

Administrasi ZFS Solaris

Agus Setiawan

august.kerenz@gmail.com

<http://www.agussetiawan.net>

Lisensi Dokumen:

Copyright © 2003-2007 IlmuKomputer.Com

Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.

Pada tulisan ini akan membahas tentang penggunaan ZFS di Solaris maupun OpenSolaris.

1. Pengertian ZFS

Filesistem ZFS adalah filesistem baru revolusioner yang secara fundamental mengubah bagaimana meng-administrasi suatu filesistem, dengan fitur dan keuntungan yang tidak ditemukan pada filesistem manapun saat ini. ZFS di desain untuk menjadi filesistem yang scalable, mudah di administrasi dan bersifat robust.

- ZFS Storage Pool

ZFS menggunakan konsep 'pool storage' untuk me-manage storage fisik. Secara histori, filesistem dibangun diatas single device fisik. Untuk pengalamatan berbagai device dan menyediakan redundansi data dikenalkan konsep 'volume manager' yang menyediakan image pada single device. ZFS mengeliminasi penggunaan manajemen volume secara bersama-sama, maka dibuatlah virtualisasi volume yang kemudian disatukan menjadi storage pool. Storage pool ini memiliki karakteristik fisik misal layout device, redundansi data, dll. Ketika sebuah storage baru ditambahkan pada storage pool, maka bisa langsung digunakan tanpa melakukan pengaturan yang lebih banyak. Pada berbagai kondisi storage pool bisa dibuat virtual memori tambahan pada system.

- Transactional Semantics

ZFS merupakan filesistem yang bersifat transaksional, yang artinya filesistem selalu konsisten pada disk. Pada filesistem tradisional akan menuliskan data pada tempat yang sama, sehingga jika pada saat proses *write* terjadi kerusakan tiba-tiba atau mati lampu, maka filesistem menjadi Inconsistency (tidak konsisten). Untuk menyelesaikan problem

ini dapat menggunakan perintah *fsck* yang akan memperbaiki filesistem yang tidak konsisten. Solusi lain bisa menggunakan konsep *journaling* yaitu merekam proses yang dilakukan pada jurnal yang terpisah, dan jika terjadi crash pada system, proses yang di rekam tadi bisa ulang lagi / di replay berarti ada dua kali penulisan data yang sama.

Dengan filesistem yang bersifat transaksional ini data di atur dengan model *copy-on-write*, dimana penulisan data di lakukan pada blok yang baru, setelah penulisan benar-benar selesai dan sukses, baru kemudian pointernya di update dan blok yang lama dianggap kosong dan bisa ditulis buat file yang lain. Jika ada gangguan selama penulisan, maka data lama yang digunakan. Sehingga pada ZFS ini tidak ada penulisan data dua kali (overwrite) atau corrupt pada saat system crash. Jadi, ZFS tidak membutuhkan perintah *fsck*.

- **Checksums and Self-Healing**

ZFS mendukung pool storage dengan berbagai level redundansi data, termasuk mirroring dan RAID-5. Ketika ZFS mendeteksi adanya data blok yang rusak, ZFS akan mengambilkan copy-annya dari hasil redundansi yang lain, memperbaiki data dan menggantikannya dengan hasil copy-an yang bagus.

- **Scalability**

ZFS di desain dengan tingkat skalabilitas yang tinggi.

- a. satu pool ZFS maksimum bisa nampung total data sebesar 128bit. sekitar 18 milyar milyar kali lebih besar dari 64bit filesystem yang ada sekarang.
- b. maksimum 2^{64} ZFS snapshot
- c. maksimum 2^{48} entries dalam sebuah directory
- d. maksimum bisa membuat single file sebesar 16 EiB (baca: ExbiByte)
1 exbibyte = 260 bytes = 1,152,921,504,606,846,976 bytes = 1,024 pebibytes
- e. maksimum bisa memiliki file attribute sebesar 16 EiB
- f. maksimum bisa memiliki pool sebesar 256 ZiB (ZebiByte)
1 zebibyte = 270 bytes = 1,180,591,620,717,411,303,424 bytes = 1,024 exbibytes
- g. maksimum memiliki 2^{56} attributes dalam tiap file
- h. maksimum memiliki 2^{56} file dalam sebuah directory
- i. maksimum memiliki 2^{64} buah devices dalam sebuah zpool
- j. maksimum memiliki 2^{64} zpool salam sebuah system
- k. maksimum memiliki 2^{64} filesystem dalam sebuah zpool

- **ZFS Snapshots**

Snapshots adalah hasil copy dari filesistem atau volume yang sifatnya read-only. Proses snapshots bisa dibuat dengan waktu yang singkat dan mudah. Snapshots mengkonsumsi space yang tidak terlalu besar.

