How Does Xen Work?

Date: December 2009
Version: 1.0

Table of Contents

Executive Summary

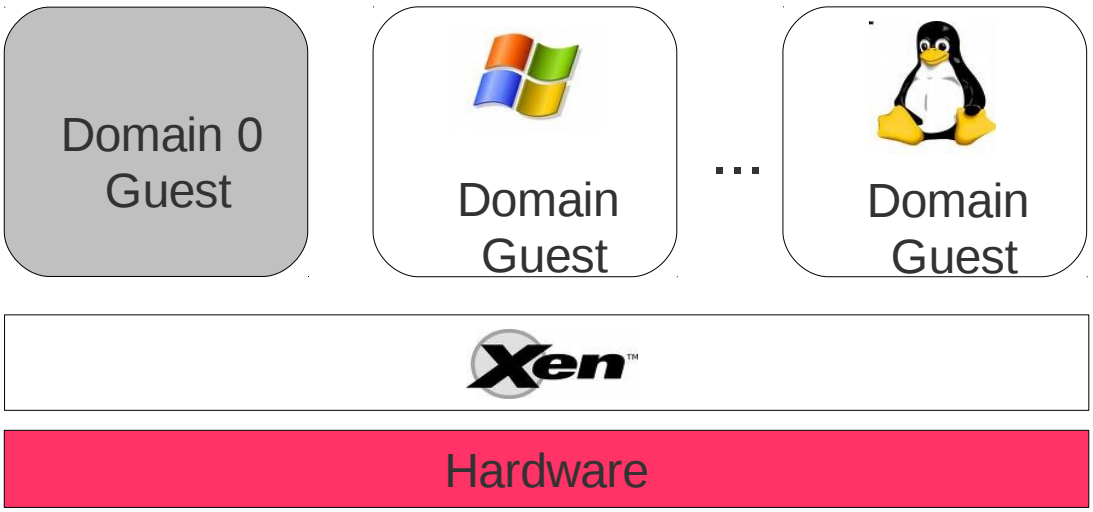
This document contains a high level architectural overview of the Xen hypervisor and the associated tools and applications necessary to support a complete virtualization environment. The architecture is based on Xen 3.4 released in late 2009 and is only an introduction to the overall Xen architecture. For a more complete description of the architecture please reference the many Xen books available.

Xen Environment Components

A Xen virtual environment consist of several items that work together to deliver the virtualization environment a customer is looking to deploy:

- Xen Hypervisor
- Domain 0 Guest
  - Domain Management and Control (Xen DM&C)
- Domain U Guest (Dom U)
  - PV Guest
  - HVM Guest

The diagram below shows the basic organization of these components.



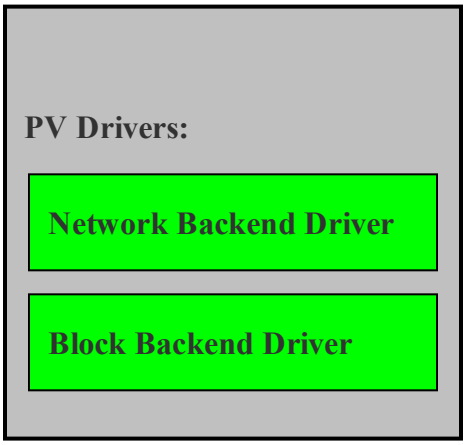
Xen Hypervisor

The Xen hypervisor is the basic abstraction layer of software that sits directly on the hardware below any operating systems. It is responsible for CPU scheduling and memory partitioning of the various virtual machines running on the hardware device. The hypervisor not only abstracts the hardware for the virtual machines but also controls the execution of virtual machines as they share the common processing environment. It has no knowledge of networking, external storage devices, video, or any other common I/O functions found on a computing system.

Domain 0

Domain 0, a modified Linux kernel, is a unique virtual machine running on the Xen hypervisor that has special rights to access physical I/O resources as well as interact with the other virtual machines (Domain U: PV and HVM Guests) running on the system. All Xen virtualization environments require Domain 0 to be running before any other virtual machines can be started.

Two drivers are included in Domain 0 to support network and local disk requests from Domain U PV and HVM Guests (see below); the Network Backend Driver and the Block Backend Driver. The Network Backend Driver communicates directly with the local networking hardware to process all virtual machines requests coming from the Domain U guests. The Block Backend Driver communicates with the local storage disk to read and write data from the drive based upon Domain U requests.

