# Sem limites

*Um sistema de arquivos virtualmente impossível de ser esgotado, com múltiplas ferramentas que permitem, inclusive, a montagem automática de RAID níveis 0, 1 e Z. Não é uma promessa de ficção científica: esse sistema existe e foi criado pela Sun.*

**por Alexandre Borges**

Em 2004, a Sun abriu o código do Solaris para que a comunidade de desenvolvedores pudesse contribuir com o seu desenvolvimento e também pudesse trocar experiências de maneira mais ampla. Essa abertura tentava repetir o movimento extremamente bem sucedido que o Linux iniciou no mundo de TI e que mudou, de maneira profunda, o modo como se encara – sobretudo no mundo corporativo – o progresso e uso de softwares nos dias de hoje.

Esta abertura de código e mudança de pensamento dentro da Sun resultou no projeto *OpenSolaris*. A primeira parte do código do Solaris que a empresa disponibilizou foi a revolucionária ferramenta de *troubleshooting* chamada *Dtrace*, que vem ajudando decisivamente os administradores a solucionarem os problemas de performance causados por aplicações rodando em ambiente Solaris. A seguir o próprio código do sistema operacional foi aberto, e a empresa começou a se alinhar com a inevitável tendência iniciada por seguidores do Linux e do código aberto.

## Variedades

Em todos os sistemas Unix e suas variantes, a preocupação com performance e tolerância a falhas sempre está presente. Sem dúvidas, o ponto crítico para o sucesso de um projeto é a escolha de qual tipo de sistema de arquivos será utilizado para a instalação e execução de suas aplicações.

Dependendo do sistema operacional utilizado, não se tem muita opção em relação ao sistema de arquivos a escolher: é isso o que vemos no mundo Windows, com seu FAT32 e as versões v.4 e v.5 do NTFS. Isso ocorre de maneira nitidamente diferente (e melhor, por quê não?) nos sistemas Linux, com seu amplo suporte para diversos tipos e conceitos de filesystems, como o *ext2*, *ext3*, *ReiserFS v.3* e *v.4*, *vfat*, entre muitos outros.

O sistema de arquivo em disco comumente usado pelo Solaris é o *UFS* (*Unix File System*), que une performance com segurança na integridade dos dados. A Sun sempre ofereceu o UFS como seu sistema de arquivos para disco de boot, assim como também para discos e *storages* que abrigam aplicações.

Existe um documento muito interessante que o leitor pode consultar, e que compara a performance do UFS, ext3 e ReiserFS v.3 [1].

Seguindo a avalanche de tantas boas novidades do Solaris 10, a Sun decidiu também investir na concepção de um novo file system que pudesse melhorar a performance e

usabilidade para trabalhar com arquivos no Solaris, elevando o mesmo a um novo patamar de performance de I/O para as aplicações críticas e competindo com os grandes filesystems de mercado: o ZFS (Zetabyte File System).

## Surge o ZFS

O ZFS surgiu por volta de 2004, ainda quando o Solaris 10 estava no estágio de versão beta. Nessas versões iniciais, o ZFS era apenas distribuído internamente dentro da Sun, no formato de dois pacotes para instalação à parte (*SUNWzfsr* e *SUNWzfsg*), pois havia alguns problemas mais sérios (que fazem parte de qualquer processo de criação de software). Oficialmente, o ZFS apenas veio a aparecer "*bundled*" no Solaris 10, release 06/06.

O ZFS tem muitas *features* interessantes, que simplificam muito o dia-a-dia de um administrador:

