

Criando máquinas virtuais de alta performance.

Multiplicando Pingüins

O conceito de distribuir os recursos de um único computador em diversas instâncias de software não é recente. A origem da virtualização de computadores vem da época dos mainframes, computadores de grande porte destinados a rodar várias instâncias de um sistema operacional, cada qual atendendo a um usuário ou finalidade específica.



POR MARCO ANTONIO BOCARDO SINHORELI

Diferentemente de outros softwares destinados a dividir seu hardware, o Xen consegue, de maneira muito tranquila, dar a ilusão de que existem diversas máquinas em sua rede, com praticamente o mesmo desempenho de uma máquina Linux real. Estudos comparativos demonstram que o Xen, em diversas situações, é superior em desempenho a softwares comerciais e Open Source, como você pode comprovar no site oficial do programa [1].

O Xen é um monitor de máquina virtual para hardware x86, desenvolvido e mantido pela Universidade de Cambridge, capaz de suportar a execução de diversos sistemas operacionais sem precarizar ou prejudicar o desempenho e com isolamento total entre as máquinas virtuais, garantindo assim maior segurança. O Xen, atualmente na versão 2.0.5 (estável), tem suporte às versões 2.4.x e 2.6.x do kernel Linux.

A comunidade tem contribuído bastante no que diz respeito à criação de versões para outros sistemas operacionais. Atualmente existem versões para o NetBSD, FreeBSD e Plan 9 [2]. A versão para Windows XP está sendo baseada numa versão instável (ainda em desen-

volvimento) do Xen e não está disponível por causa de duas restrições de licenciamento do software.

Recursos do Xen

O Xen necessita de um “pré-kernel” no Linux hospedeiro para disponibilizar as máquinas virtuais. No momento do boot, o GRUB descompacta esse pré-kernel e define os parâmetros relativos à fatia de memória que o kernel linux da máquina hospedeira irá gerenciar. Esse espaço reservado para o kernel privilegiado destina-se ao uso pelos serviços gerenciados do Xen e outros inicializados pelo *init*. Cada máquina virtual (ou *domínio*) tem seu próprio arquivo de configuração (sob `/etc/xen/auto` – falaremos dele mais adiante), podendo haver variações no hardware disponível (processador, memória, swap, e discos).

O Xen utiliza por padrão o *scheduler* BVT. Para que você entenda melhor, o scheduler é usado para atribuir o peso correto para cada máquina virtual no processador central, conforme o peso atribuído para o domínio. O processador central nunca estará inativo se houver tarefas a serem executadas por um domínio. O BVT faz uso do “tempo virtual” para

tomar decisões de como deverá ocorrer a distribuição dos processos dos domínios no processador. Cada domínio tem um tempo e uma fatia do processador associada em nano-segundos.

O *Atropos Scheduler* (`sched=atropos`, no arquivo de configuração) é um scheduler de tempo real leve. Ele garante o compartilhamento absoluto da CPU, com a vantagem de compartilhar o tempo ocioso do processador central quando ele não está em uso. Ao usar o *Atropos*, não o configure para utilizar 100% ou mais do processador. Tente utilizar sempre uma quantidade menor que a capacidade existente para garantir um comportamento regular.

O *Round Robin Scheduler* (opção `sched=rrobin`) está incluído como uma demonstração simples de uma API interna de scheduler do Xen e não é indicado para uso em produção.

Disponibilizando VBDs (Virtual Block Devices) para as máquinas virtuais

É possível exportar diretamente de *Dom0* (a máquina hospedeira) dispositivos de blocos para os outros domínios, utilizando protocolos de rede padrão (NBD, iSCSI,

NFS etc), além de volumes e partições físicas para prover o diretório raiz para as máquinas virtuais.

Exportar diretamente partições de *Dom0* para os outros domínios é simples: os arquivos de configuração dos domínios utilizam a *tag phy*: para especificar um dispositivo físico a ser exportado. Exemplo da linha a ser utilizada:

```
disk = ['phy:hda3,sda1,w']
```

Com isso, dizemos que */dev/hda3* será o volume físico a ser exportado para o domínio em modo de leitura e escrita como */dev/sda1*.

