ZFS — O novo sistema de arquivos da Sun (parte 2)

Ainda sem limites

Se você se interessou pela primeira parte da apresentação do ZFS, vai gostar de saber que há outros recursos ainda mais impressionantes a conhecer.

Continuando a apresentação do inovador sistema de arquivos ZFS, iniciada na última edição da **Linux Magazine**[1], vamos nos aprofundar em seus recursos mais avançados, mostrando mais alguns benefícios que usuários de Linux e Solaris podem obter no uso desse sistema de arquivos.

Em ambientes onde é necessário coletar estatísticas dos sistemas de arquivos, o ZFS traz um simples, porém suficiente, suporte para a coleta das mesmas de forma simples e rápida para discos e *pools* (volumes).

No **exemplo 1**, o sistema tem um pool (*pool-one*) com um *mirror* — isso é, volumes com conteúdo espelhado, para melhorar a característica de segurança dos dados — onde cada componente do mirror tem dois discos.

Substituições

É claro que nenhum sistema é à prova de transtornos; um dos principais obstáculos encontrados quando se manipula dados é a falha física de discos. Supondo que o Solaris ou Linux tenham sistemas de arquivos ZFS construídos sobre *RAID* 1 ou *RAID* Z (recordando: RAID Z, em ZFS, é uma versão melhorada do RAID 5), é possível, em caso de problemas, substituir um disco ruim por outro, sem perder os dados.

Exemplo 1: Volumes espelhados

```
# zpool iostat -v
                              capacity     operations    bandwidth
pool                        used  avail   read  write   read  write
---------                   ---   ---     ---    ---     ---    ---
pool-one                   56.2M   231M     0      0      49      0
    mirror                 28.0M   164M     0      0      34      0
      c0d1                    -      -     0      0     568      0
      c1d1                    -      -     0      0     534      0
    mirror                 28.2M  67.3M     0      0      15      0
      /zfs-teste/zfsfile1     -      -     0      0     331      0
      /zfs-teste/zfsfile2     -      -     0      0     345      0
---------                   ---    ---    ---    ---     ---    ---
```

Exemplo 2: Pool criado

```
# zpool list
NAME        SIZE  USED  AVAIL  CAP  HEALTH  ALTROOT
linuxmag   95.5M  138K  95.4M   0%  ONLINE  -
```

Exemplo 3: Criação de um sistema de arquivos

```
# zfs create linuxmag/zfs1
# zfs list
NAME            USED  AVAIL  REFER  MOUNTPOINT
linuxmag       1.52M  62.0M    19K  /linuxmag
linuxmag/zfs1  1.40M  62.0M  1.40M  /linuxmag/zfs1
```

Exemplo 4: Status de um espelhamento

```
# zpool status
    pool: linuxmag
   state: ONLINE
   scrub: none requested
config:

NAME                    STATE     READ WRITE CKSUM
linuxmag                ONLINE       0     0     0
  mirror                ONLINE       0     0     0
    /zfs-teste/zfsfile1 ONLINE       0     0     0
    /zfs-teste/zfsfile2 ONLINE       0     0     0

errors: No known data errors
```

É lógico que isso é o mínimo que se espera em qualquer sistema crítico, entretanto o ZFS facilita muito essa funcionalidade.

O procedimento abaixo evidencia o conceito, e será feito com arquivos, já que o leitor pode não ter discos suficientes para o laboratório, ou não possuir ambientes virtualizados adequados para tais comandos.

Nos passos abaixo, seguem os comandos para a criacão de arquivos em disco com 100 MB de tamanho que serão usados como discos:

```
# mkdir /zfs-teste
# for x in seq 1 5; do
mkfile 100m \
/zfs-teste/zfsfile${x};
done
```

Agora que os arquivos que simulam os discos usados para testes já estão disponíveis, agrupa-se os mesmos com o comando `zpool`, o que possibilitará posteriormente a criacão dos sistemas de arquivos utilizando ZFS:

```
# zpool create linuxmag mirror
➥zfs-teste/zfsfile1 /zfs-teste/
➥zfsfile2
```

Visualizando os pools existentes no sistema, o **exemplo 2** mostra que já podemos encontrar aquele criado no passo anterior.

Com o pool *linuxmag* já construído, é possível agora criar tantos sistemas de arquivos ZFS quantos forem necessários (**exemplo 3**). Deve-se ressaltar que, como o pool é baseado em um sistema de espelhamento, o procedimento será efetuado em quaisquer sistemas de arquivos pertencentes ao mesmo pool.

