Actual Virtualisation, Finally

Virtualisation

Otago Polytechnic
Dunedin, New Zealand

Weird idea

For a change, we will talk about running proper virtual machines on top of some hardware.

Basic idea

The main idea is that we run a piece of software, generally called a *hypervisor* on top of our hardware that provides an environment in which guest operating systems can run.
However, there are several variations on this theme.

Emulation

With *emulation* we create a virtual environment that models a hardware architecture that is different from the underlying host. We do this to develop software for hardware that does not yet exist or to run old software build for an architecture that is no longer available.

Full virtualisation

Full virtualisation presents guests with a virtual environment that matches the target architecture of the underlying hosts. It models the hardware layer completely so that the guest operating system doesn't require any modification to run in the virtual environment.

Paravirtualisation

Paravirtualisation also presents guests with a virtual environment that matches the target architecture of the underlying hosts, but unlike full virtualisation, paravirtualisation requires that guest operating systems are modified to adapt to the virtual environment. This typically offers some performance advantages over full virtualisation.

Hosted vs. bare metal

Another dimension to consider is whether the hypervisor is *hosted* or *bare metal*.

- A bare metal hypervisor runs directly on the host hardware, without an operating system.
- A hosted hypervisor runs on top of a full operating system.

These are also referred to as type1 and type 2 virtualisation, respectively, which is stupid.

Xen

We are going to work with the Xen hypervisor over the next few weeks. Xen is an open source hypervisor that originated at the University of Cambridge in 2001. It provides a paravirtualised (sometimes full) environment for x86 architectures.

Protection rings

The x86 architecture supports four *protection rings*. Usually the operating system runs at ring 0 and user applications run at ring 3.

With Xen, the hypervisor runs at ring 0, guest operating systems at ring 1, and applications at ring 3. Since full access to the hardware is only available at ring 0, this lets the hypervisor control resources and isolate guest domains from each other.

Domain 0

Xen guests run in *domains*. In order to provide a management interface there is always one special domain, *Domain0* or just *Dom0*. In Dom0 we (typically) run a Linux operating system that itself runs services that we use to work with the hypervisor and its guest domains. Dom0 is started as soon as a Xen host is booted.

Regular guests run in unprivileged domains, usually called *DomU*.

Device drivers

Dom0 has direct access to the physical hardware and has native drivers for it. It then presents simplified abstract drivers to DomU guests. Alternatively, Dom0 can delegate device access to special *driver domains* to manage the actual hardware drivers.

XEND

Dom0 runs a special service `xend`, the Xen management daemon. `xend` works with the hypervisor to start, stop, and manage guest domains. We typically interact with `xend` using `xm`, the Xen management interface.

On occasion, Xen hosts communicate with `xend` daemons running on other hosts to manage thing like live migrations of guest domains.

XenStore

Dom0 also runs *XenStore*, a shared database of configuration information to which guest domains read and write. As sysadmins we can also query XenStore to get information about the status of domains. The XenStore database is organised in a way the models a file system tree that we can traverse to find information about domains.

The plan from here

Starting next time you will pair up with a classmate and be assigned a physical machine. Over the next few weeks you will install and configure Xen and see how to use ti to run virtual machines.