirom da je na klonove moguće pisati možemo npr. klonirati čitav datotečni sustav i ponovo pokrenuti sustav sa kloniranog napraviti testne instalacije softvera te po potrebi ostati na kloniranom ili se vratiti na original. Naravno realnije je klonirati samo jedan dio datotečnog sustava koji koristi operativni sustav ili aplikacija.

Potrošimo prostor i pogledajmo što možemo učiniti da bi proširili zfs pool.

```
# dd if=/dev/urandom of=/zfs_mirror/dataset2clone/250Mrandom2.file bs=10M count=25
# zfs list
NAME                         USED  AVAIL  REFER  MOUNTPOINT
zfs_mirror                   976M  7.83M    22K  /zfs_mirror
zfs_mirror/dataset1          500M  7.83M   500M  /zfs_mirror/dataset1
zfs_mirror/dataset2          225M  7.83M   225M  /zfs_mirror/dataset2
zfs_mirror/dataset2@vlado     17K      -   175M  -
zfs_mirror/dataset2clone     250M  7.83M   425M  /zfs_mirror/dataset2clone
```

Preostalo nam je niti 8Mb i moramo povećati pool. Dvije naredbe proširuju pool tj. omogućuju nam dodavanje blok uređaja u pool.

`zpool attach` i `zpool add`

Obje dodaju disk ali bitno razlika je što s attach spajamo device na device, npr ako

upotrijebim add na ovu našu situaciju dobiti ću mirror od 3 diska no to ne želimo jer želimo povećanje prostora a ne sigurnosti. Zpool add dodaje u pool uređaj koji se može ali i ne mora sastojati od n uređaja a ti uređaji mogu tvoriti neku raid kombinaciju ili ne. To nam daje slobodu da u ovo naše mirror polje jednostavno dodamo 1 disk uređaj za koji znamo da je sa raid1 polja s ugrađenog hardverskog kontrolera, isto tako nam daje slobodu da degradiramo redundanciju što ću upravo napraviti dodavanjem samo 1 lvm disk uređaja u zfs pool :

```
# zpool add -f zfs_mirror /dev/4ZFS/zfsdisk3

# zfs list
NAME                        USED  AVAIL  REFER  MOUNTPOINT
zfs_mirror                  976M  1024M    22K  /zfs_mirror
zfs_mirror/dataset1         500M  1024M   500M  /zfs_mirror/dataset1
zfs_mirror/dataset2         225M  1024M   225M  /zfs_mirror/dataset2
zfs_mirror/dataset2@vlado    17K      -   175M  -
zfs_mirror/dataset2clone    250M  1024M   425M  /zfs_mirror/dataset2clone
```

Dakle dodajući 1 disk veličine ~ 1Gb u zfs pool koji sam nazvao zfs_mirror dobio sam upravo toliko slobodnog prostora više, znači definitivno taj zadnji 1Gb nije zaštićen kao prvi Gb za koja sam potrošio 2 diska. Nazvati pool mirror je očito bila greška jer on može sadržavati i druge kombinacije, ali to što nam zfs to dozvoljava je prednost, jer mi možemo kombinirati bilo kakve uređaje sa bilo kojeg izvora, nešto može biti lokalno nešto sa SAN/NAS infrastrukture.
Disk uređaj spojen na os sa data storagea može imati neku vrstu redundancije o kojoj zfs ne mora brinuti, ali on to može ako mi želimo.

Ispravan način za dodavanje prostora i zadržavanje redundancije :

zpool add zfs_mirror **mirror** /dev/4ZFS/zfsdisk4 /dev/4ZFS/zfsdisk5

Da bi zadržali nivo redundancije nakon dodavanja 1 diska (/dev/4ZFS/zfsdisk3) kreirati ću još jedan logički volumen zfsdisk6 koji ću pridružiti volumenu zfsdisk3 čime će zfs automatski početi s dvostrukim zapisavanjem na oba diska dok će kapacitet zfs pool ostati nepromjenjen.
lvm> lvcreate -C y   -p rw -L 1G -n zfsdisk6 4ZFS /dev/sda8
  Logical volume "zfsdisk6" created

zpool attach zfs_mirror /dev/4ZFS/**zfsdisk3** /dev/4ZFS/**zfsdisk6**
zfs list
NAME   USED AVAIL REFER MOUNTPOINT
zfs_mirror   976M 1.98G   22K /zfs_mirror
zfs_mirror/dataset1   500M 1.98G   500M /zfs_mirror/dataset1
zfs_mirror/dataset2   225M 1.98G   225M /zfs_mirror/dataset2
zfs_mirror/dataset2@vlado   17K   -   175M -
zfs_mirror/dataset2clone   250M 1.98G   425M /zfs_mirror/dataset2clone

Sve ove operacije se rade online tj. one ne ometaju rad aplikacija ili operativnog sustava, a administratoru sustava olakšavaju proširivanje diskova i backup. Pojačana sigurnost može se dobiti ako se svakom korisniku da zaseban file system, što neupućenom može služiti kao veliki zahvat no u biti se to u zfs svede na kreiranje ZFS poola za sve korisnike dok svaki

korisnik ima svoj dataset. ZFS poznaje kvote i omogućuju replikacije.
U stanju je detektirati greške i oktloniti ih a u svakom slučaju ima daleko više primjena i mogućnosti nego što sam ih stigao isprobati, konkurentni file system mu je BTRFS ali on je tek u razvoju.
Koliko vidim glavna prednost ovoga filesystema je u tome što nudi sve u jednom, dobijete disk managemant,volume managmant,snapshot,kloniranje,enkripciju,softverski raid i navodno izvrsne perfomanse zbog izvrsnih algoritama a time upravljate kroz jedno sučelje (zfs i zpool).
U svakom slučaju ZFS djeluje ozbiljno i razrađeno. Problem sa nepojavljivanjem linka za .zfs snapshot folder pripisujem linux user-space varijanti.

Kratak pregled zfs komandi.

Literatura:

1.Man stranice zpool i zfs
2.Wikipedia
3.Predavanje "Solaris zamjena za Linux" – prisustvovao u Srcu