Através do comando `zpool status` (**exemplo 4**) é fácil verificar quais sistemas de arquivos existem dentro do pool, e ainda qual é o arranjo em questão (nesse caso, espelho).

A partir desse ponto, a existência de um sistema de arquivos ZFS permite que o mesmo seja populado com dados de forma tradicional:

```
# cd /boot/grub
# cp -r * /linuxmag/zfs1
```

Dentro desse sistema de testes, pode-se simular uma falha desligando-se um dos "discos" e verificando-se o que ocorre, como mostra o **exemplo 5**. Note que a própria saída de `zpool status` sugere que o disco seja reativado com a opção `online` ou trocado por outro que funcione corretamente com a opção `replace`. No caso de discos rígidos reais, após a substituição do

Exemplo 5: Simulação de uma falha

```
# zpool offline linuxmag /zfs-teste/zfsfile1
# zpool status
   pool: linuxmag
   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
linuxmag                DEGRADED     0     0     0
  mirror                DEGRADED     0     0     0
    /zfs-teste/zfsfile1 OFFLINE      0     0     0
    /zfs-teste/zfsfile2 ONLINE       0     0     0

errors: No known data errors
```

Exemplo 6: Comando zpool replace

```
# zpool replace linuxmag /zfs-teste/zfsfile1 /zfs-teste/zfsfile3
# zpool status
    pool: linuxmag
   state: ONLINE
   scrub: resilver completed with 0 errors on Tue Aug  7 03:09:16 2007
config:

NAME                      STATE     READ WRITE CKSUM
linuxmag                  ONLINE       0     0     0
  mirror                  ONLINE       0     0     0
    /zfs-teste/zfsfile3   ONLINE       0     0     0
    /zfs-teste/zfsfile2   ONLINE       0     0     0

errors: No known data errors
```

Exemplo 7: Exportação de um pool

```
01 # zpool export linuxmag
02 # zpool list
03 No pools available to import
04 # zfs list
05 no datasets available
```

Exemplo 8: Informando um diretório na importação

```
# zpool import -d /zfs-teste
   pool: linuxmag
     id: 18420575686731237941
   state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

linuxmag                  ONLINE
  mirror                  ONLINE
    /zfs-teste/zfsfile3   ONLINE
    /zfs-teste/zfsfile2   ONLINE
```

componente defeituoso (o "disco" `zfsfile2`, em nosso exemplo), usaríamos o comando `zpool replace` de acordo com o **exemplo 6**.

Observe que, logo após a troca de disco, o pool — e todos os seus sistemas de arquivos, por consequência — passa por um processo de re-sincronização dos dados (chamado de *resilvering*), até o disco novo conter exatamente os mesmos dados do disco remanescente no espelho RAID 1 durante a falha.

Importar e Exportar

Certamente uma excelente característica do ZFS é sua capacidade de exportar e importar pools de uma máquina para outra, como todos os seus volumes associados. Essa técnica sempre foi muito usada em ambientes de cluster, como o próprio *Sun Cluster*, onde se tornou habitual o uso de ferramentas como o *Solaris Volume Manager* (antigo *Disk Suite*) ou o *Veritas Volume Manager* para a realização de tal tarefa.

Com o uso do ZFS no Solaris 10 ou Linux, exportar e importar tais pools ficou muito simples. Os passos mostrados no **exemplo 7** evidenciam essa facilidade. Após exportar o pool (**linha 1**), pode-se verificar que, uma vez exportado, o mesmo não mais fica disponível para nosso host (**linha 3**), assim como quaisquer sistemas de arquivos contidos dentro do mesmo (**linha 5**).

O **exemplo 7** mostra que embora exista um pool (*linuxmag*) e dentro dele haja um sistema de arquivos espelhado (*zfs1*), ambos não aparecem em quaisquer comandos pelo fato de estarem exportados, ou seja, estarem disponíveis para a aquisição por outra máquina que seja capaz de visualizar o mesmo pool. A ocorrência de acesso a um único pool por múltiplas máquinas é mais comum em ambientes onde os discos estão em *storages* compartilhados.

Caso o comando `export` não funcione, relatando *device busy* (dispositivo ocupado), deve-se tentar o mesmo comando com a opção `-f` (*force*). Note que o comando `zpool export -f` não acarreta perda de dados, já que o ZFS é totalmente transacional, ou seja, para alterações incompletas é realizado um *roolback* das operacões (no mesmo sentido dos bancos de dados) e, para alterações finalizadas, é realizado um *commit*.

