



LDOMS I/O BEST PRACTICES NETWORK AVAILABILITY WITH LOGICAL DOMAINS

Peter A. Wilson, Systems Technical Marketing

Sun BluePrints™ Online

Part No 820-5681-11
Revision 1.1, 9/25/08

Table of Contents

Network Availability With Logical Domains	1
Introducing Logical Domains	1
Network Availability Overview	3
Availability Through Redundancy	3
IP Multipathing In the Solaris OS	5
IP Multipathing In Logical Domains	6
Configuring Network Availability	10
Prerequisites for IPMP with Logical Domains	10
Configuring IPMP with Two I/O Domains	13
Configure a Second I/O Domain	13
Configure IPMP in the Guest Domain	17
Test IPMP in the Guest Domain.....	18
Configuring IPMP in a Single I/O Domain	20
Configure the I/O Domain.....	20
Configure IPMP in the Guest Domain and Test	22
Summary	23
About The Author	23
Acknowledgments.....	24
References	24
Ordering Sun Documents	24
Accessing Sun Documentation Online	24

Chapter 1

Network Availability With Logical Domains

Virtually every server in a modern datacenter depends on network connectivity in order to access resources and provide services to clients. The best practice for enhancing network connectivity is to configure multiple physical paths to the network so that if a network adapter, cable, or upstream switch fails due to equipment problems or human error, the second path continues to carry traffic. Redundant physical connections must be complemented with software that can fail over between links in the event of an error. The Solaris™ Operating System supports IP multipathing (IPMP). This allows a system to be configured to automatically fail over a group of IP addresses hosted on multiple interfaces while load sharing outbound traffic across all available links.

The best practices for maximizing a physical server's connectivity to a physical network are well understood — but how do these rules translate into the virtual world? What does it mean to have multiple network connections to a server when those connections and the server itself are virtual? This Sun BluePrints™ Series article is the second in a series that is designed to help map I/O best practices from the physical world into the virtual world supported by Logical Domains.

Introducing Logical Domains

Logical Domains is a virtualization technology supported on Sun servers with CoolThreads™ technology — those powered by UltraSPARC® T1, T2, and T2 Plus processors. Logical Domains, or LDoms, allow server resources to be partitioned and allocated to separate virtual machines. Resources can be allocated down to the CPU thread, cryptographic processor, memory, and PCI bus.

The Logical Domains architecture consists of four classes of virtual machines that run on a thin hypervisor layer (Figure 1): a single *control domain* manages the virtualized environment; one or more *I/O domains* own real I/O devices that *service domains* use to provide virtual I/O services to *guest domains*. Each domain runs in its own dedicated partition of the server hardware. A single domain can take on multiple roles: for example I/O and service domains are usually combined into a single domain that handles physical and virtual I/O. In many configurations, the control domain is combined with an I/O and service domain. In this paper, we refer to a combined I/O and service domain as simply an I/O domain.

Logical Domains allow individual PCI root nexus nodes (referred to as PCI buses in this article) to be allocated to I/O domains so that each one 'owns' a PCI bus and any devices connected to it. On servers with two PCI buses, two I/O domains can be configured to provide multiple paths from guest domains to I/O resources. If an I/O domain, its PCI bus, or its peripherals fail, or the domain needs to be rebooted, a guest

domain can continue to access its I/O resources through the second I/O domain given an appropriate configuration in the guest domain.

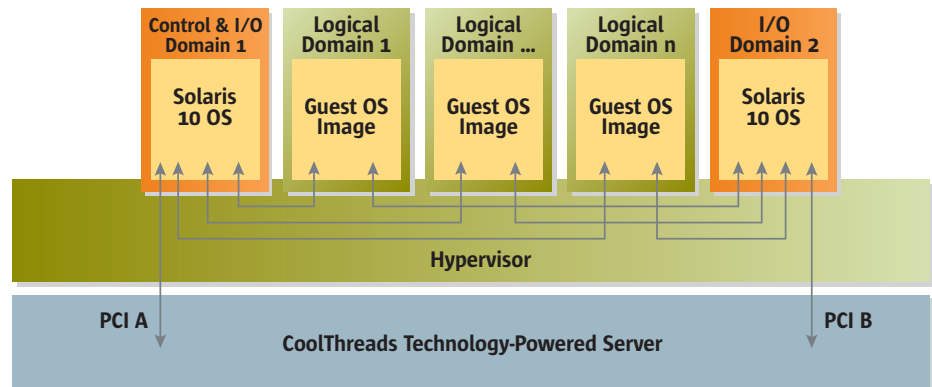


Figure 1. Logical Domains supports multiple guest domains each with their own secure partition of server hardware. I/O is handled by I/O domains that ‘own’ one or more PCI buses.

