Name xVM, xvm – Solaris x86 virtual machine monitor

Description The Solaris xVM software (hereafter xVM) is a virtualization system, or *virtual machine monitor*, for the Solaris operating system on x86 systems. Solaris xVM enables you to run multiple virtual machines simultaneously on a single physical machine, often referred to as the *host*. Each virtual machine, known as a *domain*, runs its own complete and distinct operating system instance, which includes its own I/O devices. This differs from zone-based virtualization, in which all zones shares the same kernel. (See `zones`(5).)

Each domain has a name and a UUID. Domains can be renamed but typically retain the same UUID. A domain ID is an integer that is specific to a running instance. It changes on every boot of the guest domain. Non-running domains do not have a domain ID.

The xVM hypervisor is responsible for controlling and executing each of the domains and runs with full privileges. xVM also includes control tools for management of the domains. These tools run under a specialized control domain, often referred to as *domain 0*.

To run Solaris xVM, select the entry labelled `Solaris xVM` in the `grub`(5) menu at boot time. This boots the hypervisor, which in turn boots the control domain. The control domain is a version of Solaris modified to run under the xVM hypervisor. At the point at which the control domain is running, the control tools are enabled. In most other respects, the domain 0 instance behaves just like a "normal" instance of the Solaris operating system.

The xVM hypervisor delegates control of the physical devices present on the machine to domain 0. Thus, by default, only domain 0 has access to such devices. The other *guest* domains running on the host are presented with virtualized devices with which they interact as they would physical devices.

The xVM hypervisor schedules running domains (including domain 0) onto the set of physical CPUs as configured. The xVM scheduler is constrained by domain configuration (the number of virtual CPUs allocated, and so forth) as well as any run-time configuration, such as pinning virtual CPUs to physical CPUs.

The default domain scheduler in xVM is the credit scheduler. This is a work-conserving, fair-share domain scheduler; it balances virtual CPUs of domains across the allowed set of physical CPUs according to workload.

xVM supports two levels of virtualization. In the first, the operating system is aware that it is running under xVM. This level is called *paravirtualization*. In the second level, called *fully virtualized*, the guest operating system is not aware that it is running under xVM. A fully virtualized guest domain is sometimes referred to as a *Hardware-assisted Virtual Machine* (HVM).

With paravirtualization, each device, such as a networking interface, is presented as a fully virtual interface, and specific drivers are required for it. Each virtual device is associated with a physical device and the driver is split into two. A *frontend* driver runs in the guest domain and communicates over a virtual data interface to a *backend* driver. The *backend* driver currently

runs in domain 0 and communicates with both the frontend driver and the physical device underneath it. Thus, a guest domain can make use of a network card on the host, store data on a host disk drive, and so forth.

Solaris xVM currently supports two main split drivers used for I/O. Networking is done by means of the `xnb` (xVM Networking Backend) drivers. Guest domains, whether Solaris or another operating system, use `xnb` to transmit and receive networking traffic. Typically, a physical NIC is used for communicating with the guest domains, either shared or dedicated. Solaris xVM provides networking access to guest domains by means of MAC-based virtual network switching.

Block I/O is provided by the `xdb` (xVM Disk Backend) driver, which provides virtual disk access to guest domains. In the control domain, the disk storage can be in a file, a ZFS volume, or a physical device.

Using ZFS volumes as the virtual disk for your guest domains allows you to take a snapshot of the storage. As such, you can keep known-good snapshots of the guest domain OS installation, and revert the snapshot (using `zfs rollback`) if the domain has a problem. The `zfs clone` command can be used for quick provisioning of new domains. For example, you might install Solaris as a guest domain, run `sys-unconfig`(1M), then clone that disk image for use in new Solaris domains. Installing Solaris in this way requires only a configuration step, rather than a full install.

When running as a guest domain, Solaris xVM uses the `xnf` and `xdf` drivers to talk to the relevant backend drivers. In addition to these drivers, the Solaris console is virtualized when running as a guest domain; this driver interacts with the `xenconsoled`(1M) daemon running in domain 0 to provide console access.

A given system can have both paravirtualized and fully virtualized domains running simultaneously. The control domain must always run in paravirtualized mode, because it must work closely with the hypervisor layer.

As guest domains do not share a kernel, xVM does not require that every guest domain be Solaris. For paravirtualized mode and for all types of operating systems, the only requirement is that the operating system be modified to support the virtual device interfaces.

Fully-virtualized guest domains are supported under xVM with the assistance of virtualization extensions available on some x86 CPUs. These extensions must be present and enabled; some BIOS versions disable the extensions by default.