- ZFS é um filesystem de 128 bits, podendo assim endereçar e suportar bilhões de terabytes em dados;
- O gerenciamento de volume, antes feito com ferramentas como o *Solaris Volume Manager* (antigo *Solstice Disk Suite*) e *Veritas Volume Manager* (produto da Veritas – atual Symantec), já está integrado com o ZFS, e pode-se criar sistemas de arquivos do tipo ZFS sobre volumes (chamados de "*pools*") em RAID 0, RAID 1 e RAID Z (este último é um tipo de RAID 5 melhorado na operação de escrita) de forma trivial;
- Suporte para *Sun Cluster* 3.2 e *Zones* em Solaris 10;
- Todo filesystem ZFS tem propriedades associadas e em sua maioria elas podem ser alteradas on-line de modo que não é necessário desmontá-lo para que as mudanças tornem-se ativas;
- Não existe mais a necessidade de alterar-se o arquivo `/etc/vfstab` (o equivalente do `/etc/fstab` no Linux) para que sejamos capazes de montar sistemas de arquivos ZFS no momento da inicialização da máquina até porque, ao criar um sistema ZFS, sua montagem é automática;
- Não há mais a necessidade de trabalhar com *slices* (partições) nos discos usados para ZFS. É possível, se desejado, utilizar um ou diversos discos inteiros para volumes (*pools*) em ZFS. O Solaris usa, para isto, um layout de partições *EFI*. Deve-se ressaltar que ainda é factível usar partições para construção de pools que receberão sistemas de arquivos ZFS, entretanto esta não é a opção mais utilizada em sistemas críticos;
- Todas as operações de escrita são feitas usando a técnica *copy-on-write*, tornando os dados sempre consistentes e dispensando o uso do comando fsck no ZFS;
- Suporte completo para ACLss (que podem ser visualizadas e modificadas facilmente com os comandos *ls* e *chmod* ao invés de utilizar os tradicionais *getfacl* e *setfacl*);
- Facilidade para importar e exportar pools entre sistemas Self-healing (autocorreção) para dados em pools no arranjo RAID 1 e RAID Z através do uso de algoritmos de checksum.

## Usando o ZFS

Vamos explanar algo sobre os comandos e conceitos utilizados pelo ZFS, para que o leitor possa fazer sua própria avaliação em relação as qualidades desse sistema.

Para fazer testes com ZFS, o leitor poderá utilizar qualquer um dos seguintes cenários:

- Instalar o Solaris 10 (no mínimo release 11/06 – questão de estabilidade) em uma máquina com processadores SPARC, utilizando discos internos ou externos para testes com ZFS;
- Instalar o Solaris 10 (release 11/06) em máquinas com arquitetura x86, também com discos internos ou externos para testes.;
- Instalar o Solaris 10 (release 11/06) no VMware (que agora tem sua versão gratuita para Linux e Windows) ou no VirtualBox. A vantagem de sistemas virtualizados é que o leitor pode criar muitos discos virtuais e fazer quaisquer testes necessários com o ZFS;
- Usar o Solaris 10 em qualquer das situações acima, contudo ao invés de usar discos virtuais, usar arquivos em disco. (usando o comando `mkfile`).

Utilizaremos em nossos exemplos o Solaris 11, com a opção de arquivos em disco via *mkfile* em alguns exemplos e discos virtuais (via Vmware) em outros exemplos. Todos os exemplos enunciados aqui podem ser reproduzidos, facilmente, em qualquer ambiente equipado com alguns requisitos mínimos.

Em geral, quando pensamos na criação e uso de filesystems, sempre vêm à mente os seguintes passos:

- particionamento de discos (usando os comandos `format` em Solaris ou `fdisk` em Linux) para dedicar uma partição para cada filesystem;
- criação do sistema de arquivos em si (comandos `newfs` no Solaris ou `mkfs` no Linux) de modo que seja possível construir uma estrutura de arquivos e diretórioS;
- Tornar disponível o filesystem para uso geral (comando `mount` em ambos os ambientes).

A utilização do ZFS é muitíssimo mais simples: ele não trabalha com o conceito de particionamento usando essas etapas, ou seja, não é necessário criar uma partição à parte para um sistema criado a partir de ZFS e nem, muito menos, é preciso montar (comando *mount*)

explicitamente o filesystem, já que ele é montado automaticamente na criação e no boot da máquina.

