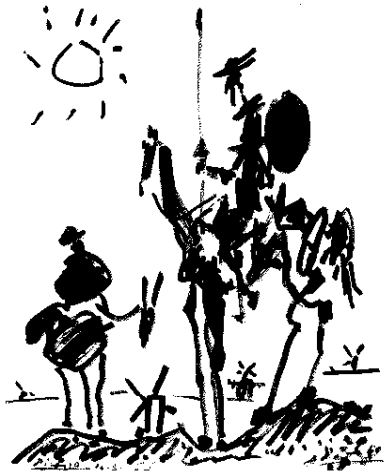
ZFS Zettabyte.

David Galán Ortiz.
www.opensolarisblog.org
dgalan@opensolarisblog.org

< Spain OpenSolaris Users Groups >

開
放
的
열린
مفتوح
libre
मुक्त
ముక్త
livre
libero
ముక్త
开放的
açık
open
nyílt
פתוח
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை

opensolaris
<OrangeBooks>



USE IMPROVE EVANGELIZE

LICENCIA....3

Referencias 3

ZFS ZETTABYTE....4

Introducción a ZFS....4
Crear un pool....6
Crear sistemas de ficheros ZFS....7
Propiedades de un ZFS....8
Creación cuotas y reserva de espacio....8
Sistema de ficheros comprimido....9
Sistema de ficheros de solo lectura....10
Cambiar el punto de montaje 10
Ver las propiedades de un ZFS....11
Gestión del POOL....12
Añadir nuevos discos al pool....12
Quitar discos al pool....13
Eliminar un pool....13
Reemplazar un disco del pool....13
Crear un mirror (RAID 1)....14
Crear un mirror desde un disco existente....14
Reemplazar un disco del mirror....15
Crear RAID-Z 16
Snapshots 17
Borrar snapshots o fotos....19
Estados de ZFS 20
Operaciones entrada y salida....23

Licencia

Referencias

ZFS Zettabyte

Introducción a ZFS

ZFS (Zettabyte File System) es el nuevo sistema de archivos incorporado a Solaris 10. Es un sistema de archivos de 128 bits y su límite de tamaño máximo es de 256 cuatrillones de zettabytes[1].

En la wikipedia se hace la siguiente referencia sobre la capacidad de ZFS:

"Como ejemplo de las capacidades expresadas por estos números, si un usuario crease 1000 ficheros por segundo, tardaría unos 9000 años en alcanzar el límite impuesto por el número de ficheros.".

En este capitulo pretendemos ver la parte práctica de ZFS, si deseas conocer los límites teóricos de ZFS puedes ver los valores de referencia en la wikipedia o en www.sun.com .

Las principales características de ZFS son:

- *Tamaño máximo* de 256 cuatrillones de zettabytes

[1] Información detallada sobre la unidad de medida en la wikipedia: http://es.wikipedia.org/wiki/Zettabyte

- *Administración sencilla por comandos o web.* (nos olvidamos de format, newfs, mount, vfstab, etc..)

- *Copy-on-write* (ZFS no sobrescribe los nuevos datos directamente, crea los datos en un nuevo bloque y posteriormente cambia los punteros de datos y realiza la escritura definitiva. Con este método siempre esta garantizada la integridad de los datos y no es necesario el uso de utilidades como *fsck*)

- *Snapshots* (capturas). Podemos sacar un "foto" de forma rápida a todo un sistema de ficheros. Podemos instalar un paquete en el sistema y si este no cumple nuestras expectativas podemos realizar un rollback para volver al estado anterior.

- *Comprensión*. Podemos definir un sistema de ficheros donde toda la información este comprimida.

- *Mirror y RAID-Z:* Se pueden definir de forma muy sencilla mirroring entre discos y RAID-Z.

Actualmente cuando trabajamos con herramientas como Solaris Volume Manager hay que crear las particiones (slice), agrupar discos y crear la base de datos que alberga la configuración y estado de todos los volúmenes. Todo esto implica la ejecución de comandos como *prtvtoc*, *fmthard*, *metadb*, *metainit* y *metaroot* sin contar las modificaciones en el fcihero /etc/vfstab.

Con ZFS todo se resume a dos comandos *zfs* y *zpool*. Tal como podemos ver en la figura 4.1 ZFS trabaja con un pool que esta formado por todos los dispositivos físicos. Las características del pool son:

- El pool esta formado por dispositivos de almacenamiento de igual o diferentes capacidades.

