

Monte um cluster de servidores VMWare

Alta rotatividade

O que acontece se duas máquinas virtuais num hospedeiro VMWare utilizarem recursos demais? Numa máquina limitada, ambos os hóspedes sofrerão uma lentidão. Mas, se tivermos um cluster de hospedeiros, essa situação pode ser praticamente eliminada.

por Bruno Gola

Rafael Rosa – www.sxc.hu



A idéia de utilizar o VMWare [1] como um software para um ambiente de testes de sistemas operacionais ou teste de produtos é maravilhosa. Mas quando esse ambiente acaba sendo tanto de homologação quanto de produção e desenvolvimento, se tudo não for muito bem planejado, pode acabar virando um pesadelo.

Em uma empresa onde são necessárias várias máquinas virtuais ligadas ao mesmo tempo, algumas rodando servidores web e servidores de aplicação, outras servindo como desktops de produção, apenas um servidor VMWare não vai resolver o problema. É aí que surge a necessidade de outro servidor para também fazer o serviço de hospedeiro de máquinas virtuais. Mas deixar separados dois servidores que desempenham a mesma tarefa não faz sentido. Afinal de contas, imagine que você tenha uma máquina virtual que precise ser ligada, mas sua máquina hospedeira está sem recursos. Você só terá duas opções: ou você desliga uma das máquinas virtuais já ligadas, ou espera. Pode ser que a sua outra máquina esteja com recursos suficientes para executar mais uma máquina virtual, porém, transferir os dados da máquina virtual de uma para a outra é uma solução um tanto ineficiente, já que ficar transferindo máquinas toda vez que faltarem recursos não parece realmente eficiente. Portanto, a solução pode ser a criação de um cluster de máquinas virtuais.

O cluster

Quando me refiro a clusters, este seria um servidor controlando alguns “rich clients”. O servidor principal guarda todas as máquinas virtuais em seu disco, por exemplo, em `/home/vmware/VirtualMachines/` (o usuário `vmware` é utilizado como um usuário administrador das máquinas virtuais). O diretório `/home` é exportado via NFS para as outras máquinas hospedeiras VMWare, que não precisam de disco rígido, já que a inicialização é feita a partir da máquina principal, via rede, usando o padrão PXE. Assim, a máquina principal guarda o kernel e o diretório raiz (`/`) de cada máquina do cluster, e exporta-os via NFS no momento da inicialização. As outras máquinas só utilizam processamento e memória. Na verdade, você pode usar o disco dessas máquinas e montar uma partição, por exemplo, em `/work`, para facilitar o processo de criação de máquinas. Nesse caso, a máquina é criada em `/work`, que é o disco físico da máquina, e depois transferida para o disco do cluster. Essa solução ainda diminui a carga de entrada e saída (leitura e escrita) na máquina principal e na rede. O restante do disco pode e deve ser usado para `swap`.

Hardware

O hardware utilizado consistia de servidores Dell com processador Intel Pentium 4 3,20 GHz, 4 GB de memória e interface de rede Gigabit Ethernet PCI Express, outros servidores Dell com processador Intel Pentium 4 3,00 GHz, 4 GB de memória e interface

de rede Gigabit Ethernet PCI Express. Na versão que utilizamos do `PXElinux`, tivemos alguns problemas com a placa de rede `Broadcom BCM5751` na hora da inicialização remota, portanto tivemos de usar a solução do **quadro Inicialização sem suporte a PXE**. O servidor que controla os outros é um Dell com processador Intel Pentium 4, 3,20 GHz, 4 GB de memória e interface de rede Gigabit Ethernet PCI Express, com 4 discos de 100 GB configurados com `RAID 10` (interface `LSI Logic MegaRAID`).