## Criando pools e filesystems

Antes de tudo, vamos criar alguns arquivos em disco:

```
bash-3.00# mkdir /zfs-teste
bash-3.00# mkfile 100m /zfs-teste/
➥zfsfile1
bash-3.00# mkfile 100m /zfs-teste/
➥zfsfile2
bash-3.00# mkfile 100m /zfs-teste/
➥zfsfile3
bash-3.00# mkfile 100m /zfs-teste/
➥zfsfile4
bash-3.00# mkfile 100m /zfs-teste/
➥zfsfile5
```

Agora, também para referência, podemos listar quais discos temos em nossa máquina de teste:

```
bash-3.00# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
     0. c0d0 <DEFAULT cyl 1563 alt
➥2 hd 255 sec 63>
     /pci@0,0/pci-ide@7,1/ide@0/
➥cmdk@0,0
     1. c0d1 <DEFAULT cyl 202 alt
➥2 hd 64 sec 32>
     /pci@0,0/pci-ide@7,1/ide@0/
➥cmdk@1,0
     2. c1d1 <DEFAULT cyl 202 alt
➥2 hd 64 sec 32>
     /pci@0,0/pci-ide@7,1/ide@1/
➥cmdk@1,0
Specify disk (enter its number): ^D
```

Agora podemos criar um pool. Um pool pode ser interpretado como um volume, usando quaisquer tipos de RAID (0, 1 e 5), onde é possível criar quantos filesystems ZFS forem requeridos, sem qualquer tipo de restrição ou particionamento. É como se fosse um espaço livre para uso. Assim:

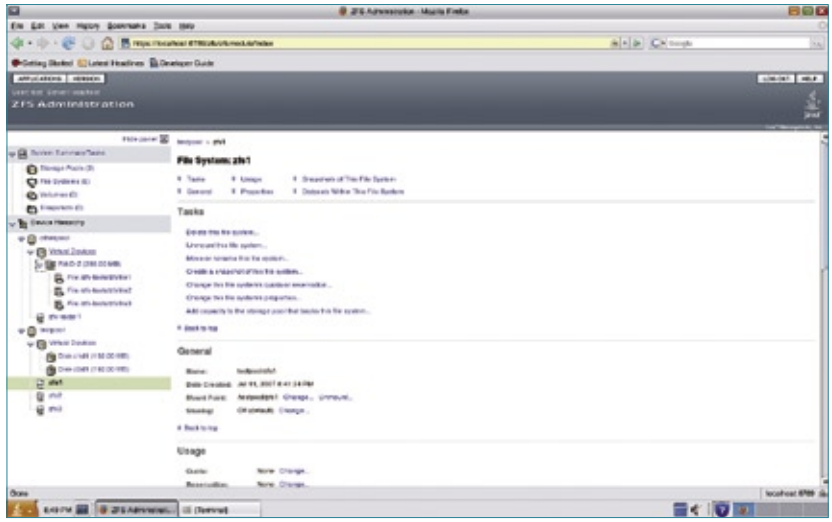

**Figura 1** Os principais comandos de administração de infraestrutura *ZFS* estão reunidos em uma única interface web.

```
bash-3.00# zpool create testpool
➥mirror c0d1 c1d1
bash-3.00# zpool list
NAME                    SIZE    USED
➥AVAIL    CAP  HEALTH     ALTROOT
testpool                192M    112K
➥192M          0%   ONLINE     -
bash-3.00# zpool status
pool: testpool
state: ONLINE
scrub: none requested
config:
NAME    STATE     READ WRITE CKSUM
testpool ONLINE    0     0     0
mirror   ONLINE    0     0     0
c0d1     ONLINE    0     0     0
c1d1     ONLINE    0     0     0
errors: No known data errors
```

Os comandos acima possuem alguns pontos interessantes, que merecem ser comentados:

- o pool pode ter qualquer nome que o usuário deseje lhe dar. Usamos, neste caso, o nome `testpool`;
- pode-se criar o pool com qualquer configuração de RAID. Para RAID 0, não especifique nada. Para RAID 1 especifique `mirror` e para RAID 5 especifique `raidz`;
- o overhead do pool é mínimo (112 kb), sobrando muito espaço disponível para uso;
- podemos verificar a saúde do pool (coluna `health`) utilizando o comando `zpool list` ou seu respectivo `status` usando o comando `zpool status`, assim como possíveis erros de checksum e estatísticas de leitura e escrita. Os estados possíveis de disco ou pool são *ONLINE*, *DEGRADED*, *FAULTED*, *OFFLINE*, *UNAVAILABLE*.