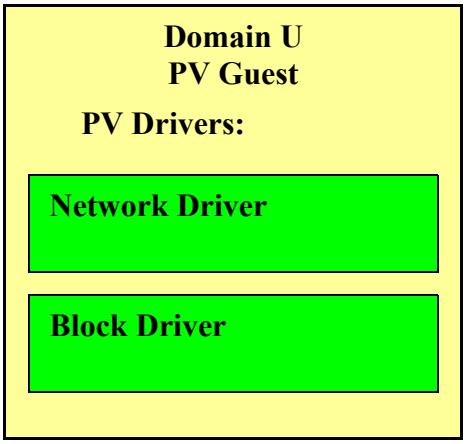


Domain U

DomainU guests have no direct access to physical hardware on the machine as a Domain0 Guest does and is often referred to as unprivileged. All paravirtualized virtual machines running on a Xen hypervisor are referred to as **Domain U PV Guests** and are modified Linux operating systems, Solaris, FreeBSD, and other UNIX operating systems. All fully virtualized machines running on a Xen hypervisor are referred to as **Domain U HVM Guests** and run standard Windows or any other unchanged operating system.

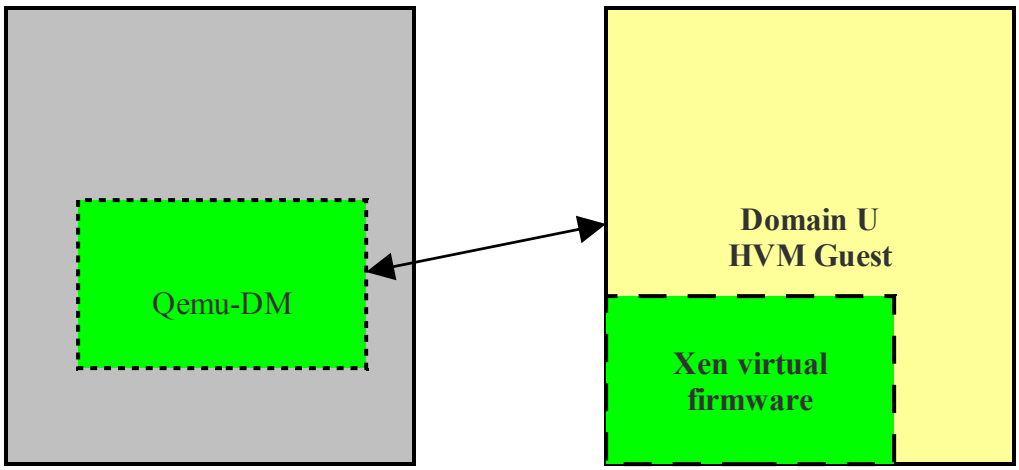
The Domain U PV Guest virtual machine is aware that it does not have direct access to the hardware and recognizes that other virtual machines are running on the same machine. The Domain U HVM Guest virtual machine is not aware that it is sharing processing time on the hardware and that other virtual machines are present.

A Domain U PV Guest contains two drivers for network and disk access, PV Network Driver and PV Block Driver.



A Domain U HVM Guest does not have the PV drivers located within the virtual machine; instead a special daemon is started for each HVM Guest in Domain 0, Qemu-dm. Qemu-dm supports the Domain U HVM Guest for networking and disk access requests.

The Domain U HVM Guest must initialize as it would on a typical machine so software is added to the Domain U HVM Guest, Xen virtual firmware, to simulate the BIOS an operating system would expect on startup. More information on the Xen virtual firmware is presented later in this document.