- El pool puede crecer y encoger añadiendo y quitando discos.

- Los sistemas de ficheros ZFS comparten el pool y se puede definir cuotas y reservar de espacio para un solo sistema de ficheros.

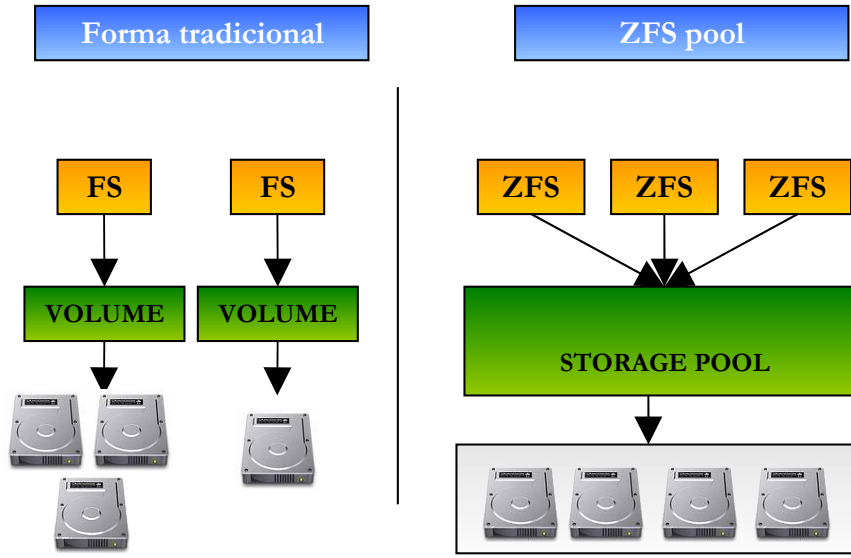


Figura 4.1

Crear un pool

Vamos a crear un pool de 4GB comprendido por lo discos c1d0 y c1d1 ambos con un tamaño de 2GB. Para crear un pool llamado babilonia utilizamos el comando *zpool* de la siguiente forma:

zpool create –f [nombre del pool] [lista de dispositivos]

```
#/usr/sbin/zpool create -f babilonia c1d0 c1d1
```

Para verificar que se ha creado correctamente ejecutamos el comando *zpool list*:

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

```
babilonia            3,88G   59,5K   3,87G     0%  ONLINE     -
#
```

Podemos ver que el tamaño del pool es de 3,88GB y es accesible desde */babilonia*.

Crear sistemas de ficheros ZFS

Ya tenemos creado el pool y ahora podemos comenzar a definir sistemas de ficheros. Vamos a crear dos sistemas de ficheros uno para instalar aplicaciones y otro para datos utilizando el comando *zfs* con la opción *create*:

zfs create [nombre del pool/ sistema de ficheros]

```
# /usr/sbin/zfs create babilonia/aplicaciones
# /usr/sbin/zfs create babilonia/datos
```

Para ver los nuevos sistema de ficheros ejecutamos el comando *zfs* con la opción *list*:

```
# zfs list
NAME                  USED  AVAIL  REFER  MOUNTPOINT
babilonia             138K  3,81G  27,5K  /babilonia
babilonia/aplicaciones  24,5K  3,81G  24,5K  /babilonia/aplicaciones
babilonia/datos       24,5K  3,81G  24,5K  /babilonia/datos
```

La salida muestra los dos nuevos sistemas de ficheros que también son visibles con el comando *df –k*. Se puede observar que todos los sistemas de ficheros comparten el mismo espacio de 3,81GB que es el tamaño del pool.

Los sistemas de ficheros creados con ZFS se montan automáticamente y de forma persistente. Con ZFS nos hemos ahorrado los pasos con los comandos *format*, *newfs*, *mount* y la posterior edición del fichero */etv/vfstab* para dejarlo de forma persistente.

Propiedades de un ZFS

Los sistemas de ficheros ZFS tienen propiedades que pueden ser activadas o desactivadas según nuestras necesidades. Para establecer el valor de una propiedad se ejecuta el comando *zfs* con las opciones *set* para establecer o *get* para leer. Veamos algunos de los mas interesantes:

Creación cuotas y reserva de espacio

ZFS nos da la posibilidad de controlar el espacio de los sistemas de ficheros estableciendo cuotas y reserva de espacio para los sistemas de ficheros. Continuando con el pool de los ejemplos anteriores vamos a establecer una cuota para el sistema de ficheros babilonia/aplicaciones y reservar espacio del pool para babilonia/datos.

