Creating Fault-Tolerant Xen Virtualization at the Network Adapter Layer

In development and production environments, fault tolerance at the physical network layer can be critical. Using the Linux® OS bonding driver with standardized networking configurations enables highly available, fault-tolerant physical layers for Xen guest domains.

BY SCOTT COLLIER AND BRIAN GAUTREAU

Related Categories:

Data networking

Dell ninth-generation servers

Dell PowerEdge servers

Fault tolerance

Linux

Virtualization

Xen virtualization

Visit www.dell.com/powersolutions for the complete category index.

Xen virtualization is an open source technology designed to securely host multiple virtual machines—also called domains—on a single physical system, helping increase server utilization through a free, robust alternative to expensive proprietary virtualization technologies. In addition to helping increase server utilization, however, hosting multiple domains on stand-alone physical systems can increase the need for reliable fault-tolerant hardware to help maintain maximum domain uptime. For example, an individual network adapter can fail for a number of reasons (upstream switch failure, faulty network interface card, bad cabling, and so on), and such a failure on a physical server hosting multiple domains could create significant problems for the IT department responsible for maintaining resource availability.

This article discusses how administrators can implement Linux bonding with Xen to help increase fault tolerance and resource availability, and how they can test the resulting configuration.

Xen network infrastructure and bonding

Xen has two primary networking architecture options: routing and bridging. Routing simply puts packets onto a physical wire after they have been routed through the server; this option includes little or no isolation for guest domains from the host domain or other guest domains. Bridging, in contrast, allows the isolation of network traffic from domain traffic, which can allow testing without actually putting packets onto a wire and potentially interfering with other host communication.

In Xen, a bridge is equivalent to a virtual switch presented to the domains. Each physical server can have several bridges, with multiple domains attached to each bridge; each bridge can, in turn, be connected to a physical network interface card, although this is not necessary. By default, Xen presents a single bridge, xenbr*x* (where *x* corresponds to the number of the primary interface, typically 0). The host domain (dom0) and new domains (domUs) are attached to the default xenbr*x* bridge unless otherwise specified.

Bridges typically consist of a connection to a physical interface (an uplink) and a virtual interface (a patch to a server). The physical interface can be any network adapter in the server; the virtual interface corresponds to the domain Ethernet interface, as shown in Figure 1.

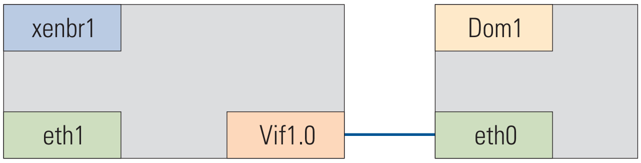

Figure 1. Xen bridge showing a connection to a virtual domain interface

Linux provides a native bonding module for achieving adapter fault tolerance. The bonding driver is documented in the Linux kernel at /usr/share/doc/kernel-doc-*kernelversion*/Documentation/networking/bonding.txt. Although bonding has several possible modes, including load balancing and link aggregation, the active/backup policy helps provide a simple, fail-safe method for fault tolerance. Using this policy enables one adapter to be used as a primary interface while the second is kept in a standby state so it can rapidly take over if the primary adapter fails, helping ensure that the server configuration is not subject to a single point of failure at the network adapter layer. Using bonding in Xen is equivalent to bonding in the Fedora or Red Hat® Enterprise Linux operating systems.

Bonded server adapter test configuration

To demonstrate the configuration and testing of bonded server adapters in a virtualized environment, in September 2006 Dell support engineers configured a Dell™ PowerEdge™ 2950 server with two dual-core Intel® Xeon® processors at 3.00 GHz with Intel Virtualization Technology, dual on-board Broadcom NetXtreme II BCM5708 network adapters along with an additional BCM5708 adapter on an expansion card, and an integrated PowerEdge Expandable RAID Controller (PERC) 5/Di. For the Xen environment, the engineers used Fedora Core 5 with Xen 3.0.2. Figure 2 illustrates the virtualized test environment.

Configuration of bonding, bridges, and guest domains

To set up bonded interfaces, the test team first loaded the bonding driver and configured the bond. The drivers are typically loaded during startup along with the passing of options to the driver by using the /etc/modprobe.conf file. Although additional driver options and modes are available for the driver and included in the kernel documentation, the test team set up an active/backup configuration by limiting the bond to the following:

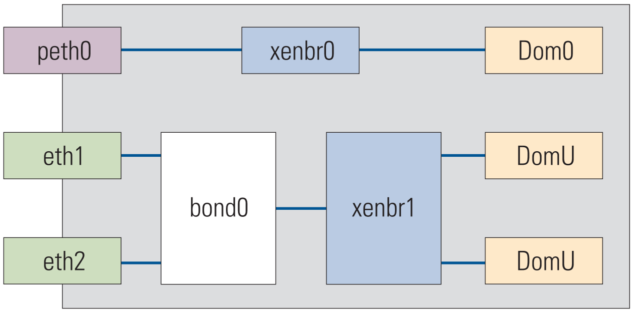

Figure 2. Xen virtualized test environment

```
alias bond0 bonding
options bond0 miimon=100 type=active-backup
```