- **Simple Administration**

ZFS merupakan filesistem yang mudah untuk di administrasi. Proses administrasi menggunakan 2 perintah yaitu **zpool** dan **zfs**. Masing-masing perintah tersebut mempunyai sub-command, options, dan argument. Perubahan pada ZFS langsung bisa dilihat hasilnya tanpa harus melakukan reboot atau restart.

2. Kapasitas File Sistem

Berikut table kapasitas file system yang sering digunakan oleh Unix/Linux user :

File System	Max File Size	Max Volume Size
UFS	up to 32 petabytes	up to 1 yottabyte
ext3	up to 2 terabytes	up to 32 terabytes
ext4	up to 16 terabytes	up to 1 exabytes
FAT32	4 gigabytes	up to 8 terabytes
NTFS	16 exabytes	16 exabytes
ZFS	16 exabytes	2^{18} exabytes

Catatan:

Max File Size : ukuran maksimum sebuah file yang bisa di buat dengan filesistem

Max Volume Size : ukuran maksimum sebuah volume / disk / storage untuk menyimpan filesistem

Standar prefix decimal yang umumnya digunakan untuk mendeskripsikan kuantitas dari storage / media penyimpanan :

10^3	Kilo	Thousands
10^6	Mega	Millions
10^9	Giga	Billions
10^{12}	Tera	Trillions
10^{15}	Peta	Quadrillions
10^{18}	Exa	Quintillions
10^{21}	Zetta	Sextillions
10^{24}	Yotta	Septillions

3. Praktik ZFS

Ada dua command / perintah yang digunakan untuk manage storage dengan ZFS yaitu pertama, membuat pool dan mengatur karakteristiknya menggunakan command `/usr/sbin/zpool`, kedua, mengatur filesistem pada pool-nya menggunakan command `/usr/bin/zfs`.

Sebelum simulasi, penulis cek dulu box yang digunakan untuk praktik administrasi ZFS di bawah ini :

login as: agus

Using keyboard-interactive authentication.

Password:

Last login: Thu Jul 23 22:55:57 2009

Sun Microsystems Inc. SunOS 5.11 snv_111b November 2008

agus@opensolaris:~\$ pfexec su -

Sun Microsystems Inc. SunOS 5.11 snv_111b November 2008

You have new mail.

23:10:13 root@opensolaris:~

Box-nya penulis remote via putty, jadi ga langsung di box OpenSolaris-nya.

Pertama-tama kita buat dulu device storage disknya menggunakan perintah mkfile, misalkan kita mau bikin 5 storage disk. Anggap saja storage disk virtual yang kita buat ini seperti layaknya sebuah hardisk nyata, jadi biar enak ngebayanginnya. Tapi sebelum buat, kita cek dulu kapasitas disk nyata yang sisa berapa, biar nanti kita bisa tentuin ukuran masing-masing storage disk virtualnya, lakukan dengan perintah :

df -h

Filesystem Size Used Avail Use% Mounted on

rpool/ROOT/opensolaris

5.7G 4.2G 1.5G 74% /

Masih sisa lumayan banyak, 1,5 GB. Partisi OpenSolaris ini cuman dikasih 5,7 GB, baru kepakai 74%. OK, kita lanjutkan saja dengan pembuatan storage disk virtual dengan perintah mkfile.

mkfile 200m /vdisk1

mkfile 200m /vdisk2

mkfile 200m /vdisk3

mkfile 200m /vdisk4

mkfile 200m /vdisk5

200m artinya ukuran virtual disk-nya 200MB. kalo g=Giga. vdisk adalah nama yang penulis berikan dengan nomor urutan 1 sampai 5. Sampai disini, kita sudah bisa buat virtual disknya. Dari virtual disk inilah kita nanti akan mencoba mempraktikan fungsi-fungsi ZFS.