The combination of Logical Domains and IPMP in the Solaris OS can be used to increase network availability between guest operating systems and the physical network. This combination of technologies can be used on servers with CoolThreads technology having either one or two PCI buses, or equivalent I/O devices such as the 10 Gigabit Ethernet NIU in the case of UltraSPARC T2 processor-based systems. Now, if a single I/O path fails, the guest OS still retains network connectivity.

About This Article

This Sun BluePrints article discusses the approaches and trade-offs for increasing network availability using multiple network paths between the physical network and logical domains.

- Chapter 2, “Network Availability Overview” on page 3, provides an overview of using multiple paths to the network to increase availability, and which solutions are available on which Sun servers.
- Chapter 3, “Configuring Network Availability” on page 10, gives common background and steps for configuring either solution.
- Chapter 4, “Configuring IPMP with Two I/O Domains” on page 13, provides step-by-step instructions for configuring IPMP via two I/O domains.
- Chapter 5, “Configuring IPMP in a Single I/O Domain” on page 20, provides step-by-step instructions for configuring IPMP using a single I/O domain.

Chapter 2

Network Availability Overview

The way to improve network availability — the ability of a real or virtual server to maintain network connectivity — is to use redundant links that eliminate single points of failure. Following a path from the network to the server or guest OS, there are many points at which human error and hardware failure can cause a network link to fail, causing a loss of connectivity (Table 1). Switch ports can fail or be configured incorrectly so that they prevent the flow of network traffic. Cables can be plugged into the wrong ports, their conductors or connectors can break, and they can fail if their pattern of twists is disrupted by incorrect connector placement or though bending at a tight radius. On the server, network interfaces can fail, as can their connectors.

Within the server, failures of PCI bus controllers and interfaces to processors are likely to be associated with catastrophic events that cause the server itself to fail. When virtualization is used, however, software is interposed between the physical NIC and the guest operating systems, adding another potential point of failure. With Logical Domains, an I/O domain ‘owns’ the PCI bus to which a physical NIC is attached, and it acts as an intermediary between the hardware and the guest OS. Specifically, the I/O domain is configured with a *virtual switch* to which guest domains are connected by means of *Virtual NICs*. The guest domains themselves see a virtual network interface that can be managed similarly to a physical interface. Given that an I/O domain can fail, or it may need to be rebooted after OS updates are applied, it must be included in the list of possible points of failure.

Table 1. Network-related points of failure

Network Component	Reasons for Failure
Switch ports and switches	Hardware failure, misconfiguration, firmware update
Cabling	Human error, connector or conductor failures
NIC	Hardware, software, or connector failure
PCI Bus	Hardware failure, would probably seriously impair other server functions (such as access to disks)
I/O Domain	OS upgrades requiring reboot, LDoms configuration changes requiring reboot, OS failure

Availability Through Redundancy

The first step toward increasing network availability is establishing redundant paths between guest domains and the physical network. Multiple I/O domains provide redundant paths to the physical hardware so that an interruption in a single I/O domain’s availability, however momentary, does not stop the flow of network traffic. Multiple physical paths, from the PCI bus to the upstream switch ports, can protect against the more common failures including simply removing a network cable.

The ways in which Logical Domains software can map multiple physical connections to multiple virtual devices within guest domains depends on the server architecture. For the purpose of network connectivity, a server can be configured with as many I/O domains as it has PCI buses or Network Interface Units (NIUs). Multiple virtual network devices also can be provided by a single I/O domain. So whether a server has one or two PCI buses, a guest domain can always be configured with two virtual network devices that give it some level of redundancy.