In paravirtualized mode, Solaris identifies the platform as `i86xpv`, as seen in the return of the following `uname`(1) command:

```
% uname -i
i86xpv
```

Generally, applications do not need to be aware of the platform type. It is recommended that any ISA identification required by an application use the `-p` option (for ISA or processor type) to `uname`, rather than the `-i` option, as shown above. On x86 platforms, regardless of whether Solaris is running under xVM, `uname -p` always returns `i386`.

You can examine the `domcaps` driver to determine whether a Solaris instance is running as domain 0:

```
# cat /dev/xen/domcaps
control_d
```

Note that the `domcaps` file might contain a string that is similar to, but does not exactly match, the preceding.

xVM hosts provide a service management facility (SMF) service (see `smf`(5)) with the FMRI:

```
svc:/system/xvm/domains:default
```

...to control auto-shutdown and auto-start of domains. By default, all running domains are shutdown when the host is brought down by means of `init 6` or a similar command. This shutdown is analogous to entering:

 # virsh shutdown *mydomain*

A domain can have the setting:

```
on_xend_stop=ignore
```

...in which case the domain is not shut down even if the host is. Such a setting is the effective equivalent of:

 # virsh destroy *mydomain*

If a domain has the setting:

```
on_xend_start=start
```

...then the domain is automatically booted when the xVM host boots. Disabling the SMF service by means of `svcadm`(1M) disables this behavior for all domains.

Solaris xVM is partly based on the work of the open source Xen community.

Control Tools

The control tools are the utilities shipped with Solaris xVM that enable you to manage xVM domains. These tools interact with the daemons that support xVM: `xend`(1M), `xenconsoled`(1M), and `xenstored`(1M), each described in its own man page. The daemons are, in turn, managed by the SMF.

You install new guest domains with the command line `virt-install`(1M) or the graphical interface, `virt-manager`.

The main interface for command and control of both xVM and guest domains is `virsh`(1M). Users should use `virsh`(1M) wherever possible, as it provides a generic and stable interface for controlling virtualized operating systems. However, some xVM operations are not yet implemented by `virsh`. In those cases, the legacy utility `xm`(1M) can be used for detailed control.

The configuration of each domain is stored by `xend`(1M) after it has been created by means of `virt-install`, and can be viewed using the `virsh dumpxml` or `xm list -l` commands. Direct modification of this configuration data is not recommended; the command-line utility interfaces should be used instead.

Solaris xVM supports live migration of domains by means of the `xm migrate` command. This allows a guest domain to keep running while the host running the domain transfers ownership to another physical host over the network. The remote host must also be running xVM or a compatible version of Xen, and must accept migration connections from the current host (see `xend`(1M)).

For migration to work, both hosts must be able to access the storage used by the `domU` (for example, over iSCSI or NFS), and have enough resources available to run the migrated domain. Both hosts must currently reside on the same subnet for the guest domain networking to continue working properly. Also, both hosts should have similar hardware. For example, migrating from a host with AMD processors to one with Intel processors can cause problems.

Note that the communication channel for migration is not a secure connection.

Glossary

backend
: Describes the other half of a virtual driver from the *frontend* driver. The backend driver provides an interface between the virtual device and an underlying physical device.

control domain
: The special guest domain running the control tools and given direct hardware access by the hypervisor. Often referred to as *domain 0*.

domain 0
: See *control domain*.

frontend
: Describes a virtual device and its associated driver in a guest domain. A frontend driver communicates with a *backend* driver hosted in a different guest domain.

full virtualization
: Running an unmodified operating system with hardware assistance.

guest domain
: A virtual machine instance running on a host. Unless the guest is domain 0, such a domain is also called a *domain-U*, where *U* stands for “unprivileged”.

host
The physical machine running xVM.

Hardware-assisted Virtual Machine (HVM)
A fully-virtualized guest domain.

node
The name used by the `virsh`(1M) utility for a host.

virtual machine monitor (VMM)
Hypervisor software, such as xVM, that manages multiple domains.

See Also `uname`(1), `dladm`(1M), `svcadm`(1M), `sys-unconfig`(1M), `virsh`(1M), `virt-install`(1M), `xend`(1M), `xenconsoled`(1M), `xenstored`(1M), `xm`(1M), `zfs`(1M), **Broken Link (Target ID: ATTRIBUTES-5)**, **Broken Link (Target ID: GRUB-5)**, **Broken Link (Target ID: LIVE-UPGRADE-5)**, **Broken Link (Target ID: SMF-5)**, **Broken Link (Target ID: ZONES-5)**

Notes For a physical NIC to be used to network guest domains, it must be GLDv3-compatible. If the interface is not marked as `legacy` in the output of `dladm show-link`, then it is compatible with xVM networking.

Currently, Live Upgrade (see `live_upgrade`(5)) is not supported under xVM. In order to Live Upgrade a host, you must boot into `i86pc` first.