Kita cek dulu, kalau berhasil, maka tampilan 5 vdisk tersebut akan seperti ini :

```
# ls -l /  
  
total 1024362  
  
..  
  
-rw——T  1 root root 209715200 2009-07-23 23:30 vdisk1  
-rw——T  1 root root 209715200 2009-07-23 23:30 vdisk2  
-rw——T  1 root root 209715200 2009-07-23 23:35 vdisk3  
-rw——T  1 root root 209715200 2009-07-23 23:35 vdisk4  
-rw——T  1 root root 209715200 2009-07-23 23:36 vdisk5
```

a. Bikin disk stripping / RAID 0

Kita menggabungkan lebih dari satu disk untuk dijadikan satu pool dengan ukuran yang besar, ukuran pool sesuai dengan jumlah dari ukuran masing-masing disk. Kelemahan dari RAID 0 ini tidak adanya redundancy, jadi kalau salah satu disk rusak, maka data akan yang hilang dan ga bisa di akses

Biar nanti ke depan tidak kesulitan memahami tulisan ini, pembaca perlu mengerti apa itu pool storage? yaitu pengorganisasian dari beberapa device fisik seperti hardisk / storage lainnya menjadi satu logical pool.

Cara membuatnya seperti ini :

```
# zpool create poolnol /vdisk1 /vdisk2 /vdisk3 /vdisk4
```

poolnol adalah nama untuk pool dengan model RAID 0, terdiri dari vdisk1, vdisk2, vdisk3, vdisk4. Berarti ukuran poolnol akan sama dengan jumlah vdisk1, vdisk2, vdisk3, vdisk4. Cara ngecek pool yang sudah di bikin.

```
# zpool list  
  
NAME      SIZE  USED  AVAIL  CAP  HEALTH  ALTROOT  
  
..
```

```
poolnol 780M 77.5K 780M 0% ONLINE -
```

```
..
```

Ukuran poolnol 780MB berarti sama dengan jumlah vdisk1+vdisk2+vdisk3+vdisk4 yang masing-masing vdisk ukurannya 200MB. Memang tidak mutlak sama, akan tetapi mendekati. Kita liat statusnya :

```
# zpool status
```

```
pool: poolnol
```

```
state: ONLINE
```

```
scrub: none requested
```

```
config:
```

```
NAME      STATE      READ WRITE CKSUM
```

```
poolnol   ONLINE     0   0   0
```

```
/vdisk1   ONLINE     0   0   0
```

```
/vdisk2   ONLINE     0   0   0
```

```
/vdisk3   ONLINE     0   0   0
```

```
/vdisk4   ONLINE     0   0   0
```

```
errors: No known data errors
```

Kemudian, hasil mount otomatisnya, bisa kita liat juga..ini yang membedakan dengan di sistem operasi lain, klo di OpenSolaris sudah otomatis terbentuk mount pointnya, jadi ga perlu edit2 file /etc/fstab atau /etc/mnttab.

```
# mount
```

```
...
```

```
/poolnol on poolnol
```

```
read/write/setuid/devices/nonbmand/exec/xattr/atime/dev=2d9000c on Thu Jul 23  
23:54:09 2009
```

b. Bikin file sistem pake ZFS

Setelah itu, kita bikin filesistemnya, skenario-nya, poolnol yang terdiri dari 4 vdisk itu kita bagi untuk 6 user yaitu user A,B,C,D,E, dan F.

```
# zfs create poolnol/A
```

```
# zfs create poolnol/B
```

```
# zfs create poolnol/C
```

```
# zfs create poolnol/D
```

```
# zfs create poolnol/E
```

```
# zfs create poolnol/F
```