The most desirable configuration is to have two physical network connections attached to a server through two different PCI buses or a PCI bus and an NIU, through two different I/O domains, and on to two different virtual network devices in a guest OS. On servers with a single PCI bus, where using an NIU is impractical (see below), two network interfaces can still be connected through a single I/O domain, however this does not remove the possibility of that I/O domain acting as a single point of failure.

Server I/O Architecture Summary

Table 2 summarizes the I/O architecture for all Sun servers with CoolThreads technology as of the date of this article. The set of recommended network and I/O domain configurations are enumerated below.

Table 2. Sun server configurations affecting I/O domain configurations with Logical Domains.

Sun Server	PCI Buses	NIUs	Number of interfaces connected to each PCI bus			
			PCI Bus	Disk Controllers	Built-In Ethernet Ports	Expansion Slots
Sun SPARC Enterprise T5240 Server	2	0	A	1	0	3
			B	0	4	3
Sun SPARC Enterprise T5140 Server	2	0	A	1	0	2
			B	0	4	1
Sun SPARC Enterprise T5220 Server	1	1	A	1	4	6
			NIU	0	0	2x 10 Gbps Ethernet
Sun SPARC Enterprise T5120 Server	1	1	A	1	4	3
			NIU	0	0	2x 10 Gbps Ethernet
Sun Fire / Sun SPARC Enterprise T2000 Server	2	0	A	1	2	1
			B	0	2	4
Sun Fire / Sun SPARC Enterprise T1000 Server	2	0	A	0	0	1
			B	1	4	0

- Two of Sun's servers, the Sun SPARC Enterprise T5120 and T5220 servers, have one PCI bus and one NIU. The NIU can be configured to replace one or two PCI Express card slots with one or two 10 Gigabit Ethernet interfaces.
 - If you wish to use the NIU and 10 Gigabit Ethernet, you can use two I/O domains, one connected to one or more Gigabit Ethernet NICs on the PCI bus, and one connected to one or more 10 Gigabit NICs on the NIU.

- If you wish to use Gigabit Ethernet using the built-in interfaces or PCI Express-based interfaces, you can use them in any combination through a single I/O domain.
- On servers with two PCI buses, use any combination of built-in network interfaces and additional PCI Express expansion-slot NICs to establish at least one path through each of two PCI buses. Configure two I/O domains, one connected to each PCI bus, to provide two virtual network devices to guest domains. If expansion slots are not available for additional NICs, use a single I/O domain connected to two or more built-in interfaces.
- The Sun SPARC® Enterprise T2000 server has two PCI buses and two built-in network interfaces connected to each bus. It is the only server in Sun's current product line that can support two I/O domains and two physically independent networks without the use of an expansion slot.
- The Sun SPARC Enterprise T1000 Server has two PCI buses but one expansion slot. If this slot is not occupied, use a PCI Express NIC for one network connection, and one built-in interface for the second one.

IP Multipathing In the Solaris OS

For more information on IP Multipathing, please consult the *System Administration Guide: IP Services*, part number 816-4554, available at docs.sun.com.

In the previous section we described how to configure physical links and I/O domains to best establish redundant paths to guest domains. This section describes how to use those links to increase availability.

IP Multipathing in the Solaris OS configures two or more physical interfaces into an IPMP group. When an IPMP group is configured, the IPMP daemon (`in.mpathd`) monitors the health of each link and fails over the group's primary IP address if the interface currently supporting it fails. Thus the interfaces configured in an IPMP group all contribute to making one or more IP address continuously available to the outside world.

IPMP can accommodate different speed links, so an IPMP group can be configured to use 10 Gigabit Ethernet on an NIU along with built-in and PCI Express-based Gigabit Ethernet interfaces.

IPMP uses a 'public' address on each interface that fails over if network connectivity is lost on one of them. Typical use is an active-active configuration where inbound traffic arrives over the interface supporting one of the public IP addresses. Outbound traffic is spread across the available interfaces to increase the IPMP group's throughput. This is an excellent strategy for services such as Web servers, where a small incoming request results in a much larger response. An alternative approach is to load balance incoming traffic across each of the IPMP group's public addresses. Although two interfaces are most often configured into an IPMP group, a larger number can be configured for higher availability and throughput.