Surpresa

É claro que o uso da opcão `force` é desnecessário quando for possível desmontar os sistemas de arquivos antes; todavia, quando o acesso aos discos estiver comprometido (por uma falha física do mesmo, por exemplo), não há outra alternativa.

Mesmo assim, esse comando traz uma surpresa: segundo sua saída, não há pools para importar:

```
# zpool export -f linuxmag
# zpool import
No pools available to import
```

Isso ocorre porque a opção `import` procura discos exportados apenas

Exemplo 9: Criação de um pool em RAID Z

```
# zpool create otherpool raidz /zfs-teste/zfsfile1 /zfs-teste/zfsfile2 /zfs-teste/zfsfile3
# zpool list
NAME         SIZE  USED  AVAIL  CAP  HEALTH     ALTROOT
otherpool    286M  189K   286M   0%  ONLINE      -
# zfs create otherpool/zfs-raidz-1
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
otherpool              153K   158M  25.3K  /otherpool
otherpool/zfs-raidz-1  24.0K   158M  24.0K  /otherpool/zfs-raidz-1
```

Exemplo 10: Criação de um snapshot

```
01 # cp /etc/nsswitch.conf /otherpool/zfs-raidz-1
02 # cp /etc/hosts /otherpool/zfs-raidz-1
03 # cp /etc/services /otherpool/zfs-raidz-1
04 # ls -lah /otherpool/zfs-raidz-1
05 -r--r--r-- 1 root root 1.0K Jul 9 23:11 hosts
06 -rw-r--r-- 1 root root 1.8K Jul 9 23:11 nsswitch.conf
07 -r--r--r-- 1 root root 4.0K Jul 9 23:11 services
08 # zfs snapshot otherpool/zfs-raidz-1@snap1
09 # zfs list
10 NAME                          USED  AVAIL  REFER  MOUNTPOINT
11 otherpool                     163K   158M  25.3K  /otherpool
12 otherpool/zfs-raidz-1        34.0K   158M  34.0K  /otherpool/zfs-
   ➥raidz-1
13 otherpool/zfs-raidz-1@snap1     0    -     34.0K   -
```

no diretório `/dev/dsk`, e em nenhum outro lugar. Nesse exemplo, como foram usados arquivos no lugar de discos reais, o correto seria especificar o diretório onde procurar os pools exportados, através do parâmetro `-d /zfs-teste`, como indica o **exemplo 8**.

Enfim, para importar o pool anteriormente exportado, basta especificá-lo:

```
# zpool import -d /zfs-teste
➥linuxmag
# zpool list
NAME        SIZE  USED  AVAIL  CAP
➥HEALTH  ALTROOT
linuxmag   95.5M  138K  95.4M   0%
➥ONLINE   -
```

Snapshots

Solaris e Linux têm um comportamento um tanto árido quanto ao apagamento de arquivos: uma vez apagados, não há possibilidade de recuperação. No ZFS, não é necessário lançar mão de ferramentas externas para implementar esse recurso; pode-se usar o interessante recurso de *Snapshots*.

Um snapshot é como uma fotografia (estática, naturalmente, como toda fotografia) dos dados, somente para leitura. Com isso, em caso de desastres, é possível voltar o sistema de arquivos a um momento anterior ao acidente, de forma que seja possível recuperar a informação perdida.

O **exemplo 9** demonstra a aplicação de snapshots sobre um novo pool, chamado *otherpool*, composto por três discos (no caso, novamente usamos arquivos) em RAID Z, sobre os quais criamos um sistema de arquivos ZFS. Após copiarmos alguns arquivos do diretório `/etc/` para o novo sistema de arquivos (**exemplo 10**, **linhas 1** a **7**), criamos o snapshot (comando `zfs snapshot`) do mesmo, com o nome de `zfs-raidz-1@snap1` (**linha 8**).

Note os tamanhos usado (*USED*) e referenciado (*REFER*) pelo snapshot. O tamanho referenciado é de 34 KB, pois o sistema *zfs-raidz-1* ocupa esse mesmo tamanho. Entretanto, o snapshot ocupa 0 KB. Isso ocorre porque o snapshot emprega o conceito de *copy-on-write*: um arquivo somente é copiado para o snapshot quando seu conteúdo no sistema de arquivos original for alterado.