Estableciendo una cuota a un sistema de ficheros limitamos el espacio máximo que puede tener. Para crear una cuota utilizamos el comando *zfs* y la opción *set quota=1GB*:

zfs set quota=[tamaño] [nombre del pool/ sistema de ficheros]

```
# # /usr/sbin/zfs set quota=1GB babilonia/aplicaciones
```

Si ejecutamos el comando *zfs list* para ver los sistemas de ficheros observaremos que se ha establecido la cuota correctamente:

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
babilonia              138K  3,81G    27,5K  /babilonia
babilonia/aplicaciones  24,5K  1024M    24,5K  /babilonia/aplicaciones
babilonia/datos        24,5K  3,81G    24,5K  /babilonia/datos
```

Una cuota limita el espacio de un sistema de ficheros pero no garantiza dicho espacio. Para reservar el espacio para un sistema de ficheros dentro del pool ejecutamos zfs con el parámetro *set reservation=1GB:*

zfs set reservation=[tamaño] [nombre del pool/ sistema de ficheros]

```
# /usr/sbin/zfs set reservation=1GB babilonia/datos
```

El sistema de ficheros *babilonia/datos* tiene reservados 1GB del pool. Si el resto de ficheros llena el pool no peligra nuestro espacio reservado. Ejecutando *zfls list* se puede ver que el pool *babilonia* tiene como usado 1GB y para el resto de sistemas de ficheros hay disponible 2,81GB. Salida de *zfs list*:

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
babilonia              1,00G  2,81G    27,5K  /babilonia
babilonia/aplicaciones  24,5K  1024M  24,5K  /babilonia/aplicaciones
babilonia/datos         24,5K  3,81G  24,5K  /babilonia/datos
```

Realizar una reserva de espacio no implica un limite de cuota para el sistema de ficheros este puede utilizar todo el tamaño del pool. Se pueden combinar las opciones de cuota y reserva para un sistema de ficheros.

Sistema de ficheros comprimido

ZFS ofrece nuevas posibilidades como tener un sistema de ficheros con información comprimida. El comando para habilitar la compresión del sistema de ficheros es *zfs* con el parámetro *compression=on.* El siguiente ejemplo activa la compresión para el sistema de ficheros babilonia/datos:

zfs set compression=[on/off] [nombre del pool/sistema de ficheros]

```
#/usr/sbin/zfs set compression=on babilonia/datos
```

Para obtener datos del ratio de compresión del sistema de ficheros:

zfs get [nombre de la propiedad] [nombre del pool/sistema de ficheros]

```
#zfs get compressratio babilonia/datos

NAME            PROPERTY       VALUE                  SOURCE
babilonia/datos  compressratio  1.00x                  -
```

Sistema de ficheros de solo lectura

Para establecer un sistema de ficheros como solo lectura utilizamos el colmando *zfs* con el parámetro *readonly=on*

zfs set readonly *=[on/off] [nombre del pool/sistema de ficheros]*

```
#/usr/sbin/zfs set readonly=on babilonia/datos
```

Cambiar el punto de montaje

Para cambiar el punto de montaje de un sistema de ficheros ejecutamos el comando *zfs* estableciendo el nuevo punto de montaje con el parámetro *mountpoint*. El siguiente ejemplo muestra como cambiar el punto de montaje de *babilonia/datos* a */aulaunix:*

zfs set mountpoint =[/punto de montaje] [nombre del pool/sistema de ficheros]

```
#cd /
# zfs set mountpoint=/aulaunix babilonia/datos
```

Verificamos el cambio ejecutando zfs list:

```
# zfs list
NAME                    USED  AVAIL  REFER  MOUNTPOINT
babilonia               1,00G  2,81G  25,5K  /babilonia
babilonia/aplicaciones  24,5K  1024M  24,5K  /babilonia/aplicaciones
babilonia/datos         24,5K  3,81G  24,5K  /aulaunix
```

Ver las propiedades de un ZFS

Hasta ahora hemos utilizado el comando *zfs set* para establecer propiedades del sistema de archivos. Para ver los valores de las propiedades de un ZFS ejecutamos *zfs* con la opción *get*:

zfs get [nombre de la propiedad] poll/sistemdeficheros

```
# zfs get quota  babilonia/aplicaciones
NAME                    PROPERTY  VALUE  SOURCE
babilonia/aplicaciones  quota     1G     local
```