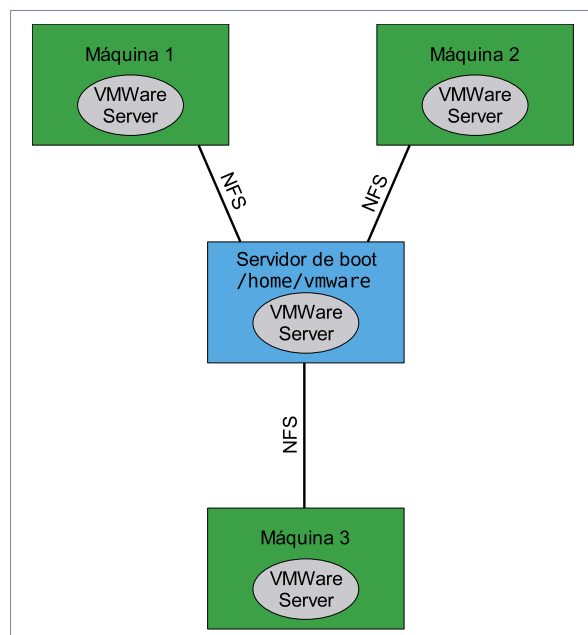


Figura 1 Diagrama da relação entre os servidores.

Inicialização sem suporte a PXE

Quando a inicialização pelo padrão PXE não é possível, podemos utilizar o carregador de inicialização (o *GRUB*, nesse exemplo) como alternativa.

Tendo um sistema básico instalado, basta passar as opções corretas para o kernel, para que este monte o diretório raiz via rede, utilizando NFS. O trecho a seguir mostra a configuração do GRUB para fazer isso:

```
title Debian GNU/Linux, kernel 2.6.13-5-486
root (hd0,0)
kernel /vmlinuz root=/dev/nfs nfsroot=192.168.0.101:/home/cluster/
rootfiles/01-00-12-3f-76-83-94 ip=192.168.0.103::192.168.0.1:
255.255.255.0:pushkin:eth0:off rw devfs=nomount mem=4096m 26
savedefault
boot
```

Com essa configuração, é possível rodar até 28 máquinas simultaneamente (sete por servidor), cada uma com uma média de 256 MB de RAM. Dependendo do tipo de máquina necessária, pode-se rodar até mais máquinas por servidor.

Configurando o ambiente

Para a montagem do cluster, será necessário um servidor *DHCP* e o *TFTP* para a inicialização via rede, além de um servidor NFS. A **figura 1** mostra a relação entre os servidores.

Em um servidor *Debian*, os serviços necessários podem ser instalados da seguinte forma:

```
apt-get install dhcp3-common dhcp3-server
atftpd atftpd nfs-kernel-server
```

O **exemplo 1** mostra uma configuração simples para o servidor DHCP. Note que é importante ter o endereço MAC de cada interface de rede das máquinas hospedeiras VMWare, para que na hora da inicialização cada um encontre seus arquivos (seu diretório raiz).

O parâmetro *next-server* define qual servidor TFTP deve ser utilizado para a inicialização; já o *filename* informa qual arquivo deve ser utilizado para essa inicialização. É possível fazer essa definição individualmente para cada máquina, bastando colocar dentro da definição de *host*. É importante notar também o parâmetro *ignore unknown-clients*, que faz com que o nosso servidor DHCP não interfira no resto da rede.

Algumas máquinas possuem placas de rede que não suportam a inicialização via PXE; nesse caso, existe uma solução utilizando o disco apenas com um carregador de inicialização e um sistema básico instalados, montando o diretório raiz usando NFS (**quadro 1**).

A configuração do servidor TFTP é bastante simples, e é armazenada no arquivo */etc/default/atftpd* (no *Debian*). Um exemplo desse arquivo seria:

```
USE_INETD=false
OPTIONS="--daemon --port 69
--retry-timeout 5 --mcast-port 1758
--mcast-addr 192.168.0.0-255 --mcast-ttl 1
--maxthread 100 /home/cluster/bootfiles"
```

A opção mais importante é a última, que especifica o caminho que servirá como raiz para o TFTP (no caso, */home/cluster/bootfiles*).