Qualquer disco ou partição considerado como um dispositivo de bloco pelo Linux poderá ser exportado para os domínios virtuais. Por exemplo, se temos discos iSCSI ou volumes NBD montados em *Dom0*, eles poderão ser exportados para os domínios utilizando a sintaxe *phy*:. Veja um exemplo:

```
disk = ['phy:vg/lvm1:sda2:w']
```

Utilizando "loop devices" como VBD

É possível criar em *Dom0* os chamados *loop disks* arquivos que são montados e tratados como discos rígidos reais para serem usados como dispositivo de armazenamento das máquinas virtuais. O primeiro passo é criar os tais *loop disks*. Por exemplo, para criar um disco de 1 GB digite o comando:

```
# dd if=/dev/zero of=/loopdisk bs=1k \
seek=1024k count=1
```

Após isso será necessário formatá-lo:

```
# mkfs -t ext3 loopdisk
```

Caso o comando solicite confirmação, responda **Y**. O arquivo de configuração das máquinas virtuais ficará assim:

```
['file:/caminho/para_o/arquivo/loopdisk,\
sda1,w']
```

Existem limitações que poderão influenciar na decisão de qual tipo de armazenamento usar. Um deles é a alta requisição I/O por parte das máquinas virtuais no dispositivo físico de *Dom0* onde está o *loop disk*. Outro detalhe: por padrão, o kernel Linux define que o número máximo de *loop devices* que podem ser montados simultaneamente é oito. Esse limite é definido pelo parâmetro *max_loop* do módulo *loop.o*. Caso desejemos mais VBDs, devemos providenciar algumas alterações. O meio mais fácil é compilar o módulo *loop.o* diretamente no kernel (opção *CONFIG_BLK_DEV_LOOP* na configuração do kernel) e passar o parâmetro *max_loop=n* (onde *n* é o número máximo de *loop devices* montados) para o kernel pelo *bootloader*.

Utilizando volumes LVM como VBDs

Um volume LVM (*Logical Volume Manager*) é uma solução bastante agradável para uso como VBD, principalmente por possibilitar o redimensionamento dinâmico do volume, permitindo o ajuste do espaço em disco disponível de acordo com a necessidade. Para inicializar uma partição com suporte a um PV (*Physical Volume* – Volume físico) LVM digite:

```
# pvcreate /dev/sda6
```

Com o comando abaixo, crie um VG (*Volume Group*) com um nome qualquer, por exemplo *vg*, na partição física:

```
# vgcreate vg /dev/sda6
```

Se desejar inserir um outro disco ou partição no grupo digite:

```
# pvcreate /dev/sdb1
# vgcreate vg /dev/sdb1
```

Assim as duas partições pertencerão ao mesmo VG e serão vistas como um único dispositivo.

Agora crie um LV (*Logical Volume* – Volume lógico) com o tamanho desejado, neste exemplo 4 GB:

```
# lvcreate -L4096M -n vmdisk1 vg
```

Vamos formatá-lo com o comando:

```
# mkfs -t ext3 /dev/vg/vmdisk1
```

Agora basta indicar o VBD no arquivo de configuração do domínio virtual:

```
disk = [ 'phy:vg/vmdisk1,sda1,w' ]
```

Você pode clonar volumes LVM. Isto é possível desde o kernel 2.6.8; no entanto, esta versão possui um bug, já corrigido nas mais recentes. Para utilizar esse recurso basta digitar:

```
# lvcreate -s -L1024M -n clonedisk1 \
/dev/vg/myvmdisk1
# lvcreate -s -L1024M -n clonedisk2 \
/dev/vg/myvmdisk1
```

Assim criamos dois clones de 1 GB do volume */dev/vg/myvmdisk1*. Se tiver de redimensioná-los use o utilitário *lvextend*:

```
# lvextend +100M /dev/vg/clonedisk1
```

Nesse caso acrescentamos mais 100 MB ao volume */dev/vg/clonedisk1*.