Cek hasil mounting-nya :

```
# mount
```

```
.....
```

```
/poolnol/A on poolnol/A
```

```
read/write/setuid/devices/nonbmand/exec/xattr/atime/dev=2d9000d on Fri Jul 24  
05:52:14 2009
```

```
/poolnol/B on poolnol/B
```

```
read/write/setuid/devices/nonbmand/exec/xattr/atime/dev=2d9000e on Fri Jul 24  
05:52:35 2009
```

```
/poolnol/C on poolnol/C
```

```
read/write/setuid/devices/nonbmand/exec/xattr/atime/dev=2d9000f on Fri Jul 24  
05:52:40 2009
```

```
/poolnol/D on poolnol/D
```

```
read/write/setuid/devices/nonbmand/exec/xattr/atime/dev=2d90010 on Fri Jul 24  
05:52:48 2009
```

```
/poolnol/E on poolnol/E
```

```
read/write/setuid/devices/nonbmand/exec/xattr/atime/dev=2d90011 on Fri Jul 24  
05:52:51 2009
```

```
/poolnol/F on poolnol/F
```

```
read/write/setuid/devices/nonbmand/exec/xattr/atime/dev=2d90012 on Fri Jul 24  
05:52:53 2009
```

Konsep pool storage yang dibagi-bagi untuk beberapa user / aplikasi, bisa dilihat sebagai berikut :

```
# zfs list | grep "poolnol"

poolnol          240K  748M  27K /poolnol
poolnol/A        19K   748M  19K /poolnol/A
poolnol/B        19K   748M  19K /poolnol/B
poolnol/C        19K   748M  19K /poolnol/C
poolnol/D        19K   748M  19K /poolnol/D
poolnol/E        19K   748M  19K /poolnol/E
poolnol/F        19K   748M  19K /poolnol/F
```

Dari ZFS diatas, penggunaan poolnol-nya masih bersifat share, artinya sapa yang butuh banyak, maka nanti akan diberikan sesuai kebutuhan disk-nya. Misal, kita mau bikin file sebesar 50m di poolnol/A dengan nama transformer, kemudian di poolnol/E, sebesar 10m, nama filenya diehard :

```
# mkfile 50m /poolnol/A/transformer
# mkfile 10m /poolnol/E/diehard
```

Kita cek file-nya :

```
# ls -l /poolnol/A/

total 51215
-rw——T 1 root root 52428800 2009-07-24 06:09 transformer

# ls -l /poolnol/E/

total 10244
-rw——T 1 root root 10485760 2009-07-24 06:50 diehard
```

Cek status poolnol :

```
# zfs list | grep "poolnol"

poolnol          60.4M  688M  27K /poolnol
poolnol/A        50.0M  688M  50.0M /poolnol/A
```



```
poolnol/B      19K 688M 19K /poolnol/B
poolnol/C      19K 688M 19K /poolnol/C
poolnol/D      19K 688M 19K /poolnol/D
poolnol/E     10.0M 688M 10.0M /poolnol/E
poolnol/F      19K 688M 19K /poolnol/F
```

Coba kamu liat pada poolnol/A dan poolnol/E, disk yang digunakan sebesar 50MB dan 10MB.

Sukses :D

Kita lanjutkan lagi, sekarang kita tambah poolnol dengan vdisk5, kalo kamu liat diatas pada zpool status, maka poolnol baru terdiri dari 4 disk, sekarang kita tambah 1 lagi biar jadi 5.

Sebelum ditambah :

```
# zpool status poolnol

pool: poolnol

state: ONLINE

scrub: none requested

config:
```

```
NAME      STATE    READ WRITE CKSUM
poolnol   ONLINE    0   0   0
/vdisk1   ONLINE    0   0   0
/vdisk2   ONLINE    0   0   0
/vdisk3   ONLINE    0   0   0
/vdisk4   ONLINE    0   0   0
```

```
errors: No known data errors
```

Setelah ditambah 1 vdisk5:

```
# zpool add poolnol /vdisk5

# zpool status poolnol

pool: poolnol

state: ONLINE

scrub: none requested

config:

NAME      STATE      READ WRITE CKSUM
poolnol   ONLINE    0   0   0
/vdisk1   ONLINE    0   0   0
/vdisk2   ONLINE    0   0   0
/vdisk3   ONLINE    0   0   0
/vdisk4   ONLINE    0   0   0
/vdisk5   ONLINE    0   0   0

errors: No known data errors
```