É fundamental ressaltar também que o snapshot é visível, nesse caso, sob a estrutura `/otherpool/zfs-raidz-1/.zfs/snapshot`. Sob esse diretório poderia haver diversos snapshots, cada um refletindo um momento específico do sistema de arquivos *zfs-raidz-1*.

Se apagarmos do sistema de arquivos *zfs-raidz-1* do arquivo `nsswitch.conf`, que ocupa 1,8 KB, teremos uma saída diferente para o comando `zfs list` (**exemplo 11**). Note que o snapshot *snap1* passou a ocupar espaço, justamente

Exemplo 11: Snapshot após alteração

```
# rm /otherpool/zfs-raidz-1/nssswitch.conf
# zfs list
NAME                          USED  AVAIL  REFER  MOUNTPOINT
otherpool                     200K   158M  25.3K  /otherpool
otherpool/zfs-raidz-1        56.6K   158M  32.0K  /otherpool/zfs-
raidz-1
otherpool/zfs-raidz-1@snap1  24.6K   -     34.0K  -
```

Exemplo 12: Recuperação de um snapshot

```
# zfs rollback otherpool/zfs-raidz-1@snap1
# zfs list
NAME                          USED  AVAIL  REFER  MOUNTPOINT
otherpool                     165K   158M  26.6K  /otherpool
otherpool/zfs-raidz-1        34.0K   158M  34.0K  /otherpool/zfs-
raidz-1
otherpool/zfs-raidz-1@snap1      0      -  34.0K  -
# ls -lh otherpool/zfs-raidz-1
total 20
-r--r--r-- 1 root root  1.0K Jul  9 23:11 hosts
-rw-r--r-- 1 root root  1.8K Jul  9 23:11 nsswitch.conf
-r--r--r-- 1 root root  4.0K Jul  9 23:11 services
```

Exemplo 13: Teste de corrupção de arquivos

```
01 # zpool create sunpool mirror c0d1 c1d1
02 # zfs create sunpool/zfs9
03 # cp /etc/s* /sunpool/zfs9
04 # ls -lah /sunpool/zfs9
05 total 130
06 -rwxr-xr-x 1 root root   23 Jul 10 06:57 scrollkeeper.conf
07 -rw-r--r-- 1 root root  965 Jul 10 06:57 sdp.conf
08 -r--r--r-- 1 root root 4.0K Jul 10 06:57 services
09 -r-xr-xr-x 1 root root  973 Jul 10 06:57 setmnt
10 -r---- 1 root root  346 Jul 10 06:57 shadow
11 ...
12
13 # dd if=/dev/zero of=/dev/rdsk/c0d1 count=170 bs=512k
14 # zpool status -x
15 all pools are healthy
16 # ls -lh /sunpool/zfs9
17  total 130
18 -rwxr-xr-x 1 root root   23 Jul 10 06:57 scrollkeeper.conf
19 -rw-r--r-- 1 root root  965 Jul 10 06:57 sdp.conf
20 -r--r--r-- 1 root root 4.0K Jul 10 06:57 services
21 -r-xr-xr-x 1 root root  973 Jul 10 06:57 setmnt
22 -r---- 1 root root  346 Jul 10 06:57 shadow
23 ...
```

Exemplo 14: Comando zpool scrub

```
01 # zpool scrub sunpool
02 # zpool status sunpool
03
04 pool: sunpool
05 state: ONLINE
06 status: One or more devices has experienced an unrecoverable
   ➥error.  An
07 attempt was made to correct the error. Applications are
   ➥unaffected.
08 action: Determine if the device needs to be replaced, and clear
   ➥the errors
09 using 'zpool clear' or replace the device with 'zpool replace'.
10    see: http://www.sun.com/msg/ZFS-8000-9P
11  scrub: scrub completed with 0 errors on Tue Jul 10 07:08:20 2007
12 config:
13
14 NAME      STATE   READ WRITE CKSUM
15 sunpool   ONLINE     0     0     0
16   mirror  ONLINE     0     0     0
17     c0d1  ONLINE     0     0    96
18     c1d1  ONLINE     0     0     0
19
20 errors: No known data errors
```

no tamanho do arquivo apagado (1,8 KB) somado ao overhead associado à criação de um arquivo no sistema de arquivos (aproximadamente 22,8 KB). A razão disso é que o arquivo em questão foi alterado (no caso, apagado) no sistema de arquivos original, e, portanto, foi imediatamente copiado para o snapshot.