Usando compartilhamentos NFS como VBDs

Inicialmente será necessário um sistema de arquivos exportado em um servidor NFS. Isso pode ser feito adicionando a linha a seguir ao arquivo */etc/exports*:

```
/exports/rootvm1 1.2.3.4/24 (rw,sync,\
no_root_squash)
```

Finalmente, no arquivo de configuração da máquina virtual (veja [listagem 2](#)), indique que deseja usar um compartilhamento NFS como partição de root :

```
root = '/dev/nfs'
nfs_server = '1.1.1.1'
nfs_root = '/exports/rootvm1'
```

A máquina virtual precisa de acesso à rede no momento do boot. Para isso você deve atribuir a ela um endereço IP, seja manualmente (com os utilitários `ip`, `netmask`, `gateway` e `hostname`) ou automaticamente via DHCP, com o parâmetro `dhcp = 'dhcp'` no arquivo de configuração da máquina virtual.

É importante lembrar que um sistema de arquivos raiz montado via NFS tem problemas de estabilidade sob uma alta carga de requisições (e esse problema não é específico do Xen). Nesse caso, em aplicações de missão crítica o uso de NFS não é indicado.

Um cenário de uso

Imagine que você precisa disponibilizar diversos serviços em dez servidores diferentes sendo que cada servidor teria, conforme o serviço que será fornecido, uma certa quantidade de memória e espaço em disco. Considere também que na maior parte do tempo os processadores destes servidores estarão ociosos. Além disso seus servidores precisarão estar ligados à rede, mas precisam também ter um certo nível de segurança.

Tradicionalmente esse cenário exigiria grande investimento em infraestrutura de servidores e outros ativos de rede, o que poderia inviabilizar a execução de seu projeto. Mas se em vez de várias máquinas tivermos um único servidor com mais poder de processamento e bastante memória e espaço em disco, o Xen poderá tranquilamente satisfazer esse cenário com maior qualidade, mais segurança e muita economia de recursos. O

investimento em infraestrutura sofreria uma redução drástica, possibilitando a implantação da solução.

O cenário acima descrito necessita dos seguintes itens:

- servidores Apache com 5 GB de espaço em disco e 256MB de RAM.
- servidor MySQL com 3 GB de espaço em disco e 256MB de RAM.
- servidor PostgreSQL com 2 GB de espaço em disco e 512MB de RAM.
- servidor SMTP com 3 GB de espaço em disco e 512MB de RAM.
- servidor POP3/IMAP com 10 GB de espaço em disco e 512MB de RAM.
- servidor LDAP com 2 GB de espaço em disco e 256 de RAM.

Teremos que dimensionar o Xen para que cada máquina virtual forneça a quantidade adequada de espaço em disco e memória. Teremos ainda que providenciar para que o sistema Linux hospedeiro, que gerenciará as máquinas virtuais, tenha quantidade de memória e espaço em disco suficientes para a tarefa. Precisaremos da seguinte configuração: para a memória das máquinas virtuais, (256 MB x 7) + (512 MB X 3), ou 3328 MB de RAM, e (5 GB x 5) + (2 GB x 2) + (3 GB x 2) + 10 GB = 45 GB de espaço em disco. Além disso, precisaremos de 512 MB de RAM e 5 GB de espaço em disco para o sistema hospedeiro. Ou seja, precisaremos de uma máquina com 3.75 GB de RAM e 50 GB de espaço em disco para implementar nosso cenário.

Como teremos concorrência entre as várias instâncias no que diz respeito ao uso do processador, a melhor alternativa é disponibilizar um servidor capaz de suportar a alta latência de requisições sem degradação dos serviços. Para o cenário anteriomenre descrito, utilizaremos o scheduler padrão do Xen, o BTV, para que o próprio virtualizador faça dinamicamente a distribuição dos processos em execução em todas as máquinas virtuais.