Con la opción *all* podemos ver todos los valores de las propiedades de un ZFS y cuales son sus valores por defecto:

```
# zfs get all  babilonia/aplicaciones

NAME                    PROPERTY       VALUE                  SOURCE
babilonia/aplicaciones  type           filesystem             -
babilonia/aplicaciones  creation       Fri Jan 26 19:11 2007  -
babilonia/aplicaciones  used           24,5K                  -
babilonia/aplicaciones  available      1024M                  -
babilonia/aplicaciones  referenced     24,5K                  -
babilonia/aplicaciones  compressratio  1.00x                  -
babilonia/aplicaciones  mounted        yes                    -
babilonia/aplicaciones  quota          1G                     local
```

```
babilonia/aplicaciones  reservation   none                    default
babilonia/aplicaciones  recordsize    128K                    default
babilonia/aplicaciones  mountpoint    /babilonia/aplicaciones default
babilonia/aplicaciones  sharenfs      off                     default
babilonia/aplicaciones  checksum      on                      default
babilonia/aplicaciones  compression   off                     default
babilonia/aplicaciones  atime         on                      default
babilonia/aplicaciones  devices       on                      default
babilonia/aplicaciones  exec          on                      default
babilonia/aplicaciones  setuid        on                      default
babilonia/aplicaciones  readonly      off                     default
babilonia/aplicaciones  zoned         off                     default
babilonia/aplicaciones  snapdir       hidden                  default
babilonia/aplicaciones  aclmode       groupmask               default
babilonia/aplicaciones  aclinherit    secure                  default
```

Gestión del POOL

Añadir nuevos discos al pool

Para ampliar el espacio disponible en un pool tenemos que añadir nuevos discos a la máquina o utilizar particiones (slices) no usada en otro disco. Para ampliar el pool utilizamos el comando *zpool* con la opción *add*:

zpool add –f [nombre del pool] [dispositivo a añadir]

```
bash-3.00# zpool list -H
babilonia      1,98G   77,5K   1,98G   0%      EN LÍNEA      -

bash-3.00# /usr/sbin/zpool add -f babilonia c0d1

bash-3.00# zpool list -H
babilonia      3,97G   184K    3,97G   0%      EN LÍNEA      -
```

Tal como podemos ver en la salida de la ejecución de *zpool add* el espacio a aumentado de 1,89G a 3,97G. Se pueden añadir tantos discos como sean necesarios.

Quitar discos al pool

Se pude sacar un disco que forma parte del pool utilizamos la orden *zpool remove* :

zpool remove [nombre del pool] [dispositivo aquitar]

```
# zpool remove babilonia c0d1
```

Eliminar un pool

Si tenemos que eliminar un pool por completo y dejar los dispositivos libres para otro uso utilizamos el comando *zpool*:

zpool destroy –f [nombre del pool]

```
# /usr/sbin/zpool destroy -f babilonia
```

Esta opción es "destructiva" por lo que tenemos que estar muy seguros de que no tenemos información valiosa en el pool a borrar.

Reemplazar un disco del pool

En caso de que un de los discos falle y se tenga que reemplazar utilizamos el comando *zpool* con el parámetro *replace*:

zpool replace –f [pool] [dispositivo a reemplazar] [dispositivo nuevo]

Antes del cambio:

```
# zpool status
  conjunto: babilonia
 estado: ONLINE
 limpiar: resilver completed con 0 errores en Wed Jan 31 16:46:57 2007
```

```
config:

        NAME        STATE     READ WRITE CKSUM
        almoroz     ONLINE       0     0     0
          mirror    ONLINE       0     0     0
            c0d1    ONLINE       0     0     0
            c1d1    ONLINE       0     0     0

errores: ningún error de datosconocido
```

Crear un mirror (RAID 1)

Si tuviéramos que crear un espejo de dos discos utilizando herramientas como Solaris Volume Manager realizaríamos varias tareas para logar nuestro objetivo. Con ZFS es muy sencillo tan solo necesitamos con los comandos zpool y zfs.

zpool create –f [nombre del pool] mirror [dispositivos]

```
# zpool create -f  babilonia mirror c0d1 c1d1
```

Con *zpool status* verificamos si se ha creado correctamente:

```
bash-3.00# zpool status
  conjunto: babilonia
 estado: ONLINE
 limpiar: no se ha solicitado ninguna
config:

NAME            STATE     READ WRITE CKSUM
babilonia       EN LÍNEA       0     0     0
mirror          EN LÍNEA       0     0     0
c0d1            EN LÍNEA       0     0     0
c1d1            EN LÍNEA       0     0     0
```

Crear un mirror desde un disco existente.