They then created the interface startup script, /etc/sysconfig/network-scripts/ifcfg-bond0, which contains the basic information to start the interface:

```
DEVICE=bond0
BOOTPROTO=none
ONBOOT=no
```

Next, they edited the /etc/sysconfig/network-scripts/ifcfg-eth1 file to add it to the bond:

```
DEVICE=eth1
BOOTPROTO=none
ONBOOT=no
MASTER=bond0
SLAVE=yes
```

They also did so for /etc/sysconfig/network-scripts/ifcfg-eth2:

```
DEVICE=eth2
BOOTPROTO=none
ONBOOT=no
MASTER=bond0
SLAVE=yes
```

The preceding configurations create network interfaces that, upon starting the bond, enslave the physical interfaces to the bond. To help ensure that the interfaces started properly, the team rebooted and verified that they were available:

```
# ifconfig bond0
# ifconfig eth1
# ifconfig eth2
```

The next step was to create the additional bridge for the guest domains using the `brctl` command:

```
# brctl addbr xenbr1
```

The bridge and the bond should be configured to disable Address Resolution Protocol (ARP) and multicast. In the test environment, the bond interface used a Media Access Control (MAC) address that would not interfere with public MAC addresses:

```
# ip link set xenbr1 arp off
# ip link set xenbr1 multicast off
# ip link set xenbr1 up
# ip link set bond0 address fe:ff:ff:ff:ff:ff
# ip link set bond0 arp off
# ip link set bond0 multicast off
```

To add the bond to the bridge, the team again used the `brctl` command, and then brought up the bond interface and enslaved the physical interfaces in one step using the `ifup` command:

```
# brctl addif xenbr1 bond0
# ifup bond0
```

The last configuration step was to configure a guest domain to use the new bridge. To do so, the following must be set in the /etc/xen/*guestdomain* configuration file (although additional parameters can be set for an interface, `bridge=xenbr1` is the critical option):

```
vif = [ 'type=ioemu, bridge=xenbr1' ]
```

At any time before, during, or after starting the domains, administrators can use the `brctl` command to view the bridge and the interfaces attached to it, as shown in Figure 3. All of the preceding commands can be added to a shell script and automatically executed when booting to bring up the interface.

```
# brctl show
bridge name   bridge id           STP enabled   interfaces
xenbr0        8000.feffffffffff   no            peth0

vif0.0
xenbr1        8000.feffffffffff   no            bond0
# xm create RHEL4_HVM_Guest
Using config file “RHEL4_HVM_Guest”.
Started domain RHEL4_HVM_Guest
# brctl show
bridge name   bridge id           STP enabled   interfaces
xenbr0        8000.feffffffffff   no            peth0

vif0.0
xenbr1        8000.feffffffffff   no            vif1.0
                                                bond0
```

Figure 3. Bridge and interface information returned using the brctl command

```
# tail  /var/log/messages
xendomain0 kernel: bonding: bond0: releasing active
   interface eth2
xendomain0 kernel: device eth2 left promiscuous mode
xendomain0 kernel: bonding: bond0: making interface eth1
   the new active one.
xendomain0 kernel: device eth1 entered promiscuous mode
```

Figure 4. Domain0 /var/log/messages showing a backup interface taking over following primary interface failure

Fault-tolerant configuration tests

Administrators can test the high availability of the interfaces by pulling the network cables and monitoring networking services. The Dell engineers used this method by setting up a Secure Shell (SSH) session to a guest domain and then simulating the failure of an upstream switch by pulling the active network cable, after which the network failed over to the backup interface. The transition is transparent to the SSH session.

If accessing the network cabling is difficult, administrators can also test the interfaces by using the following command, which simulates a similar condition of active interface failure:

```
# ifdown eth2
```

Administrators can use the domain0 /var/log/messages to monitor the bond status, as shown in Figure 4. When an interface fails, it is immediately reported as down in /var/log/messages. If the failed interface is the active one, the backup interface takes over, and the interface loss is transparent to the guest domains.

Fault-tolerant Xen virtualization

Using Linux bonding with Xen virtualization can provide the flexibility that enterprise IT infrastructures need to help ensure the availability of guest domains and subsystems. The methods described in this article enable enterprises to implement high availability at the network adapter layer in critical virtualized development and production systems.

Scott Collier is a network engineer and senior system consultant with the Enterprise Expert Center's Network Operating System Group at Dell.

Brian Gautreau is a network engineer and system consultant with the Enterprise Expert Center's Network Operating System Group at Dell.