Se seus requisitos de hardware forem menores que os apresentados, será possível implementar tranquilamente uma solução com hardware menos potente. Os processadores suportados pelo Xen são os da família i686 ou mais recentes (Intel Pentium Pro, Celeron, Pentium II, Pentium III, Pentium IV, Xeon, e AMD Athlon ou Duron). Máquinas com dois ou mais processadores são suportadas e há suporte básico à tecnologia *Hyper-Threading* (SMT) da Intel. Atualmente o Xen opera somente em modo 32 bits, mas uma versão específica para a arquitetura x86/64 (usada nos processadores Opteron, da AMD) está sendo providenciada e uma para a arquitetura Itanium está em vias de conclusão (ambas são arquiteturas de 64 Bits). Estuda-se ainda adicionar suporte a outras arquiteturas, como PowerPC e ARM, no momento adequado.

Preparando-se para a multiplicação

O primeiro passo para a multiplicação dos pingüins é fazer o download dos fontes da versão estável mais recente do Xen (em [\[3\]](#)). No momento em que este artigo foi escrito, o Xen encontrava-se na versão 2.0.5. Precisaremos ainda resolver algumas dependências antes de iniciar a compilação do programa.

Para escrever este artigo, usei o Debian GNU/Linux. Encontrei no site do projeto, na sessão FAQ [\[4\]](#), a lista de tudo o que precisaria instalar antes de dar início à compilação do Xen. Os pacotes no Debian são: `make`, `gcc`, `libc6-dev`, `zlib1g-dev`, `python`, `python-dev`, `python-twisted`, `bridge-utils`, `iproute`, `libcurl3`, `libcurl3-dev`, `bzip2`, `module-init-tools`, `latex`, `latex2html`, `transfig` e `tgif`. O comando é:

```
# aptitude install make gcc libc6-dev 2
zlib1g-dev python python-dev iproute 2
python-twisted bridge-utils libcurl3 2
libcurl3-dev bzip2 module-init-tools 2
latex latex2html transfig tgif
```

O script de instalação providenciará os fontes do kernel, que você deverá indicar no `Makefile`. Nesse arquivo, considere que `xen0` se refere ao kernel do sistema Linux hospedeiro e `xenU` é referente ao kernel da máquina virtual. O `U` após `xen` significa *Unprivileged Kernel* (kernel sem privilégios). Exemplo:

```
# Para kernels da série 2.6.x
KernelS ?= linux-2.6-xen0 linux-2.6-xenU
# Para kernels da série 2.4.x
KernelS ?= linux-2.4-xen0 linux-2.4-xenU
```

Podemos ainda fazer variações colocando uma versão do kernel em `xen0` e outra em `xenU`, de acordo com suas necessidades. Rode o comando `make` para compilar o programa:

```
# make install-twisted all install
```

Isso vai colocar em `/boot` os kernels do hospedeiro e das máquinas virtuais. Podemos também personalizar o kernel do Xen para satisfazer alguma necessidade específica. Para o kernel das máquinas virtuais digite:

```
# cd linux-2.6.10-xenU
# make ARCH=xen menuconfig
```

Para o kernel hospedeiro digite:

```
# cd linux-2.6.10-xenU
# make ARCH=xen menuconfig
```

Configurando o gerenciador de boot

Infelizmente, para os fãs do LILO, o GRUB é, até o momento, o único gerenciador de boot que suporta as opções do Xen usadas para determinar a quantidade de memória para o sistema hospedeiro, entre outras coisas. A entrada de nosso hospedeiro no arquivo de configuração do GRUB (que fica em `/boot/grub/menu.lst`) fica assim:

```
title Xen 2.0 / XenLinux 2.6.10
  root      (hd0,1)
  kernel /boot/xen.gz dom0_mem=524288
  module /boot/vmlinuz-2.6.10-xen0
  root=/dev/hda2 ro console=tty0
```