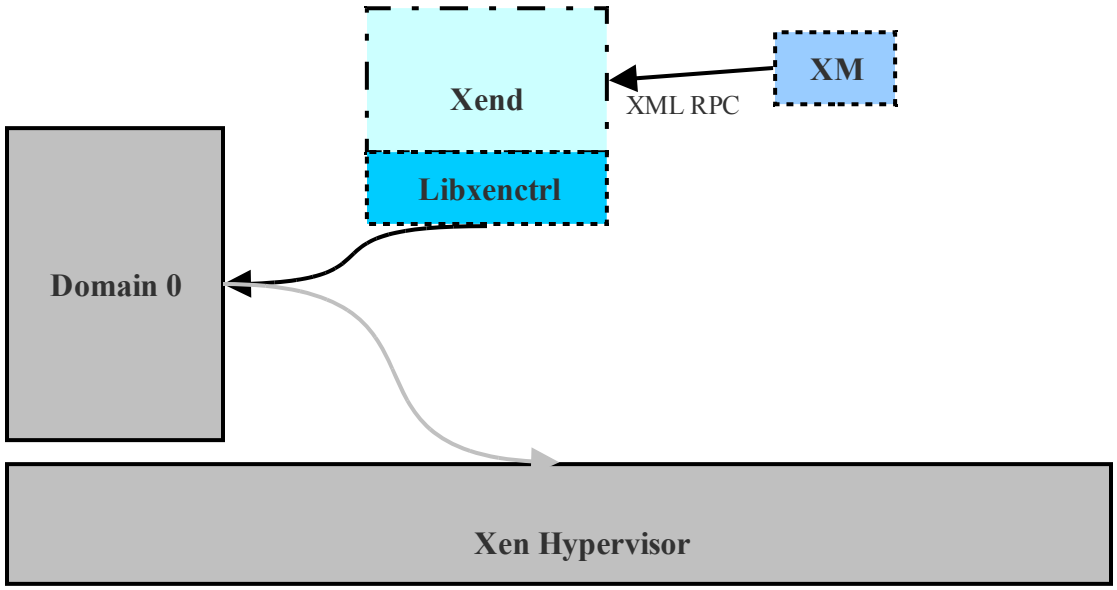


Domain Management and Control

A series of Linux daemons are classified as Domain Management and Control by the open source community. These services support the overall management and control of the virtualization environment and exist within the Domain 0 virtual machine. The diagrams below show the daemons outside the Domain 0 diagram for a clearer understanding of the architecture.

Xend

The Xend daemon is a python application that is considered the system manager for the Xen environment. It leverages the libxenctrl library (see below) to make requests of the Xen hypervisor. All requests processed by the Xend are delivered to it via an XML RPC interface by the Xm (see below) tool.



Xm

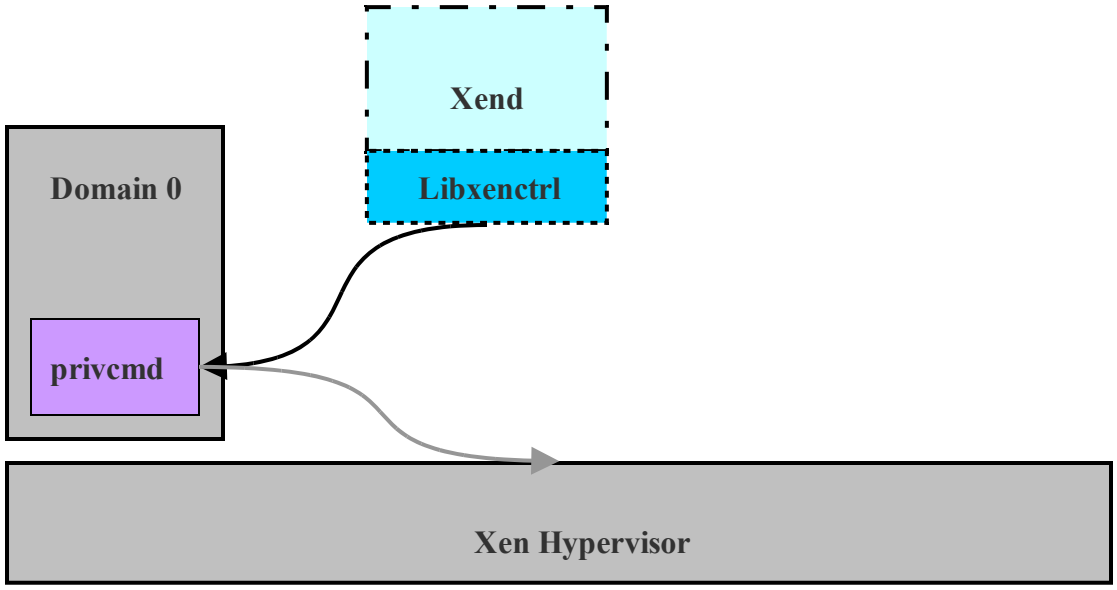
The command line tool that takes user input and passes to Xend via XML RPC.