The IPMP daemon monitors the health of the IPMP group's interfaces in two ways. It monitors the link status, and it sends periodic ICMP probes from each interface to the default router address. When configured from inside a Logical Domain, link-based failure detection does not operate because the virtual network device is always connected to a virtual switch. Instead, the IPMP daemon uses probe-based detection to recognize a failure.

IP Multipathing In Logical Domains

There are three ways in which IPMP can be configured with Logical Domains. The preferred approaches configure IPMP in the guest OS, with virtual devices supported through one or two I/O domains.

IPMP With Two I/O Domains

Two I/O domains can be used on servers with two PCI buses or with one PCI bus and an NIU. It can be used with a built-in interface and a NIC or XAUI card for the internal NIU in an expansion slot, or it can be used with two NICs in expansion slots. This solution is preferred on servers with two PCI buses as the use of two I/O domains allows one I/O domain to be rebooted without affecting network I/O.

This configuration is illustrated in Figure 2. Each of the two I/O domains owns one PCI bus and connects a virtual switch to the physical device. Each virtual switch provides a virtual network interface to guest domains.

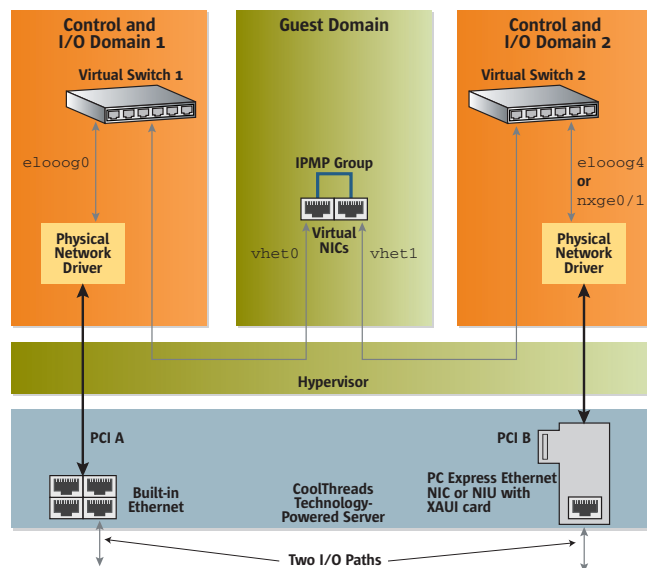


Figure 2. IPMP can be configured inside of a guest domain using virtual network devices that are supported by two I/O domains.

The IPMP group is configured from within the Solaris OS instance, and probe-based failure detection is used. The guest OS sees two virtual network devices just as in the previous example: from the perspective of the guest OS, this and the single I/O domain configuration are indistinguishable.

This solution has no single point of failure except for the server and the hypervisor, making it the preferred approach for high availability on servers with two PCI buses.

IPMP With a Single I/O Domain

A single I/O domain can be used to connect to two physical interfaces on servers with either one or two PCI buses. This is the preferred solution for servers with a single PCI bus. This solution also can be used when IPMP is implemented more for throughput than availability, as the two I/O domain solution provides more availability.

This configuration is illustrated in Figure 3. Each physical device is attached to its own virtual switch in the I/O domain. Each virtual switch is connected to a virtual network device in the guest domain. The IPMP group is configured from within the guest's Solaris OS instance. Probe-based failure detection is used to initiate failover between virtual network devices in the event of a failure.

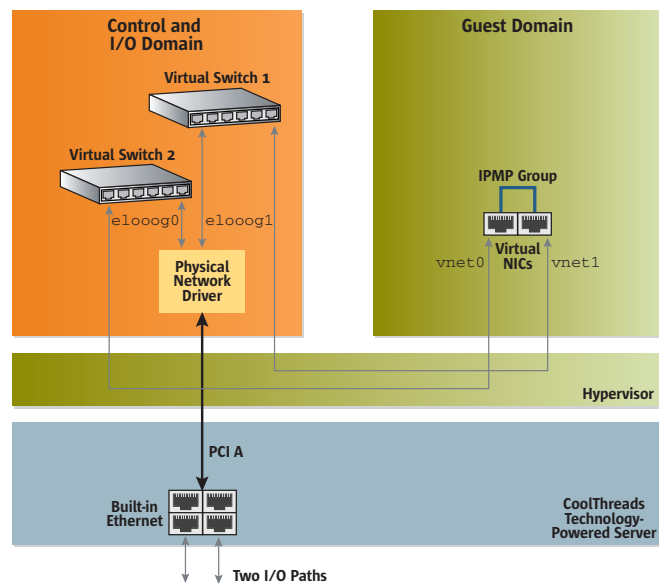


Figure 3. IPMP can be configured inside of a guest domain using virtual network devices that are supported by a single I/O domain.