Pengaruhnya, nanti ukuran poolnol akan jadi lebih besar, penambahannya sesuai dengan ukuran vdisk5 :

```
# zfs list | grep "poolnol"

poolnol          60.4M  883M  27K /poolnol
poolnol/A         50.0M  883M  50.0M /poolnol/A
poolnol/B         19K    883M  19K /poolnol/B
poolnol/C         19K    883M  19K /poolnol/C
poolnol/D         19K    883M  19K /poolnol/D
poolnol/E        10.0M  883M  10.0M /poolnol/E
poolnol/F         19K    883M  19K /poolnol/F
```

c. ZFS Snapshots

Sekarang, kita akan mencoba mempraktikkan ZFS snapshots via command line. Salah satu implementasi ZFS snapshot via GUI yaitu Time Slider.

Misalkan sekarang kita pindah direktori ke /poolnol/A :

```
# cd /poolnol/A/
```

```
# pwd
```

```
/poolnol/A
```

Bikin file dengan nama agus dan setiawan :D

```
# touch agus
```

```
# touch setiawan
```

Penulis coba bikin snapshot untuk poolnol/A :

```
# zfs snapshot poolnol/A@rabumalam
```

poolnol/A adalah filesistem ZFS-nya kemudian rabumalam adalah nama snapshotnya.

Aku bikin lagi dua file lagi dengan nama “bandung”, kemudian aku lakukan snapshot lagi, nama snap-nya kamismalam, bikin file lagi namanya “cool”, di snapshot lagi namanya jumatmalam

```
# touch bandung
```

```
# zfs snapshot poolnol/A@kamismalam
```

```
# touch cool
```

```
# zfs snapshot poolnol/A@jumatmalam
```

Kita hapus “agus” dan bikin file lagi namanya “dago” dan lakukan snapshot lagi, nama snap-nya sabtumalam

```
# rm agus
```

```
# touch dago
```

```
# zfs snapshot poolnol/A@sabtumalam
```

Jadinya, malam minggu di dago nih :D

Kita liat file apa aja yang sudah kita bikin :

```
# ls -l

total 51217

-rw-r--r-- 1 root root      0 2009-07-24 07:30 bandung
-rw-r--r-- 1 root root      0 2009-07-24 07:32 cool
-rw-r--r-- 1 root root      0 2009-07-24 07:34 dago
-rw-r--r-- 1 root root      0 2009-07-24 07:24 setiawan
-rw-----T 1 root root 52428800 2009-07-24 06:09 transformer
```

Nah, sekarang lakukan penghapusan semua file dengan perintah rm :

```
# rm bandung cool dago setiawan transformer
```

Jadinya hilang semua alias kosong, :(

```
# ls -l

total 0
```

Tapi, kamu tenang aja..karena kita sekarang lagi praktik ZFS Snapshot, jadi file yang tadi kita delete bisa kita balikin lagi.

```
# cd .zfs

# ls

shares snapshot

# cd snapshot/

# ls -l

total 6

drwxr-xr-x 2 root root 7 2009-07-24 07:32 jumatmalam
drwxr-xr-x 2 root root 6 2009-07-24 07:30 kamismalam
drwxr-xr-x 2 root root 5 2009-07-24 07:24 rabumalam
drwxr-xr-x 2 root root 7 2009-07-24 07:34 sabtumalam
```

Wew..snapshotnya sukses semua :D , mulai rabumalam sampai sabtumalam. Sekarang misalkan kita pengen balikin pada kondisi jumatmalam, tinggal jalanin perintah berikut

```
# cp -R jumatmalam/ /poolnol/A/
```

Kalo sudah, sekarang di cek, di /poolnol/A. Karena ZFS Snapshot bersifat ReadOnly, maka kita lakukan peng-copyan dari snapshot yang kita bikin.

```
# ls -l /poolnol/A/
```

```
total 2
```

```
drwxr-xr-x 2 root root 7 2009-07-24 07:44 jumatmalam
```

```
# ls -l /poolnol/A/jumatmalam/
```

```
total 51219
```

```
-rw-r--r- 1 root root      0 2009-07-24 07:44 agus
```

```
-rw-r--r- 1 root root      0 2009-07-24 07:44 bandung
```

```
-rw-r--r- 1 root root      0 2009-07-24 07:44 cool
```

```
-rw-r--r- 1 root root      0 2009-07-24 07:44 setiawan
```

```
-rw----- 1 root root 52428800 2009-07-24 07:44 transformer
```

Proses snapshot sudah selesai dan data kita kembali dengan aman.