Vamos entender as coisas. A linha:

```
kernel /boot/xen.gz dom0_mem=524288
```

Diz que `/boot/xen.gz` é o kernel de baixo nível do Xen, o "pré-kernel" mencionado anteriormente. Ele é utilizado para levantar o kernel privilegiado de `dom0` (o hospedeiro), que é o monitor Xen. A opção `dom0_mem=524288` define a quantidade de memória que o hospedeiro Xen irá utilizar, neste caso 512 MB. Já a linha:

```
module /boot/vmlinuz-2.6.10-xen0 root=
/dev/hda2 ro console=tty0
```

O parâmetro `module /boot/vmlinuz-2.6.10-xen0` define o kernel que o monitor (sistema Linux hospedeiro) irá utilizar. `root=/dev/hda2 ro` define a partição raiz de `dom0`, montada inicialmente no modo `ro` (*read-only*, somente para leitura). O parâmetro `console=tty0` especifica o terminal de I/O (entrada/saída) padrão do Xen. Veja mais opções em [6].

Para usar o Xen com kernels da série 2.6.x, é necessário desabilitar o suporte a TLS (*Thread Local Storage*) antes de carregar o kernel hospedeiro. Para isso mova o diretório `/lib/tls` para `/lib/tls.disabled`. A equipe de desenvolvimento do Xen vem trabalhando juntamente com os mantenedores das distribuições Linux para que essa incompatibilidade com as bibliotecas TLS seja resolvido.

Criando as máquinas virtuais

Para criar o sistema de arquivos raiz para nossas máquinas virtuais, usaremos o utilitário `debootstrap`. Instale-o em sua máquina (`apt-get install debootstrap`) e digite o seguinte:

```
# mkdir /usr/src/rootfs
# debootstrap --arch i386 sarge
/usr/src/rootfs http://ftp.br.debian.org/debian
```

Teremos, então, um esqueleto de um sistema básico Debian em `/usr/src/rootfs`. Será agora necessário copiar os módulos compilados do kernel `xenU` para o novo sistema de arquivos:

```
# cp -ax /lib/modules/2.6.10-xenU
/usr/src/rootfs/lib/modules
```

Conforme dissemos anteriormente, a solução com mais flexibilidade para criar os VBDs, ou discos, das máquinas virtuais é o LVM. Caso o hospedeiro use um kernel da série 2.4 use o pacote `lvm10`, mas você não terá acesso ao recurso de clonagem de discos. Caso use um kernel da série 2.6 opte pelo `lvm2`. Em nosso exemplo, `/dev/sda`, que tem 40 GB, conterá as partições do hospedeiro e parte do volume LVM. O restante ficará em um volume idêntico, chamado `/dev/sdb`. Ative os dispositivos físicos que vamos usar:

```
# pvcreate /dev/sda4
# pvcreate /dev/sdb1
# pvcreate /dev/sdb2
```

A seguir criaremos dois *Volume Groups*, um para os discos e outro para o swap:

```
# vgcreate vgdisk /dev/sda4 /dev/sdb1
# vgcreate vgsnap /dev/sdb2
```

Agora chegou a hora de criar os volumes lógicos. Fizemos um script (veja a [listagem 1](#)) para automatizar a tarefa de criação dos volumes para os cinco servidores Apache.

Agora criaremos os volumes lógicos para o servidor MySQL, que tem 3 GB de espaço em disco, e copiaremos para ele o sistema de arquivos "esqueleto" criado com o `debootstrap`:

Listagem 1: Criando volumes lógicos

```
#!/bin/bash

for ((v=1;v<=5;v++))
do
# Cria o volume lógico
lvcreate -L5120M -n apache$v vdisk
# Formata o volume em ext3
mkfs -t ext3 /dev/vdisk/apache$v
# Monta o volume em /mnt
mount -t ext3 /dev/vdisk/apache$v /mnt
# Copia os esqueleto criado com o debootstrap para o volume
cp -ax /usr/src/rootfs/* /mnt
# Desmonta o volume
umount /mnt
# Cria o volume para swap
lvcreate -L256M -n apache$v vgsdap
# Formata o volume recém-criado como área de swap
mkswap /dev/vgsdap/apache$v
done
```

```
# lvcreate -L3072M -n mysql vdisk
# lvcreate -L256M -n mysql vgsdap
# mkfs -t ext3 /dev/vdisk/mysql
# mkswap /dev/vgsdap/mysql
# mount -t ext3 /dev/vdisk/mysql /mnt
# cp -ax /usr/src/rootfs/* /mnt
# umount /mnt
```