Xenstored

The Xenstored daemon maintains a registry of information including memory and event channel links between Domain 0 and all other Domain U Guests. The Domain 0 virtual machine leverages this registry to setup device channels with other virtual machines on the system. (See Domain 0 to Domain U Communication for more details).

Libxenctrl

Libxenctrl is a C library that provides Xend the ability to talk with the Xen hypervisor via Domain 0. A special driver within Domain 0, privcmd delivers the request to the hypervisor.



Qemu-dm

Every HVM Guest running on a Xen environment requires its own Qemu daemon. This tool handles all networking and disk requests from the Domain U HVM Guest to allow for a fully virtualized machine in the Xen environment. Qemu must exist outside the Xen hypervisor due to its need for access to networking and I/O and is therefore found in Domain 0.

A new tool, Stub-dm, is in development for future versions of Xen that will remove the need for a Qemu running for every Domain U HVM Guest and will instead provide a set of services available to every Domain U HVM Guest. This feature is not available in Xen 3.2 but is currently part of the xen-unstable tree and will be released as part of Xen 3.3.

Xen Virtual Firmware

The Xen Virtual Firmware is a virtual BIOS that is inserted into every Domain U HVM Guest to ensure that the operating system receives all the standard start-up instructions it expects during normal boot-up providing a standard PC-compatible software environment.

Xen Operation

This section demonstrates how a paravirtualized Domain U is able to communicate with external networks or storage via the Xen hypervisor and Domain 0.

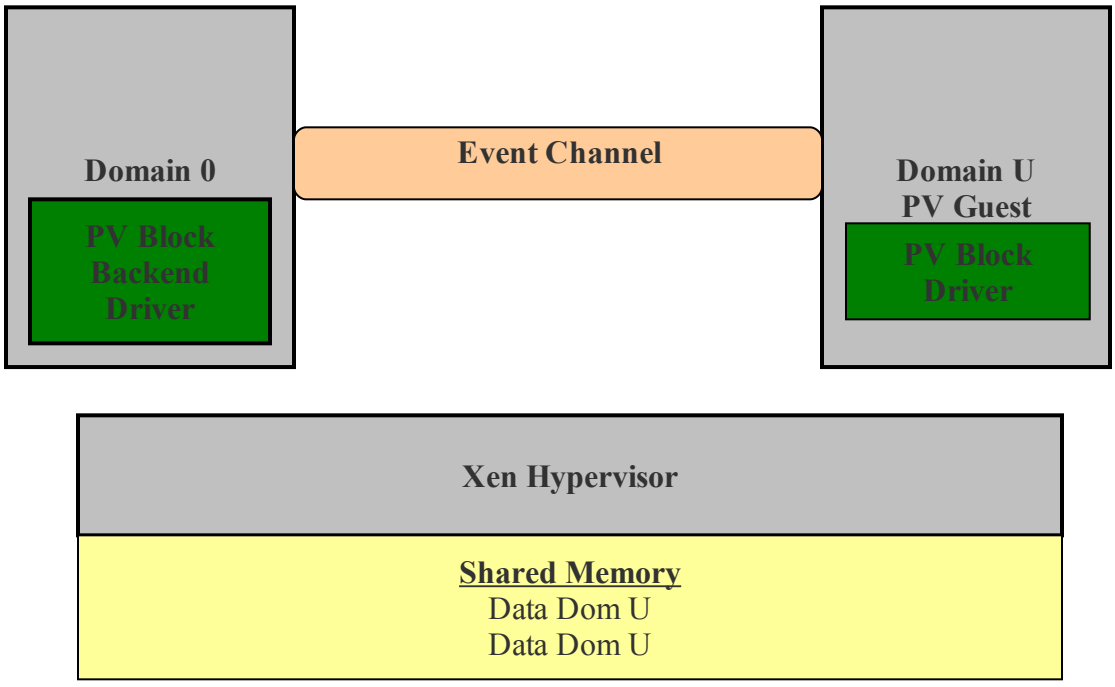
Domain 0 to Domain U Communication

As stated earlier, the Xen hypervisor is not written to support network or disk requests thus a Domain U PV Guest must communicate via the Xen hypervisor with the Domain 0 to accomplish a network or disk request. The example presented below shows a Domain U PV Guest writing data to the local hard disk.