Nice work :D

d. Destroy storage pool

Karena kita sudah selesai maen2-nya dengan RAID 0, kita destroy atau hancurkan saja poolnol-nya, lalu kita bikin model RAID 1 atau mirroring. Pool eksisting punyaku :

```
# zpool list
```

```
NAME      SIZE  USED  AVAIL  CAP  HEALTH  ALTROOT
```

```
oasis    370M  143K  370M   0%  ONLINE  -
```

```
poolnol  975M  111M  864M  11%  ONLINE  -
```

```
rpool    6.44G  5.82G  632M  90%  ONLINE  -
```

Cara menghapusnya:

```
# zpool destroy poolnol
```

```
cannot unmount '/poolnol/A': Device busy
```

```
could not destroy 'poolnol': could not unmount datasets
```

wah..error nih :(,, coba kita kasih tambahan option -f yang artinya force atau paksa.

```
# zpool destroy -f poolnol
```

Done..akhirnya bisa juga di destroy

```
# zpool list
```

```
NAME    SIZE  USED  AVAIL  CAP  HEALTH  ALTROOT
```

```
oasis   370M  143K  370M   0%  ONLINE -
```

```
rpool   6.44G  5.82G  632M  90%  ONLINE -
```

e. Bikin pool model RAID 1

RAID 1 / mirroring : artinya kita bikin mirror disk dimana kalau disk yang satu rusak, maka masih bisa menggunakan disk yang lain. Minimal pembuatan disk mirroring ini terdiri dari 2 disk. Perintahnya ga jauh beda dengan yang sebelumnya, yang membedakan sub-command-nya.

Misal, kita bikin mirror disk, namanya “poolsatu” terdiri dari vdisk1 dan vdisk 5

```
# zpool create poolsatu mirror /vdisk1 /vdisk5
```

Cek hasilnya :

```
# zpool list
```

```
NAME    SIZE  USED  AVAIL  CAP  HEALTH  ALTROOT
```

```
oasis   370M  143K  370M   0%  ONLINE -
```

```
poolsatu 195M  76K  195M   0%  ONLINE -
```

```
rpool   6.44G  5.82G  632M  90%  ONLINE -
```

Cek statusnya :

```
# zpool status poolsatu
```

```
pool: poolsatu
```

state: ONLINE

scrub: none requested

config:

NAME	STATE	READ	WRITE	CKSUM
------	-------	------	-------	-------

<i>poolsatu</i>	<i>ONLINE</i>	<i>0</i>	<i>0</i>	<i>0</i>
-----------------	---------------	----------	----------	----------

<i>mirror</i>	<i>ONLINE</i>	<i>0</i>	<i>0</i>	<i>0</i>
---------------	---------------	----------	----------	----------

<i>/vdisk1</i>	<i>ONLINE</i>	<i>0</i>	<i>0</i>	<i>0</i>
----------------	---------------	----------	----------	----------

<i>/vdisk5</i>	<i>ONLINE</i>	<i>0</i>	<i>0</i>	<i>0</i>
----------------	---------------	----------	----------	----------

errors: No known data errors

Sekarang bikin filesistemnya pake ZFS :

```
# zfs create poolsatu/home
```

```
# zfs list | grep "poolsatu"
```

```
poolsatu          101K 163M 19K /poolsatu
```

```
poolsatu/home      19K 163M 19K /poolsatu/home
```

Bikin file di /poolsatu/home nama filenya "geulis" ukurannya 2 MB aja biar cepet..

```
# mkfile 2m /poolsatu/home/geulis
```

```
# ls -l /poolsatu/home/
```

```
total 1
```

```
-rw——T 1 root root 2097152 2009-07-24 08:01 geulis
```

Untuk membuktikan apakah vdisk1 benar-benar mirror ke vdisk5 atau sebaliknya, kita akan offline-kan vdisk 5 atau kita copot dengan perintah offline & detach. Dan kita liat status akhirnya :

```
# zpool offline poolsatu /vdisk1
```

```
# zpool status poolsatu
```

```
pool: poolsatu
```

```
state: DEGRADED
```

status: One or more devices has been taken offline by the administrator.