Si tenemos un pool de un disco y queremos crear un mirror del disco acudimos al comando *zpool* con el parámetro *attach:*

zpool attach –f [pool] [dispositivo] [nuevo dispositivo]

```
zpool  attach -f  almoroz c0d1 c1d1
```

La ejecución del comando creara un mirror formado por los discos c0d1 ya existente y el nuevo disco c1d1.

Reemplazar un disco del mirror

En caso de que un de los discos falle y se tenga que reemplazar utilizamos el comando *zpool* con el parámetro *replace*:

zpool replace –f [pool] [dispositivo a reemplazar] [dispositivo nuevo]

Antes del cambio:

```
# zpool status
  conjunto: almoroz
 estado: ONLINE
 limpiar: resilver completed con 0 errores en Wed Jan 31 16:46:57 2007
config:

        NAME        STATE     READ WRITE CKSUM
        almoroz     ONLINE       0     0     0
          mirror    ONLINE       0     0     0
            c0d1    ONLINE       0     0     0
            c1d1    ONLINE       0     0     0

errores: ningún error de datosconocido
```

Realizamos el cambio del disco c1d1 por c0d1s0:

```
#/usr/sbin/zpool replace -f babilonia c0d1 c1d1
```

Y lo verificamos:

```
zpool status
 conjunto: almoroz
 estado: ONLINE
 limpiar: resilver completed con 0 errores en Wed Jan 31 16:46:57 2007
config:

        NAME          STATE     READ WRITE CKSUM
        almoroz       ONLINE       0     0     0
          mirror      ONLINE       0     0     0
            c0d1      ONLINE       0     0     0
            replacing ONLINE       0     0     0
              c0d1    ONLINE       0     0     0
              c1d1    ONLINE       0     0     0

errores: ningún error de datosconocido
# zpool status
  conjunto: almoroz
 estado: ONLINE
 limpiar: resilver completed con 0 errores en Wed Jan 31 16:46:57 2007
config:

        NAME        STATE     READ WRITE CKSUM
        almoroz     ONLINE       0     0     0
          mirror    ONLINE       0     0     0
            c0d1    ONLINE       0     0     0
            c0d1s0  ONLINE       0     0     0
```

Crear RAID-Z

ZFS permite la creación de RAID-Z muy similar a RAID 5. RAID 5 se compone como mínimo de tres discos y en cada uno de ellos se reserva un espacio con información de paridad.

RAID-Z cuenta con ventajas como la paridad distribuida simple y doble. La doble paridad permite asumir errores en uno dos discos que componen el RAIDZ.

Para crear un RAID-Z con paridad simple ejecutamos el comando zpool

zpool create –f [pool] raidz [dispositivos]

El ejemplo siguiente muestra la creación y su posterior verificación:

```
#/usr/sbin/zpool create -f babilonia raidz  c0d1 c1d1 c2t0d0
# zpool status
  conjunto: babilonia
 estado: ONLINE
 limpiar: no se ha solicitado ninguna
config:

        NAME        STATE     READ WRITE CKSUM
        babilonia   ONLINE       0     0     0
          raidz1    ONLINE       0     0     0
            c0d1    ONLINE       0     0     0
            c1d1    ONLINE       0     0     0
            c2t0d0  ONLINE       0     0     0

errores: ningún error de datos conocido
```

Para crear un RAID-Z con paridad doble ejecutamos el comando zpool

zpool create –f [pool] raidz2 [dispositivos]

```
#  zpool create -f babilonia raidz2  c0d1 c1d1 c2t0d0
# zpool status
  conjunto: babilonia
 estado: ONLINE
 limpiar: no se ha solicitado ninguna
config:

        NAME        STATE     READ WRITE CKSUM
        babilonia   ONLINE       0     0     0
          raidz2    ONLINE       0     0     0
            c0d1    ONLINE       0     0     0
            c1d1    ONLINE       0     0     0
            c2t0d0  ONLINE       0     0     0

errores: ningún error de datos conocido
```