Arquivos de configuração das máquinas virtuais

O arquivo de configuração do *daemon* do Xen, `xend-config.sxp` fica no diretório `/etc/xen/`, junto com os subdiretórios `auto` e `scripts`, além de dois arquivos de exemplo. Entre outros parâmetros, nesse arquivo é definida a porta da interface web para controle do daemon (`xend-port 8000`), os hosts que poderão se conectar a essa interface (`xend-address 'localhost'`, deixe vazio para não restringir o acesso), o modo de funcionamento das interfaces de rede virtuais (roteamento ou *bridge*), as definições de segurança para as interfaces de rede (*antispoof*) e o uso de scripts para *file-backend block device* e *enbd*.

No diretório `auto` ficam os arquivos de configuração das máquinas virtuais que serão inicializadas automaticamente no momento em que o script `/etc/init.d/xendomains` for executado. O diretório `scripts` contém os arquivos de scripts do daemon Xen.

Veja na **listagem 2** o arquivo de configuração de nossa primeira máquina virtual, que vai ser usada como servidor Web rodando o Apache. Para as demais máquinas, considere os valores declarados em nosso cenário inicial, lembrando que esses arquivos de configuração deverão estar no diretório `/etc/xen/auto`, se considerarmos que as máquinas virtuais deverão iniciar automaticamente quando o script `/etc/init.d/xendomains` for executado. O **quadro 1**, na página ao lado, resume os principais comandos que podem ser usados para operar o Xen.

Conclusão

Apesar de a instalação e configuração do Xen apresentarem um certo nível de dificuldade, o sistema não deixa nada a desejar do ponto de vista da otimização dos recursos computacionais. Conforme mencionamos neste artigo, em comparação com outros aplicativos ou sistemas de virtualização o Xen,

Listagem 2: arquivo de configuração de uma máquina virtual

```
# Define o Kernel do domínio
kernel = "/boot/vmlinuz-2.6.10-xenU"
# Define o Sistema Operacional do domínio
builder='linux'
# Quantidade de memória
memory = 256
# Nome deste domínio
name = "Apache1"
# -l determina a auto-alocação de CPU para este domínio
cpu = -1
# Número de interfaces de rede deste domínio
nics = 1
# Define o Mac Address e a bridge para este domínio. Se o Mac não for configurado, um número hexadecimal é gerado aleatoriamente.
vif = [ 'mac=aa:00:00:00:00:01, bridge=xen-br0' ]
# VBDs para este domínio.
disk = [ 'phy:vgdisk/apache1,sda1,w', 'phy:vgsdap/apache1,sda2,w' ]
# Configuração de rede. É possível fazê-la automaticamente via DHCP.
# dhcp="dhcp"
# "rootdisk", passado no momento do boot para o kernel.
root = "/dev/sda1 ro"
# Run level
extra = 2
restart = 'onreboot'
```

pelo fato de trabalhar mais próximo ao hardware, utiliza todos os seus recursos – o sistema, como vimos, lança mão de um pré-kernel para fazer chamadas de baixo nível e realizar a comunicação entre os dispositivos do sistema. Isso ocorre a partir do *boot*, quando os parâmetros do *grub* para fatiar a memória para o *Xen0* (Linux hospedeiro/monitor) são passados. As funções então já alocadas na memória do pré-kernel possibilitam a criação das demais máquinas virtuais.