Sufficient replicas exist for the pool to continue functioning in a degraded state.

action: Online the device using 'zpool online' or replace the device with 'zpool replace'.

scrub: none requested

config:

NAME	STATE	READ	WRITE	CKSUM
------	-------	------	-------	-------

poolsatu	DEGRADED	0	0	0
----------	----------	---	---	---

mirror	DEGRADED	0	0	0
--------	----------	---	---	---

/vdisk1	OFFLINE	0	0	0
---------	---------	---	---	---

/vdisk5	ONLINE	0	0	0
---------	--------	---	---	---

zpool detach poolsatu /vdisk1

zpool status poolsatu

pool: poolsatu

state: ONLINE

scrub: none requested

config:

NAME	STATE	READ	WRITE	CKSUM
------	-------	------	-------	-------

poolsatu	ONLINE	0	0	0
----------	--------	---	---	---

/vdisk5	ONLINE	0	0	0
---------	--------	---	---	---

errors: No known data errors

Cek file “geulis” masih ada atau ga :

ls -l /poolsatu/home/

total 2051

-rw——T 1 root root 2097152 2009-07-24 08:01 geulis

:D si “geulis” masih ada tuh..perhatikan juga ukuran file-nya,, tetap 2MB..

Berarti Mirror disknya berjalan dan benar2 ga ngaruh kalau disk yang satu rusak. Karena masih ada disk satu-nya lagi yang akan menyimpan datanya.

e. Fitur Kompresi

Salah satu fitur yang menarik dari ZFS ini adalah fitur kompresi-nya. Kita bisa melakukan proses kompresi per masing-masing filesystem. Dengan melakukan kompresi ini, maka kita bisa mengurangi space hardisk, karena semua isi filesystemnya akan di kompres ukurannya yang misal awalnya 100MB akan menjadi 90MB. Dan proses kompresi ini bisa dilakukan on demand.

Untuk melakukannya, berikut langkah-langkahnya :

sebelum proses kompresi

```
# zfs list
```

NAME	USED	AVAIL	REFER	MOUNTPOINT
------	------	-------	-------	------------

bunker	76.8M	281M	76.6M	/bunker
--------	-------	------	-------	---------

```
# df -h bunker/
```

Filesystem	size	used	avail	capacity	Mounted on
------------	------	------	-------	----------	------------

bunker	358M	75M	283M	22%	/bunker
--------	------	-----	------	-----	---------

proses kompresi

```
# zfs set compression=on bunker
```

cek status

```
# zfs get compression bunker
```

NAME	PROPERTY	VALUE	SOURCE
------	----------	-------	--------

bunker	compression	on	local
--------	-------------	----	-------

cek compress ratio :

```
# zfs get compressratio bunker
```

NAME	PROPERTY	VALUE	SOURCE
------	----------	-------	--------

bunker compressratio 1.00x -

proses selesai..

Selamat mencoba..

Referensi

<http://www.sun.com/bigadmin>

<http://r4inbuw.blogspot.com>

Biografi Penulis



Agus Setiawan, Muslim, lahir di Kebumen, Jawa Tengah, 10 Agustus 1987. Saat ini sedang menyelesaikan skripsi di Institut Teknologi Telkom / IT Telkom Bandung jurusan Teknik Industri.

Aktifitas saat ini menjadi OpenSolaris Leader wilayah Bandung, Jawa Barat dan Asisten Dosen Jaringan Komputer di kampusnya.

Berpengalaman sebagai teknisi, lecture, trainer di lembaga training center dengan spesifikasi Unix, Linux dan Network. Punya cita-cita ingin menjadi unix/linux engineer yang expert di bidangnya. Sertifikasi IT yang dimilikinya yaitu *Sun Certified Solaris Associate*.

Informasi lebih lanjut mengenai penulis :

G: august.kerenz@gmail.com

Y: august.kerenz@yahoo.com

F : www.facebook.com/august.kerenz

B: <http://www.agussetiawan.net>