Vale notar também que, se possuíssemos um sistema de arquivos (ufs, por exemplo) construído nos discos que usamos, deveríamos ter inserido a opção `-f`, de modo a forçar a criação. Observe o comando abaixo:

```
bash-3.00#  zpool create testpool
➥-f  mirror <disco1> <disco2>
```

O leitor pode também verificar se algum dentre os pools disponíveis apresenta problemas com o comando:

```
bash-3.00# zpool status -x
```

Como já temos nosso pool (*testpool*), podemos criar quantos filesystems forem necessários neste espaço:

```
bash-3.00# zfs create testpool/
➥zfs1
bash-3.00# zfs create testpool/
➥zfs2
bash-3.00# zfs create testpool/
➥zfs3
bash-3.00# zfs list
```

```
NAME          USED  AVAIL  REFER
➥MOUNTPOINT
testpool      186K   160M
➥22K  /testpool
testpool/zfs1    18K   160M    18K
➥/testpool/zfs1
testpool/zfs2    18K   160M    18K
➥/testpool/zfs2
testpool/zfs3    18K   160M    18K
➥/testpool/zfs3
```

Vale a pena ressaltar alguns aspectos desse último exemplo:

- todos os sistemas ZFS ocupam o mesmo espaço do pool testpool (160 Mb), tendo um overhead muito pequeno;
- todos também são montados automaticamente;
- o processo de criação de filesystems é tão fácil quanto criar diretórios. Aliás, a montagem do sistema de arquivos é feita em uma hierarquia (`/testpool/zfs1`) que é criada automaticamente. Não é preciso, em nenhum momento, usar o comando `mkdir` para criar tal ponto de montagem.

## Gerenciamento via Web do ZFS

O ZFS apresenta uma boa surpresa para quem é apreciador de interfaces gráficas e deseja administrar os filesystems ZFS (pools, discos, etc.) através de uma interface web: é o seu gerenciador de discos online (**figura 1**).

Basta digitar a seguinte URL no seu browser:

- https://localhost:6789/zfs

O login deve ser realizado, em seguida, como usuário *root*. A **figura 2** mostra a tela inicial da administração via browser do ambiente ZFS.

Se não for possível acessar esta interface de gerenciamento, deve-se verificar se o servidor da interface está no ar:

```
bash-3.00# /usr/sbin/smcwebserver
➥status
```

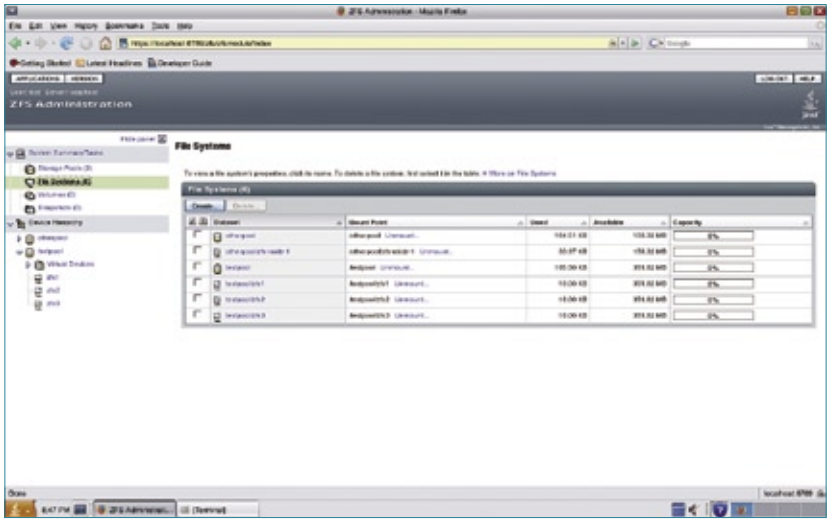

**Figura 2** Tela inicial do aplicativo via mavegador do sistema de arquivos *ZFS*.

Se o servidor está fora do ar, é possível inicializá-lo da seguinte forma:

```
bash-3.00# /usr/sbin/smcwebserver
➥start
```