The Domain U PV Guest PV block driver receives a request to write to the local disk and writes the data via the Xen hypervisor to the appropriate local memory which is shared with Domain 0. An event channel exists between Domain 0 and the Domain U PV Guest that allows them to communicate via asynchronous inter-domain interrupts in the Xen hypervisor. Domain 0 will receive an interrupt from the Xen hypervisor causing the PV Block Backend Driver to access the local system memory reading the appropriate blocks from the Domain U PV Guest shared memory. The data from shared memory is then written to the local hard disk at a specific location.

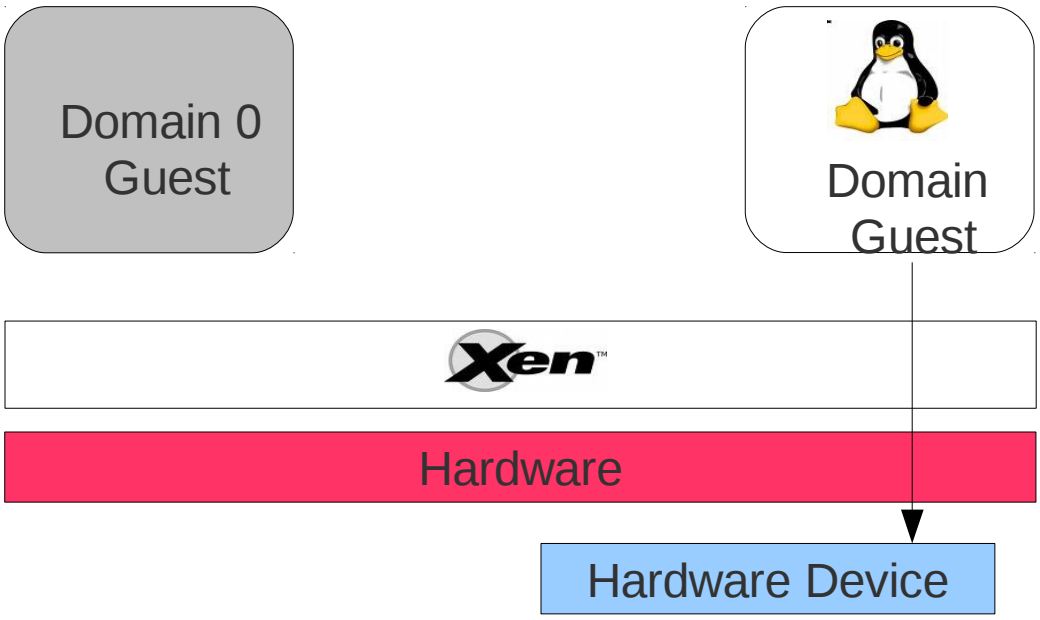
The event channel is shown below as a direct link between Domain 0 and Domain U PV Guest which is a simplified view of the way the system works. In fact, the event channel runs through the Xen hypervisor with specific interrupts registered in Xenstored allowing both the Domain 0 and Domain U PV Guest to quickly share information across local memory.

The figure below shows the previously described situation:



Xen PCI Passthru

A new feature in Xen designed to improve overall performance and reduce the load on the Dom 0 Guest is PCI Passthru which allows the Domain U Guest to have direct access to local hardware without using the Domain 0 for hardware access. The diagram below shows how this feature works:



The Domain U Guest is given rights to talk directly to a specific hardware device instead of the previous method of using Fronted and Backend drivers.

Glossary

C: http://www.cprogramming.com/; a computer programming language

Daemons: http://en.wikipedia.org/wiki/Daemon_(computer_software); a program running in the background rather than under direct control of a user

Driver: http://en.wikipedia.org/wiki/Device_driver; program allowing software to interact with hardware

Full Virtualization: http://en.wikipedia.org/wiki/Full_virtualization; a virtual machine not aware of its virtualization

Interrupt: http://en.wikipedia.org/wiki/Interrupt; signal from hardware to software requesting a specific action in software

Kernel: http://en.wikipedia.org/wiki/Linux_kernel; the central component of a computer operating system

Paravirtualized: http://en.wikipedia.org/wiki/Paravirtualization; virtual machine running on a hypervisor that is aware of it being virtualized]

Python: http://www.python.org/; a dynamic object oriented programming language

ROM BIOS: http://en.wikipedia.org/wiki/BIOS; software instructions run on a machine when turned on

XML PRC: http://www.xmlrpc.com/; method for an application to leverage another application using HTTP for the remote procedure call and XML as the encoding