Virtualisation Enhancements in Fedora 12

Release 12, the latest Fedora distribution that came out in November 2009, offers users a lot of new virtualisation features. The Dirty Dozen, as this set of virtualisation features is called, are targeted at improving security, better performance, more stability, a better platform for developers as well as some usability enhancements.

In this article, we'll look at the components that make up the virtualisation stack to run KVM-based virtual machines on Fedora. At the lowest level of the virtualisation stack, we have the Linux kernel that functions as the hypervisor. The QEMU project presents the device model, or the virtual computer to guest operating systems. The libvirt library manages various hypervisor and emulator backends and presents a consistent and stable API allowing user interface apps to interact with VMs. Finally, the virt-manager UI is used to manage the virtual machines.

This stack, comprising the Linux kernel, QEMU, libvirt and virt-manager is the recommended way of running virtual machines on top of a Fedora installation.

According to Fedora policy, all work happens in upstream projects. Fedora then automatically inherits all the goodies that come along with software updates. To enlist some of the new features that got introduced in Fedora 12 in relation to the Fedora 11 release, let's look at how these projects have evolved. The following table shows the version of the package first included in Fedora 11 and the version that's included in Fedora 12:

	Version in Fedora 11	Version in Fedora 12
Linux kernel	2.6.29	2.6.31
QEMU	0.10.4	0.11.0
libvirt	0.6.2	0.7.1
virt-manager	0.7.0	0.8.0

Note that Fedora 11 currently has kernel version 2.6.30 as part of the updates, and the other packages are updated to incorporate bug fixes from upstream releases, as well as user bug reports.

Let's now look at each of these components to identify the key changes that lead to a better experience for users.

The Linux Kernel: The key component for virtualisation in the Linux kernel is the KVM code that enables the Linux kernel to function as a hypervisor. The KVM changes include:

- Performance enhancements in the guest interrupt injection logic

- Performance enhancements in the guest SMP code
- Better support for huge pages
- Better emulation accuracy
- Support for MSI
- Fixes and improvements in the device assignment code

What this means is that guests will be more responsive, and there's more efficient interrupt handling for devices that are assigned from the host (PCI passthrough), as well as better stability.

For developers, there's improved guest debugging support and nested virtualisation on AMD hosts, which allows one to run a KVM guest within a KVM guest.

Changes in the generic kernel code that are useful for virtualisation include:

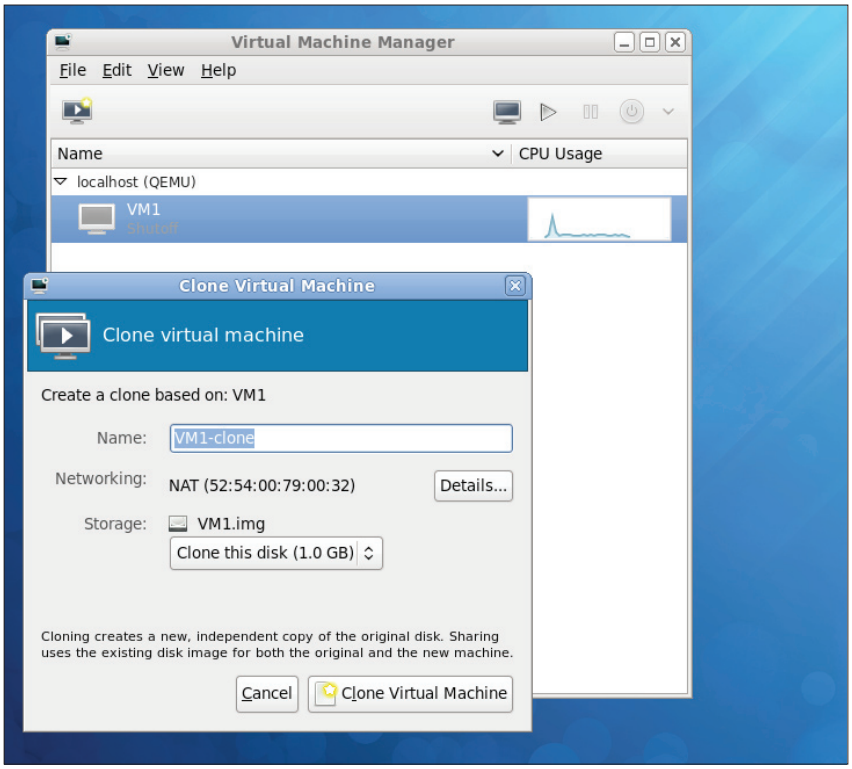
- SR-IOV (Single Root I/O Virtualisation) support
- KSM (Kernel SamePage Merging)
- VirtIO performance enhancements and fixes: block and net devices are much faster and more stable

SR-IOV support enables sharing of a physical PCI device among multiple guests. The performance while accessing such devices from the guests is at par with the performance of using the device from the host itself. This feature needs supported hardware. As of 2.6.31, the Intel 82576 Gigabit Ethernet Controller and the Neterion X3100 cards are supported. This feature needs driver support, both in the host as well as the guest. This feature builds on top of the device assignment code that was introduced in Fedora 11.

The KSM feature enables sharing of identical RAM contents to reduce physical memory usage by a host, improving the guest density. This is discussed in greater detail in a separate article titled "Introducing Kernel SamePage Merging" in this issue.