É conveniente manter o mesmo servidor ativo para os próximos boots:

```
bash-3.00# /usr/sbin/smcwebserver
➥enable
```

## Renomeando filesystems

Vez ou outra são utilizados nomes não muito apropriados para a criação de um sistema de arquivos, e isso pode prejudicar uma administração mais fácil no ambiente Solaris. Para renomear um sistema de arquivos ZFS, no entanto, basta utilizar o seguinte comando:

```
bash-3.00# zfs rename testpool/
➥zfs3 testpool/linuxmagazine
```

## Apagando pools e filesystems

Já que aprendemos a criar pools e filesystems e alterar suas propriedades, também devemos aprender a apagá-los. Começando pelo ZFS, podemos apagar elementos do sistema – ou o próprio sistema – assim:

```
bash-3.00# zfs destroy testpool/
➥zfs1
bash-3.00# zfs destroy testpool/
➥zfs2
bash-3.00# zfs destroy testpool/
➥linuxmagazine
```

Em algumas situações, não será possível destruir o filesystem já que ele pode estar sob o status `busy`. Neste caso, pode ser repitido o mesmo comando com a opção `-f` :

```
bash-3.00# zfs destroy -f
➥testpool/linuxmagazine
```

## ZFS no Linux

O ZFS já foi portado para o Linux e o dono deste bom trabalho é Ricardo Correa. O projeto usa a tecnologia *FUSE* (*Filesystem on Userspace*), que foi introduzida no kernel 2.6.14 e que trabalha muito bem com uma implementação do NTFS chamada *ntfs-3g*.

O projeto ZFS/FUSE ainda está no início e tem alguns pequenos problemas: entretanto este é o caminho natural pelo qual todo software passa. O autor não está preocupado ainda com questões de performance, mas está trabalhando para deixar o código suave, com todas

