1 Obsluha virtualizace

Představte si prázdnou serverovnu – abychom mohli administrovat unixový systém, potřebujeme *stroje*, na kterých poběží. Systém zabere nějaké místo na *discích* a budeme ho chtít připojit k *síti*. Chudí studenti si nemohou dovolit skutečný hardware, takže musí virtualizovat – škola poskytla čtyři stroje {potemkin, gromoboj, bogatyr, navarin}.ms.mff.cuni.cz, na které se lze přihlásit loginem a heslem ze SISu (pomocí univerzitního LDAPu). Je na nich Gentoo Linux s podporou KVM, čtyři sdílené disky přes dvě 8 Gb/s *Fibre Channel* linky (/virtlab/) a userspace dostatečně bohatý pro pohodlné používání.

1.1 Spuštění virtuálního stroje

Program qemu(1) standardně spouští GTK+ okno (zprovozněte si X11 forwarding přes SSH) a nepoužívá virtualizační infrastrukturu KVM. Parametry určují konfiguraci virtuálního stroje – pište si je do nějakého skriptu, ať je nemusíte opakovat.

```
# export Q="qemu-system-x86_64 -machine accel=kvm"
# ${Q}
```

1.2 Vstup a výstup

Grafický výstup můžete z GTK+ změnit parametrem -display.

```
# ${Q} -display curses
```

QEMU má ovládací konzoli zvanou „monitor“, dostupnou standardně pomocí Ctrl+Alt+2, jinak pomocí -monitor.

```
# ${Q} -monitor stdio
# ${Q} -monitor tcp::4444,server,nowait
# netcat
# ${Q} -monitor telnet::4444,server,nowait
# telnet
(qemu) info kvm
kvm support: enabled
```

Sériová konzole je praktický způsob ke správě libovolného množství strojů (skriptovatelné) i ladění jádra (netřeba paketový přenos jako u USB, příchozí data vyvolají přerušení a jsou snadno dostupná) přes pomalé/jednoduché spoje („stačí tři dráty“ z COM portu do záložního stroje s baterií a GPRS modemem). Rychlejší reakce uživatelského rozhraní znamenají spokojenějšího admina :-)

1.2.1 Doporučené použití ve sdíleném prostředí

TCP/UDP/telnet socket není chráněn před ostatními, UNIX socket není uživatelsky přívětivý (tab-completion), ale můžeme se naučit pracovat s terminály.

```
CONSOLE="-serial pty -monitor pty"
# ${Q} ${CONSOLE} 2> muj_stroj.txt
char device redirected to /dev/pts/14 (label compat_monitor0)
char device redirected to /dev/pts/15 (label serial0)
# cu -l /dev/pts/14
~.
# minicom -p /dev/pts/14
```

cu(1): Zadání ∼. (před tildou třeba zmáčknout Enter) vypne SSH klienta (vhodné, když zamrzne spojení). Podle sekce ESCAPE CHARACTERS v ssh(1) lze sekvenci bezpečně napsat jako ∼∼.
minicom(1) má Emacs-like ovládání (Ctrl+a, x).

1.3 Disky

Virtuální disk umí vytvořit program qemu-img(1).

```
# qemu-img create stroj1_disk1.img 11G
# du stroj1_disk1.img
# stat stroj1_disk1.img
```

Formát qcow2 podporuje copy-on-write snapshoty.

```
# qemu-img create -f qcow2 stroj1_disk2.img 11G
# qemu-img info stroj1_disk1.img
# qemu-io stroj1_disk1.img
write -P 0xDE 65536 1024
flush
quit
# qemu-img map stroj1_disk1.img
```

1.4 CD-ROM

Na vytvořený disk nainstalujeme OpenBSD. K instalaci z CD je třeba monitor (minimálně na přepnutí bootloaderu na sériovou konzoli pomocí set tty com0 v boot.conf(5)), potom stačí -display none.

```
CDROM="-cdrom /virtlab/1/__instalacky__/OpenBSD/*.iso"
HDD="-drive file=stroj1_disk1.img,if=virtio,media=disk"
# ${Q} ${CONSOLE} ${HDD} ${CDROM}
```

Kromě -display none se bude hodit -daemonize.

1.5 Snapshoty

Změna původního souboru rozbije jeho snapshoty!

```
# qemu-img create -f qcow2 -b base.img snapshot.img
# qemu-img info snapshot.img
# ${Q} -drive file=snapshot.img
```

Pokud nás nezajímá výsledek (chceme jen něco vyzkoušet), QEMU umí snapshot vytvořit a hned po použití smazat.

```
# ${Q} -drive file=base.img -snapshot
```

1.6 Síť

Připojíme virtuál do připraveného switche se sítí „Backbone“. Tomuto síťovému rozhraní přiřadíme virtuální síťový hardware, který má MAC adresu.

```
# ${Q} -device help
# ${Q} -device virtio-net-pci,help
BACKBONE_NET="-netdev vde,id=backbone"
BACKBONE_NIC="-device virtio-net-pci,netdev=backbone,
    mac=D0:CE:1A:B1:BE:C"
# ${Q} ${BACKBONE_NET} ${BACKBONE_NIC}
```

Chceme-li vlastní síť, pořídíme si manageovatelný switch.

```
# vde_switch -sock /tmp/pelikanm_switch1
help
# ... (démonizovat)
SITE1_NET="-netdev vde,id=site1,sock=/tmp/pelikanm_swi
SITE1_NIC="-device virtio-net-pci,netdev=backbone,
    mac=02:00:00:AA:BB:CC"
# ${Q} ${SITE1_NET} ${SITE2_NIC}
```

Rozšířit VDE síť na víc strojů znamená na každém spustit switch a spojit je pomocí vde_plug(1).

1.7 ostatní

- suspend/hibernate/resume
- VDE switch web interface
- -incoming + migrate* příkazy v monitoru
- -root pro chroot(2)
- -runas pro privsep
- -readconfig vm.ini, -writeconfig vm.ini vs. psaní parametrů ručně

Našli jste chybu? Pošlete ji na pelikan@storkhole.cz!