The device model (the virtual computer) that is presented to the guests is instantiated by QEMU. A lot of user-facing changes happen here. A short list of notable features that were introduced include:

- Installs guests from CD/DVD images located remotely over the HTTP protocol instead of having to download the images first.
- Shows the host CPU type as the guest CPU. This results in better utilisation of the host CPU features by guest applications, speeding up some of them that rely on newer features. However, enabling this feature may come at the cost of guest migration.
- Allows for the setting of migration speed and the setting of a maximum allowed downtime while live-migrating guests.
- Migration code has got a lot of fixes, which means more stable guests after live migrations.
- Many stabilisation fixes in the block code.
- The qcow2 file format saw a lot of performance and stabilisation fixes.
- Fixes and performance improvements in the virtio-block and virtio-net code.
- The ability to hot plug NICs in guests.
- A stable guest ABI, which ensures the hardware that the guest sees remains the same across QEMU upgrades. This is helpful especially for Windows guests, which may need reactivation on hardware changes.
- MSI support: the guests can now use MSI/MSI-X, resulting in better performance.
- Support for debugging guests.



Virt-manager in action

For developers, there's a lot of good news: a majority of the QEMU code was overhauled and it's in much better shape than earlier. It's now much easier to catch bugs at code review time. The development has also moved to a git tree from svn.

The libvirt development continued at a fast pace as well, with the addition of the following features, among others:

- Present a stable guest ABI
- NIC hot plug for guests
- Improved security by starting guests with lower privileges. Allows KVM guests to be started by unprivileged users.
- Ability to run guests with hugepage support: this helps reduce the number of memory pages in the host. This results in better performance by reduced

TLB misses and lesser RAM usage for page tables. However, hugepages carry the same restriction as with physical servers—if enabled, guest pages can't be swapped or ballooned.

- Enables hosts to discover new SAN storage, issue NPIV operations and present SAN storage to guests.
- Interfaces with netcf. netcf can configure bridges and network devices and pass them on to NetworkManager and libvirt. This will ease future integration of all network configuration in one place and allow one to use NetworkManager and place guests on bridges without hand-editing configuration files.
- Use cgroups to set resource limits on VMs
- Pin vcpus to physical CPUs: since VMs running under KVM are normal Linux processes, this can also be done via the command line.
- Support for creation of encrypted qcow2 images
- Compressed save image format
- Better device assignment support
- Clone storage used for guests

There are other features available as well, which don't involve KVM. Some of them include VirtualBox support, VMware ESX hypervisor support, Power Hypervisor support, etc.

The virt-manager UI has been greatly improved and leads to a better overall user experience. A few key features that are now available are:

- Improved UI, overhaul of the main manager GUI
- CPU pinning support
- System tray icon for easier access to VMs
- Ability to view and change security settings
- New VM clone wizard

Other improvements

In addition to the features mentioned above, there's also a switch from the etherboot option ROM to the gPXE-based option ROM. This allows a more stable network boot experience for virtual machines. This makes sense, because most etherboot development has moved to the gPXE project. Fedora will receive a more modern boot infrastructure, as well as quicker bug fixes.

Newly-introduced tools

All talk so far has revolved around new features that are available while actually running the virtual machines; now, a new set of tools and utilities has been added for examining guests when they're offline.

libguestfs is a library that allows access to and modification of guest disk images. guestfish is an interactive shell tool for editing guest disk images.

These tools allow users to perform a host of activities on guest disk images:

- Make batch configuration changes
- View and edit files inside guests without mounting the disk image
- Getting used/free stats for guest disks
- Performing partial guest backups
- Performing partial guest clones
- Changing settings/registry entries, etc
- Migrating between various virtualisation systems (physical to virtual, Xen to KVM, etc)

libguestfs can connect to libvirt, so existing guests can be directly referenced to within the guestfish shell. It also understands a host of file system formats—for example, ext2/3/4, btrfs, XFS, FAT, NTFS, LVM, etc.

The features presented here are not exhaustive. Moreover, there has been a lot of bug fixing in each of the projects mentioned, which makes the entire release very compelling for anyone to try out the new features and see where the future of virtualisation is headed. END

Ready reference table

Security	▪ QEMU processes started with lower privileges
Performance	▪ MSIX ▪ MSI-X support for assigned devices ▪ SR-IOV ▪ Huge-page backed memory ▪ KSM ▪ Qcow2 performance enhancements, stabilization
Developers	▪ qdev device model ▪ qemu-io ▪ Guest debugging
Stability	▪ Stable PCI bus addresses (Guest ABI)
Usability	▪ Pointing to iso images over http for guest installation ▪ Allow setting maximum downtime for migration ▪ NIC hotplug ▪ Automatic guest bridging with the netcf library ▪ Network booting with gPXE ▪ libguestfs for offline guest image viewing / editing

References

- Upstream project pages:
- Linux: *http://kernel.org*
- KVM: *http://www.linux-kvm.org*
- QEMU: *http://www.qemu.org*
- libvirt: *http://libvirt.org*
- virt-manager: *http://virt-manager.org*
- libguestfs: *http://libguestfs.org*
- Fedora 12 features and history of Fedora Virtualization: *https://fedoraproject.org/wiki/Virtualization/History*
- Interviews with some developers about new virtualisation features in Fedora 12: *http://www.fedoraproject.org/wiki/Virtualization_improvements_in_Fedora_12*

By: Amit Shah

The author is part of the virtualisation team at Red Hat and is excited to be a part of the technology that's rediscovering commodity x86 servers.