Para recuperar o arquivo do snapshot, usa-se a opção `roolback` conforme o **exemplo 12**. Após o rollback, o arquivo `otherpool/zfs-raidz-1/nsswitch.conf` é recuperado. Caso o snapshot do sistema de arquivos não seja mais necessário, pode-se apagá-lo com o comando `zfs destroy`:

```
# zfs destroy otherpool/zfs-raidz-
➥1@snap1
```

ZFS – Scrub e Checksum

O ZFS tem um mecanismo de verificação por *checksum* que garante a integridade dos dados localizados no sistema de arquivos. Quando esse mecanismo é aplicado em um pool baseado em RAID 1 ou RAID Z — somente esses níveis de RAID são suportados — a eventual corrupção de dados no sistema de arquivos não exigirá o uso do comando `fsck`. Os dados permanecerão íntegros, já que o ZFS corrige a corrupção automaticamente.

O **exemplo 13** mostra como experimentar esse recurso, criando um novo pool (**linha 1**) com um novo sistema de arquivos (**linha 2**), copiando arquivos para dentro do mesmo (**linha 3**) e em seguida corrompendo propositalmente os arquivos com o comando `dd` (**linha 13**). É possível notar que após a tentativa de corrupção os dados permanecem totalmente intactos (**linhas 18 a 22**).

Exemplo 15: Comando zpool clear

```
01 # zpool clear sunpool
02 # zpool status
03 pool: sunpool
04 state: ONLINE
05 scrub: scrub completed with 0 errors on Tue Jul 10 07:08:20 2007
06 config:
07 NAME        STATE     READ WRITE CKSUM
08 sunpool     ONLINE       0     0     0
09   mirror    ONLINE       0     0     0
10     c0d1    ONLINE       0     0     0
11     c1d1    ONLINE       0     0     0
12
13 errors: No known data errors
```

Como se não fosse suficiente, é possível ainda usar o comando `zpool scrub` (**exemplo 14**) para forçar uma verificação do sistema de arquivos (**linha 1**), e com isso se tem mais uma surpresa. Além de o ZFS não permitir que os dados sejam corrompidos, ele ainda traz informações (**linhas 6 e 7**) sobre o problema enfrentado (mesmo que tenha se recuperado), fornecendo inclusive sugestões de reparo e uma URL onde o administrador pode obter mais informações sobre o problema.

O **exemplo 15** demonstra a limpeza das informações sobre o evento com o comando `zpool clear`.

Conclusão

O ZFS tem merecido constante atenção por parte da Sun e isto fez com que desenvolvedores de outros sistemas operacionais, como o Linux[2] e o FreeBSD[3], também devotassem atencão para o mesmo, comprovando assim a qualidade indiscutível do ZFS e o comprometimento da comunidade com seu aperfeiçoamento.

O ZFS ainda ganhará mais espaço entre seus concorrentes de mercado, como *UFS*, *Ext2*, *Ext3*, *ReiserFS*, *XFS* e *VxFS*.

Não foi o intuito desta dupla de artigos (veja a primeira parte na edição 33 da **Linux Magazine**) explicar todas as vantagens e possibilidades do ZFS, mas sim ressaltar tópicos relevantes que ajudassem a escolhê-lo como base para os dados de máquinas de missão crítica. Para todos os outros recursos — que são muitos — é recomendado que o leitor procure documentos no site da Sun ou ainda busque treinamento específico na Sun Education[4]. ■

Mais informações

[1] Linux Magazine 33: http://www.linuxnewmedia.com.br/lm/issue/lm_33_dual_core/

[2] ZFS no FUSE/Linux: http://zfs-on-fuse.blogspot.com/

[3] Anúncio da inclusão do ZFS no FreeBSD: http://lists.freebsd.org/pipermail/freebsd-current/2007-April/070544.html

[4] Whitepaper Sun, ZFS vs. Ext3: http://www.sun.com/software/whitepapers/solaris10/zfs_linux.pdf

[5] Whitepaper Sun, ZFS vs. VxFS: http://www.sun.com/software/whitepapers/solaris10/zfs_veritas.pdf

[6] Curso de ZFS na Sun: http://www.sun.com/training/catalog/courses/SA-229-S10.xml

[7] Comunidade ZFS: http://www.opensolaris.org/os/community/zfs/

[8] Manual de ZFS: http://docs.sun.com

O autor

Alexandre Borges é Instrutor de Tecnologia da Informação (Senior Specialist IT - Sun Microsystems Instructor).