Para quem busca uma solução de virtualização livre e robusta, com alto desempenho e qualidade, o uso do Xen é, sem dúvida, mais que indicado. ■

INFORMAÇÕES

- [1] Comparativos de desempenho entre o Xen e outras máquinas virtuais:
www.cl.cam.ac.uk/Research/SRG/netos/xen/performance.html
- [2] Plan 9 - Um Sistema Operacional Unix mantido pela Bell Labs:
www.cs.bell-labs.com/plan9dist
- [3] Linux Magazine Brasil, 2ª. edição, Página 38, *Jogue seus dados*.
- [4] Download do Xen:
www.cl.cam.ac.uk/Research/SRG/netos/xen/downloads.html
- [5] Pacotes do Debian necessários para o Xen:
tinyurl.com/ce688
- [6] Manual do Usuário do Xen:
www.cl.cam.ac.uk/Research/SRG/netos/xen/readmes/user/user.html#s:xboot
- [7] Página oficial do Xen:
www.cl.cam.ac.uk/Research/SRG/netos/xen
- [8] The Unnofficial Xen Wiki:
xen.terrabox.com/
- [9] Linux Magazine Brasil, 7ª. Edição, Página 19, *IBM abre código de virtualização*.
- [10] Xen na Wikipedia:
en.wikipedia.org/wiki/Xen
- [11] Máquinas virtuais:
en.wikipedia.org/wiki/Virtual_machine
- [12] Comparativo entre máquinas virtuais:
en.wikipedia.org/wiki/Comparison_of_virtual_machines

Quadro 1: comandos relacionados ao Xen

Iniciar, parar, verificar o estado ou reiniciar o daemon Xen, ou forçar a re-leitura de seus arquivos de configuração do programa:

```
# /etc/init.d/xend {start|stop|status|restart|reload|force-reload}
```

Iniciar, parar ou verificar o estado dos domínios cujos arquivos de configuração podem ser encontrados no diretório `/etc/xen/auto`

```
# /etc/init.d/xendomains {start|stop|status}
```

Argumentos do xm (*Xen monitor*)

call	Chama as funções da API Xen.
Help	Mostra a ajuda do comando xm.

Comandos relacionados aos consoles das máquinas virtuais

console	Abre um console na máquina virtual. ex: <code>xm console apache1</code>
consoles	Solicita informações sobre as máquinas virtuais.

Comandos dos domínios

balloon	Configura memória nos domínios utilizando o driver <i>balloon</i> .
create	Cria um domínio.
destroy	Encerra a execução de um domínio imediatamente.
domid	Converte o nome do domínio em seu ID.
domname	Converte o ID de um domínio em seu nome.
list	Mostra informações sobre os domínios.
maxmem	Configura o limite de memória para o domínio.
migrate	Migra um domínio para outras máquinas.
pause	Pausa a execução de um domínio.
pincpu	Configura o PIN de um domínio para o uso da CPU.
restore	Reinicia a execução de um domínio a partir de seu estado salvo.
save	Salva o estado de um domínio em seu arquivo de configuração.
shutdown	Desliga um domínio.
sysrq	Envia um <i>sysrq</i> para o domínio.
unpause	Retoma a execução de um domínio pausado.

Comando relacionados ao host xen (hospedeiro)

dmesg	Lê ou limpa as mensagens no buffer do Xen.
Info	Solicita informações sobre o host Xen.
Log	Imprime o log do daemon <i>xend</i> .

Comandos de controle dos schedulers

atropos	Configura parâmetros do scheduler atropos.
bvt	Configura parâmetros do scheduler BVT.
bvt_ctxallow	Dá permissão de "context switch" ao scheduler BVT.
rrobin	Configura o scheduler <i>round robin</i> .

Comandos relacionados aos Virtual Block Devices

vbd-create	Cria um novo VBD para um domínio.
Vbd-destroy	Destrói um VBD de um domínio.
Vbd-list	Lista os VBDs de um domínio.

Comandos relacionados às interfaces de rede virtuais

vif-list	Lista as interfaces de rede virtuais.
----------	---------------------------------------