This solution eliminates the low-hanging fruit of single points of failure, namely the NIC, cable, and upstream switch ports. The I/O domain is a single point of failure, and rebooting the I/O domain will cause network connectivity to be interrupted temporarily.

IPMP Inside an I/O Domain

IPMP can be implemented entirely within an I/O domain, hiding the implementation from the guest domain. This solution can simplify guest domain configuration, and it can be used on servers with one or two PCI buses. There are two drawbacks to this approach: the single I/O domain is a single point of failure for the guest; and the I/O domain must route packets between the virtual switch and the IPMP group, adding TCP/IP stack overhead to each packet transmission. For these reasons, we do not recommend this configuration.

This configuration is illustrated for a server with two PCI buses. A single virtual network device in the guest domain connects to a virtual switch in the I/O domain. The I/O domain is configured as a router, and packets are routed from the virtual switch to the IPMP group. Figure 4 illustrates an IPMP configuration using one built-in Ethernet interface, and one interface on a separate PCI Express NIC.

This solution eliminates the NIC, network cables, and upstream switch ports as single points of failure, however the I/O domain remains as a single point of failure.

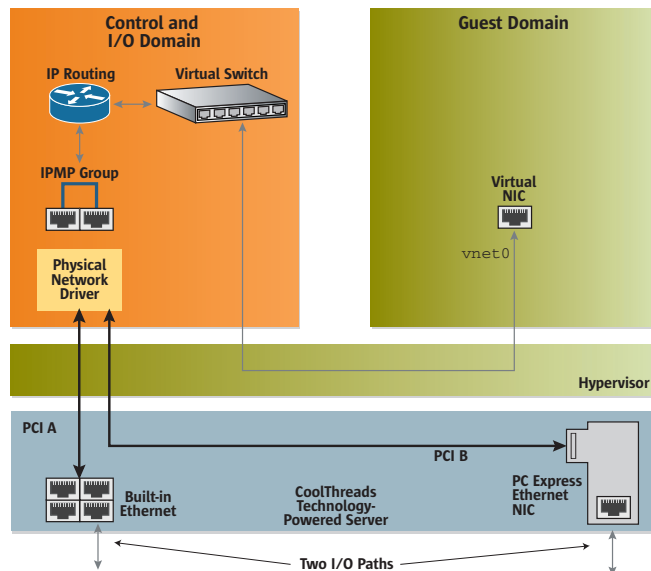


Figure 4. IPMP can be configured inside of an I/O domain, but this approach adds routing overhead to the virtual network, and the single I/O domain remains as a single point of failure.

Solution Summary

The three IPMP configurations can be supported on Sun servers with CoolThreads technology as summarized in Table 3. The table serves as a reminder that single I/O domain solutions are viable for all servers, although the single I/O domain remains as a single point of failure. The table is also a reminder that two PCI buses or a PCI bus and an NIU are required for two I/O domain solutions.

Note that more than one solution can be used on the same server. For example, one guest domain can be supported with IPMP through a single I/O domain, while another guest domain can be supported with IPMP through two I/O domains.

Table 3. Sun server configurations and how they support the three IPMP solutions.

Sun Server	PCI Buses	Applicable Solution		
		Single I/O Domain	Two I/O Domains	Inside I/O Domain
Sun SPARC Enterprise T5240 Server	2	✓	✓	✓
Sun SPARC Enterprise T5140 Server	2	✓	✓	✓
Sun SPARC Enterprise T5220 Server	1	✓	✓ (With NIU)	✓
Sun SPARC Enterprise T5120 Server	1	✓	✓ (With NIU)	✓
Sun Fire / Sun SPARC Enterprise T2000 Server	2	✓	✓	✓
Sun Fire / Sun SPARC Enterprise T1000 Server	2	✓	✓	✓

Chapter 3

Configuring Network Availability

In this chapter, we discuss the common background for configuring IPMP with two I/O domains (with steps described in Chapter 4) or with a single I/O domain (Chapter 5).