Snapshots

Los snapshots son fotos de los datos de un sistema de ficheros, esto se hace de forma instantánea y comparte el espacio de los datos no modificados. Tiene gran utilidad para realizar modificaciones sobre

servicios y si no funcionan realizar un rollback de forma sencilla. Vamos un caso práctico:

Disponemos de los siguientes sistemas de ficheros en el pool llamado babilonia:

```
# zfs list
NAME                 USED  AVAIL REFER  MOUNTPOINT
babilonia            500M  3,42G  27,5K  /babilonia
babilonia/prueba1     500M  3,42G   500M  /babilonia/prueba1
babilonia/prueba2    24,5K  3,42G  24,5K  /babilonia/prueba2
```

El sistema de ficheros babilonia/prueba1 contiene los siguientes archivos:

```
# ls -lrt
total 1024223
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero1
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero2
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero3
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero4
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero5
```

Procedemos a crear el snapshot o foto del sistema de ficheros babilonia/prueba1 con el comando *zfs* y e parámetro *snapshots*:

zfs snapshot [pool/sistemadeficheros]@nombrefoto

Ejecutamos el comando:

```
#zfs snapshot babilonia/prueba1@nombredelafoto
```

Con *zpool list* observamos que se ha creado la foto:

```
#zfs list

 NAME                 USED  AVAIL  REFER  MOUNTPOINT
babilonia            300M  3,61G  27,5K  /babilonia
babilonia/prueba1     300M  3,61G   300M  /babilonia/prueba1
```

```
babilonia/prueba1@nombredelafoto      0      -   300M  -
babilonia/prueba2     24,5K  3,61G  24,5K  /babilonia/prueba2
```

Ahora vamos a simular un pequeño desastre borrando los archivos fichero4 y fichero5:

```
# rm fichero4 fichero5
# ls -lrt
total 614541
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero1
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero2
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero3
```

Recuperamos archivos los borrados recurriendo a snapshot creado. Recuperamos con el comando *zfs* y el parámetro *rollback:*

zfs rollback [pool/sistemadeficheros]@nombrefoto

Ejecutamos el comando:

```
#zfs rollback -r babilonia/prueba1@nombredelafoto
# ls -lrt
total 1024223
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero1
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero2
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero3
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero4
-rw------T   1 root     root     104857600 Feb  2 13:10 fichero5
```

Y el resultado es la recuperación de los archivos fichero4 y fichero5. Tal como se decía al principio la foto comparte los datos con los no modificados, en nuestro ejemplo se traduce con que la foto comparte el espacio de *fichero1, fichero2 y fichero3* que no se han modificado ocupando realmente solo el tamaño de *fichero4 y fichero5*.

Borrar snapshots o fotos

Cuando ya no sea necesario conservar un snapshot podemos borrarlo con el comando *zfs* y el parámetro *destroy*:

Ejemplo:

```
#zfs list

 NAME                  USED  AVAIL  REFER  MOUNTPOINT
babilonia             300M  3,61G  27,5K  /babilonia
babilonia/prueba1      300M  3,61G   300M  /babilonia/prueba1
babilonia/prueba1@nombredelafoto      0      -   300M  -
babilonia/prueba2     24,5K  3,61G  24,5K  /babilonia/prueba2

# zfs destroy -f babilonia/prueba1@nombredelafoto
```

Estados de ZFS

Con el comando *zpool status* obtenemos información sobre el estado de un pool:

zpool status [pool]

```
#zpool status babilonia
  conjunto: babilonia
 estado: ONLINE
 limpiar: no se ha solicitado ninguna
config:

        NAME        STATE     READ WRITE CKSUM
        babilonia   ONLINE       0     0     0
          mirror    ONLINE       0     0     0
            c0d1    ONLINE       0     0     0
            c1d1    ONLINE       0     0     0

errores: ningún error de datosconocido
```

Los estados posibles son:

- *ONLINE*: el dispositivo esta operando normalmente.
- *DEGRADED*: el dispositivo virtual tiene fallos pero continúa en funcionamiento. Este caso de puede dar cuando falla un dispositivo que forma parte de un RAIDZ. Hay que sustituir lo antes posible dispositivo dañado.
- *OFFLINE*: dispositivo parado manualmente por el administrador.
- *UNAVAILABLE* dispositivo no disponible. Este estado es el siguiente a DEGRADED cuando finalmente el dispositivo es inaccesible.

