Taking full advantage of QEMU in Xen

Daniel P. Berrange <berrange@redhat.com>

Xen Summit, November 2007

redhat

Xen userspace in 3.1.x

A large number of components

- Libraries: libxenstore, libxenctrl, libxenguest
- Daemons: xend, xenconsoled, xenstored, blktapctrl
- Services: tapdisk, xen-vncfb, xen-sdlfb, qemu-dm
- Helpers: xc_save, xc_restore, pygrub

redhat

Xen userspace in 3.2.x

Still large, but less code duplication

- Xen-vncfb and xen-sdlfb removed
- LibVNCServer removed as dependancy
- Xenconsoled only required for hypervisor logging
- QEMU-DM can serve paravirt text console backend
- QEMU-DM can serve paravirt graphics console backend

redhat

Outstanding issues

Picking a few issues with Xen userspace

- PV bootloader is not exposed via graphical console
- HVM cannot do direct kernel boot
- save/restore requires co-ordination of 3 separate processes
- Overhead of python runtime causes projects to fork Xen userspace
- Code duplication for disk formats I/O in blktap & QEMU

How can these be addressed ? With QEMU !

redhat

Comparison to KVM

Everything is handled by QEMU

- Start a guest by launching QEMU
- Stop a guest by killing QEMU
- Control via the QEMU monitor console
- Paravirt driver backends within QEMU

Could Xen userspace be this simple ? Yes !

redhat

HVM direct kernel boot

- Linux protected mode lives as 0x100000
- HVM firmware also lives at 0x100000 :-(
- 2.6.20/.22 add relocatable kernels for i386/x86_64
- Load protmode kernel at 0x200000 instead
- Non-relocatable kernels need helper in code32_switch
- Helper moves protmode kernel to 0x100000
- Successfully boot F7 and F8 i386 installer kernels

redhat

Spawning domains in QEMU

Extend QEMU xenpv & xenfv machine types to

- Use libxc to create a domain
- Load kernel (or firmware) into domain
- Set HVM params/shared info page
- Write domain info at /local/domain/[ID]
- Write console, fb, disk & net front/back-end data
- Start guest execution
- Sig handler to destroy guest on TERM/INT/QUIT

No XenD required. Only xenstored & qemu-dm.

redhat

Paravirt bootloader

- Xenpv machine spawns pygrub bootloader
- Connects psuedo-TTY to pygrub and QEMU graphical console
- Bootloader is visible over SDL and VNC
- Create guest using kernel/initrd returned by pygrub
- PXE boot via the QEMU tap device

redhat

Libvirt integration

- Libvirt QEMU driver manages QEMU, KQEMU, KVM guests.
- Add 'Xen' guests via the enhanced 'qemu-xen' binary
- stop/save/save/restore via monitor commands
- disk_add/disk_remove monitor commands for device hotplug
- Hypercalls for CPU usage & pinning

redhat

Example: PV + direct kernel

A paravirt guest, with direct boot to installer, 500 mb ram, SDL graphics and text console to a psuedo-TTY

```
qemu-xen \
    -name demo \
    -M xenpv \
    -hda /var/lib/xen/images/demo.img \
    -m 500 \
    -kernel /root/fedora8.vmlinux \
    -initrd /root/fedora8.initrd \
    -serial pty
```

redhat

Example: PV + bootloader

A paravirt guest, with automatic pygrub bootloader against hda, 500 mb ram, VNC graphics and text console logging to a file

```
qemu-xen \
    -name demo \
    -M xenpv \
    -hda /var/lib/xen/images/demo.img \
    -m 500 \
    -serial file:/var/log/xen/demo.log \
    -vnc 0.0.0.0:1
```

redhat

Example: FV + direct kernel

A fullvirt guest, with direct boot to installer, 500 mb ram, no graphics and a serial port to a psuedo-TTY, installer via serial port

```
qemu-xen \
    -name demo \
    -M xenfv \
    -hda /var/lib/xen/images/demo.img \
    -m 500 \
    -kernel /root/fedora8.vmlinux \
    -initrd /root/fedora8.initrd \
    -append “console=ttyS0”
    -serial pty \
    -nographics
```

redhat

Future work

- Integrate KVM migrate framework with main QEMU
- Implement a Xen variant for the migrate framework
- ioctl() for privcmd to associate a PID with a DomID
- Kernel to destroy & cleanup domain when PID exits