Por último, devemos configurar o servidor NFS para que ele exporte o diretório raiz e o */home* para os servidores hospedeiros. A configuração do NFS é feita no arquivo */etc/exports*:

```
/home/cluster/rootfiles/01-00-14-22-7a-7d-3c
-> host1(rw,no_root_squash,async)
/home/cluster/rootfiles/01-00-12-3f-76-83-94
-> host2(rw,no_root_squash,async)
/home/cluster/rootfiles/01-00-12-3f-76-81-92
-> host3(rw,no_root_squash,async)
/home/ host1(rw,no_root_squash,async)
-> host2(rw,no_root_squash,async)
-> host3(rw,no_root_squash,async)
```

Note que os endereços MAC e as máquinas são os mesmos que foram configurados no servidor DHCP. Não se esqueça de colocar no arquivo */etc/hosts.allow* as linhas:

```
portmap: 192.168.0.102 192.168.0.103
-> 192.168.0.104
lockd: 192.168.0.102 192.168.0.103
-> 192.168.0.104
rquotad: 192.168.0.102 192.168.0.103
-> 192.168.0.104
mountd: 192.168.0.102 192.168.0.103
-> 192.168.0.104
statd: 192.168.0.102 192.168.0.103
-> 192.168.0.104
```

para liberar para as máquinas a montagem dos diretórios */e* e */home* via NFS.

Preparando para a inicialização

No nosso exemplo, a estrutura de diretórios pode ser a seguinte:

```
/home/cluster/bootfiles - arquivos de boot
-> (pxelinux.0, pxelinux.cfg/...)
/home/cluster/rootfiles - diretório raiz de
-> cada máquina (host1/, host2/, host3/, ...)
/home/vmware/VirtualMachines/ - máquinas
-> virtuais compartilhadas entre as
-> máquinas do cluster
/work/ - diretório local para instalação
-> de máquinas
```

Para gerarmos o diretório raiz de cada máquina, usamos o comando *debootstrap*:

```
apt-get install debootstrap
debootstrap --verbose --resolve-deps
-> "sarge"
-> "/home/cluster/rootfiles/default/"
-> "http://ftp.br.debian.org/debian/"
```

No comando acima, *sarge* é a versão do Debian, */home/cluster/rootfiles/default/*

Exemplo 1: dhcpd.conf

```
option domain-name "dominio.net";
option routers 192.168.0.1;
option subnet-mask 255.255.255.0;
option domain-name-servers 192.168.0.1;
```

```
ignore unknown-clients;
```

```
allow booting;
allow bootp;
next-server 192.168.0.101;
filename "pxelinux.0";
```

```
host host1 {
  hardware ethernet 00:14:22:7A:7D:3C;
  fixed-address 192.168.0.102;
  server-name "host1";
}
```

```
host host2 {
  hardware ethernet 00:12:3F:76:83:94;
  fixed-address 192.168.0.103;
  server-name "host2";
}
```

```
host host3 {
  hardware ethernet 00:12:3F:76:81:92;
  fixed-address 192.168.0.104;
  server-name "host3";
}
```

```
subnet 192.168.0.0 netmask 255.255.255.0 {
}
```

é o diretório onde serão colocados os arquivos, e <http://ftp.br.debian.org/debian/> é o repositório a ser usado para fazer o download dos arquivos.

Feito isso, basta copiar o diretório `default/` para o endereço MAC da interface de cada máquina e, para facilitar, criar um link simbólico para o hostname da máquina, ficando:

```
ln -s /home/cluster/rootfiles/
01-00-14-22-7a-7d-3c /home/cluster/
rootfiles/host1
ln -s /home/cluster/rootfiles/
01-00-12-3f-76-83-94 /home/cluster/
rootfiles/host2
ln -s /home/cluster/rootfiles/
01-00-12-3f-76-81-92 /home/cluster/
rootfiles/host3
```

Uma dica importante nesse ponto é que a máquina central já tenha todos os pacotes e atualizações necessários no momento dessa cópia. Caso contrário, será necessário refazer esses mesmos procedimentos para cada uma das máquinas.

Para especificarmos o diretório raiz de cada máquina no cluster, criamos dentro do diretório `/home/cluster/bootfiles/pxelinux.cfg/` um arquivo para cada máquina e um diretório dentro de `/home/cluster/bootfiles/`, que armazenará a imagem do kernel de cada máquina. O nome, tanto do arquivo quanto do diretório, deve ser o endereço MAC da interface de rede de cada máquina. Para facilitar, você pode criar um link simbólico, por exemplo:

```
ln -s /home/cluster/bootfiles/
01-00-14-22-7a-7d-3c /home/cluster/
bootfiles/host1
```