Con el comando *zpool status –x* obtenemos de forma rápida sin un pool esta teniendo problemas:

El siguiente ejemplo muestra un mirror compuesto por dos dispistivos fallando *c0d1* que esta totalmente inaccesible. Al ser un mirror el servicio continuo operativo pero se degrada el rendimiento y perdemos alta disponibilidad hasta que se reponga el disco dañado.

Ejemplo de un disco dañado:

```
# zpool status -x
  conjunto: babilonia
 estado: DEGRADED
estado: no se pudieron abrir uno o varios dispositivos.  Hay réplicas suficientes para
        que el conjunto pueda seguir funcionando con un menor rendimiento.
acción: adjunte el dispositivo que falta y establézcalo en línea mediante 'zpool online'.
   consulte: http://www.sun.com/msg/ZFS-8000-D3
 limpiar: no se ha solicitado ninguna
config:

        NAME        STATE     READ WRITE CKSUM
        babilonia   DEGRADED     0     0     0
          mirror    DEGRADED     0     0     0
            c0d1    UNAVAIL      0    62     0  no es posible abrir
            c1d1    ONLINE       0     0     0
```

```
errores: ningún error de datosconocido
```

La información que ofrece es detallada indicándonos el estado en el que se encuentra el mirror. En la salida del comando nos remite a la dirección web de Sun Microsystems *http://www.sun.com/msg/ZFS-8000-D3* donde podemos encontrar información y soluciones al problema.

Para solucionar este problema tenemos que añadir un nuevo disco al sistema o utilizar un slice de otro disco. Una vez localizado el disco que sustituye a c0d1 lo remplazamos con el comando *zpool replace*:

Pasos para reemplazar el disco dañado:

```
# zpool replace babilonia c0d1 c2t0d0

#zpool status babilonia
  conjunto: babilonia
 estado: ONLINE
 limpiar: resilver completed con 0 errores en Fri Feb  2 15:04:57 2007
config:

        NAME        STATE     READ WRITE CKSUM
        babilonia   ONLINE       0     0     0
          mirror    ONLINE       0     0     0
            c2t0d0  ONLINE       0     0     0
            c1d1    ONLINE       0     0     0

errores: ningún error de datosconocido
```

En el ejemplo hemos reemplazado el disco *c0d1* por *c2t0d0* y automáticamente el mirror para a estado ONLINE y replica los datos de c0d1 a c2t0d0.

También podemos optar por sacar el disco del mirror para repararlo. Tenemos que desasociar el disco *c0d1* del mirror con el comando.

zpool detach:

zpool detach [pool] [dispositivo]

```
#zpool detach babilonia   c0d1
```

El disco c0d1 ya no forma parte del mirror babilonia. Podemos someterlo a pruebas y volver a añadirlo al mirror o añadir otro disco distinto se la siguiente forma:

zpool attach –f [pool] [chisposito] [nuevo dispositivo]

```
# zpool attach -f  babilonia  c1d1 c0d1
# zpool status
  conjunto: babilonia
 estado: ONLINE
 limpiar: resilver completed con 0 errores en Fri Feb  2 15:18:50 2007
config:

        NAME        STATE     READ WRITE CKSUM
        babilonia   ONLINE       0     0     0
          mirror    ONLINE       0     0     0
            c1d1    ONLINE       0     0     0
            c0d1    ONLINE       0     0     0

errores: ningún error de datosconocido
```

Operaciones entrada y salida

ZFS proporciona el comando *zpool iostat* que reporta e sobre operaciones de lectura, escritura y ancho de banda. El siguiente ejemplo muestra la salida del comando *zpool iostat*:

zpool iostat [pool]

```
# zpool iostat babilonia
               capacity     operations    bandwidth
pool         used  avail   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
babilonia   10,1M  1,97G      0      0     41      2
```

Para obtener información de todos los dispositivos virtuales:

zpool iostat –v [pool]

```
# zpool iostat –v babilonia
              capacity     operations    bandwidth
pool        used  avail   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
babilonia   10,1M  1,97G      0      0     41      2
  mirror    10,1M  1,97G      0      0     42      2
    c1d1        -      -      0      0     44    111
    c0d1        -      -      0      0      0     52
----------  -----  -----  -----  -----  -----  -----
```

Si omitimos el nombre del pool en el comando lista la información de todos los pool disponibles.