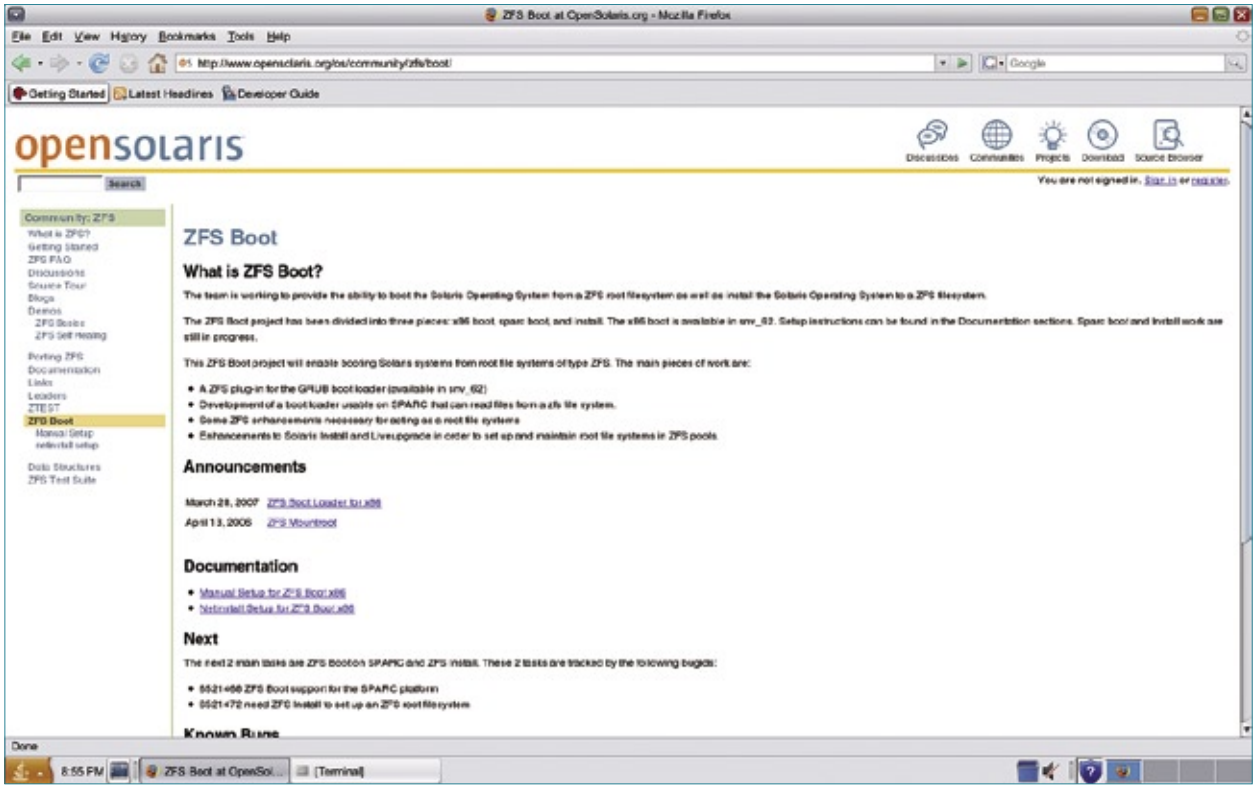


**Figura 3** Página oficial do *ZFS Boot*.

as características do ZFS funcionando bem, para somente depois pensar nos aspectos de velocidade.

Implementar o ZFS no Linux (Fedora 7 é utilizado nos passos seguintes), é muito simples. No site do ZFS on Linux/FUSE, faça o download do código-fonte mais recente, o qual no momento em que este artigo foi escrito é o arquivo *zfs-fuse-0.4.0_beta1.tar.bz2*[2].

Feito isto, siga os seguintes passos para compilar e instalar o ZFS on Linux/FUSE:

```
bash-3.00# yum install fuse-*
bash-3.00# yum install scons
bash-3.00# cd /
bash-3.00# bunzip2 zfs-fuse-0.4.0_
➥beta1.tar.bz2
bash-3.00# tar xvf zfs-fuse-0.4.0_
➥beta1.tar.bz2
bash-3.00# cd /zfs-fuse-0.4.0_
➥beta1/src
bash-3.00# scons
bash-3.00# scons install
```

Execute, a seguir, o daemon responsável pelo serviço de ZFS:

```
bash-3.00# cd /zfs-fuse-0.4.0_
➥beta1/src/zfs-fuse
bash-3.00# ./run.sh
```

## Conclusão

A Sun realmente tem desenvolvido muitos projetos interessantes e, sem dúvidas, o ZFS é um deles. Outros projetos estão em curso, como o próprio Opensolaris, onde é possível fazer download do próprio código fonte do Solaris e compilar uma versão particular [3]. Outra boa iniciativa é o projeto *OpenSparc*, que traz a versão aberta do processador T1 da Sun, que opera com 8 cores [4].

O ZFS não era suportado no disco de boot do Solaris 10 – ou seja, o filesystem que estava sob os arquivos e diretórios de sistema ainda devia ser UFS. Essa restrição, porém, já foi contornada e o leitor pode verificar o procedimento correto desta implementação em [5] (**figura 2**). Outra aplicação muito interessante é o *ZFS Test Suite*, que verifica e testa todas as características do ZFS e traz um detalhado relatório [6].

Sem dúvidas, o ZFS é um filesystem que veio para ficar. A tendência é que ele fique cada vez mais maduro, mais sólido e robusto, tanto em sua plataforma original – o Solaris – quanto em suas plataformas "por adoção", como o Linux. ■

## Mais Informações


[1] Whitepaper da Sun sobre o ZFS: http://www.sun.com/software/whitepapers/solaris10/fs_performance.pdf

[2] ZFS on FUSE/Linux: http://zfs-on-fuse.blogspot.com/

[3] Código fonte do Solaris: http://www.opensolaris.org/os/downloads/on/

[4] OpenSparc: http://www.opensparc.net/

[5] Alterações no boot do Solaris 10: http://www.opensolaris.org/os/community/zfs/boot/

[6] ZFS Test Suite: http://www.opensolaris.org/os/community/zfs/zfstestsuite/