Um exemplo do conteúdo do arquivo `/home/cluster/bootfiles/pxelinux.cfg/01-00-14-22-7a-7d-3c:`

```
label linux
kernel 01-00-14-22-7a-7d-3c/bzImage
append vga=normal root=/dev/nfs
nfsroot=192.168.0.101:/home/cluster/
rootfiles/01-00-14-22-7a-7d-3c
ip=192.168.0.102:192.168.0.1:
255.255.255.0:host1:eth0:off
rw devfs=nomount mem=4080m
```

Configurando o VMWare

Com o cluster montado e funcionando, basta configurar o VMWare para cada máquina. Na verdade não existe nenhuma configuração especial; basta fazer uma

instalação comum do VMWare, especificando o diretório das máquinas virtuais como `/home/vmware/VirtualMachines/`. A **figura 2** mostra um console do VMWare sendo executado, conectado em algum dos servidores, que está com seis máquinas ativas, entre elas Windows®, SCO UNIX® e Solaris.

Conclusão

Utilizar uma solução de cluster torna bastante prático o uso de máquinas virtuais em ambientes de produção, homologação e desenvolvimento. Outras funcionalidades podem ser implementadas, tais como políticas de autenticação centralizada para as máquinas virtuais, ou a criação de uma política de “dono e grupo” para cada máquina virtual. É de grande utilidade a criação de scripts usando-se o comando `vmware-vmx` para verificar a situação das máquinas virtuais em cada hospedeiro. O **exemplo 2** mostra um script de exemplo que utiliza o `vmware-vmx` para verificar quais máquinas estão rodando em quais hospedeiros. Essa solução ajuda bastante a organizar a sua rede de máquinas virtuais e facilita muito o uso do VMWare em um ambiente com várias máquinas. ■

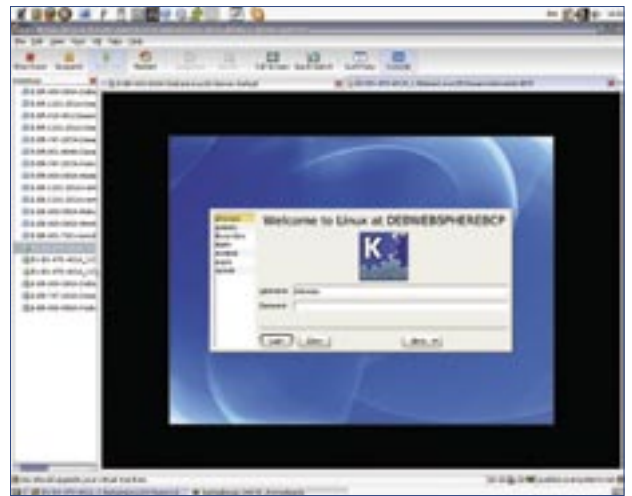


Figura 2 Servidor VMWare rodando diversas máquinas virtuais.

Mais Informações

[1] VMWare: <http://www.vmware.com>

O autor

Bruno Gola (brunogola@gmail.com) tem 19 anos, cursa o segundo ano de Ciência da Computação no Centro Universitário Senac e trabalha na área de infra-estrutura de projetos na empresa Eversystems. Utiliza exclusivamente Software Livre há 2 anos. Nas horas vagas, escuta uma boa música, toca guitarra e mergulha.

Exemplo 2: Script para verificação de localização das máquinas

```
vmwarecmd="/usr/local/lib/VMWare.com/VMware-server-1.0.0-28343/bin/vmware-cmd"
list="/home/vmware/scripts/maquinas_ativas/VirtualMachines_ativas.txt"
vmfile="/etc/vmware/vm-list"
vmware_servers="host0 host1 host2 host3"

> $!st

for host in $vmware_servers
do
echo -e "#####\n${host}\n#####\n" >>${list}

ssh -n -T -o BatchMode=yes vmware@${host} cat /etc/vmware/vm-list
| grep ^config | sed -e 's/Virtual Machines/VirtualMachines/g' | cut -d" " -f2-10
| sed -e 's//g' | sed -e 's/ /\\ /g' | sed -e 's/ /\\ /g' | \
while read machine
do
estado=$(ssh -n -T -o BatchMode=yes vmware@${host} ${vmwarecmd} ${machine} getstate)
estado=${estado#getstate() = }

if [ $estado == 'on' ]
then
echo -e "${machine}" >> ${list}
fi
done
echo "" >> ${list}
done
```