The first Sun BluePrints article in this series, *LDoms I/O Best Practices: Data Reliability with Logical Domains*, provides background material and resources for understanding Logical Domains, installing the Solaris OS, patching server firmware, installing the Logical Domains Manager (SUNWldm) package, and creating your first control and I/O domain. This, and the following chapters, assume that you are familiar with this material and that you have an I/O and guest domain to work with.

Prerequisites for IPMP with Logical Domains

There are several prerequisites for configuring either of the two solutions described in the following two chapters.

Default Route Should Be External

It is certainly possible to configure IPMP on virtual switches that are used exclusively to interconnect Logical Domains on a single server. In most cases, however, you will be configuring IPMP with network interfaces connected to a network beyond the server.

Probe-based IPMP, the only option for the two solutions we discuss, sends periodic ICMP packets to the domain's default router address. Most Solaris OS installations will define a default router that is external to the server. To verify this, `cat /etc/defaultrouter` from within a logical domain and verify that the address is external to the server.

Separate MAC Addresses Per `vnet` Device

Each virtual network device used for IPMP must have a unique MAC address. By default, the Logical Domains Manager assigns a unique MAC address to each `vnet` device when it is created, and this is the desirable behavior.

The OpenBoot™ software can override virtual network MAC address settings if the variable `local_mac_address` is set to `false`. If this is the case, each `vnet` device will assume the system's MAC address and IPMP will not work. The default setting is `local_mac_address?=true`, and you can verify this at the guest domain's `ok` prompt with the `printenv` command:

```
{0} ok printenv
```

Configure Addresses in DNS and Hosts Files

You'll need to allocate at least four IP addresses for each IPMP group that you define. Two addresses are associated with the IPMP group and can be used to access services provided by the logical domain. Two addresses are test addresses used in probe-based failover. These addresses should be allocated but should never be used to access services because these addresses are the ones that fail when a link fails.

The naming convention we use for addressing in the rest of this article is summarized in Table 4. These addresses should be defined in the `/etc/hosts` file of the guest, and by your name service (DNS or NIS).

Table 4. Four IP addresses need to be defined for each IPMP group.

Address	Interface	Purpose
guest1	vnet0	Address over which services are provided. This is the address that should be used by clients.
guest1-0	vnet0	Test address; clients should not use.
guest1-sec	vnet1	Address configured on the second interface.
guest1-1	vnet1	Test address for the second interface; clients should not use.

Following the convention outlined above, a typical `/etc/hosts` file in a guest domain would contain entries similar to the following:

```
#
# Internet host table
#
127.0.0.1    localhost
::1        localhost
192.168.0.31 guest1      loghost
192.168.0.32 guest1-sec
192.168.0.33 guest1-0
192.168.0.34 guest1-1
```

Adjust `mpathd` Timeouts

By default, the IPMP daemon uses a timeout of 10 seconds for failover, so a problem may persist for up to 10 seconds before a failover is performed. You may wish to shorten this default value, though shortening it to less than 1 second may result in too many spurious failovers due to network load or transient conditions. The timeout value is given in milliseconds in `/etc/default/mpathd`:

```
#  
# Time taken by mpathd to detect a NIC failure in ms. The minimum time  
# that can be specified is 100 ms.  
#  
FAILURE_DETECTION_TIME=10000
```

If you change this value, restart the `mpathd` daemon:

```
fraser# pkill -HUP in.mpathd
```

Chapter 4

Configuring IPMP with Two I/O Domains

When configuring IPMP in a guest domain, all the guest domain knows is that it has two virtual network devices to work with. It is completely unaware whether the two devices are provided through one or two I/O domains. So the configuration in the guest domain for IPMP with two I/O domains is exactly the same as with one I/O domain. Indeed, you can even switch between one and two I/O domains by detaching and re-attaching `vnet` devices that are connected to different virtual switches.

The only difference between the two solutions is setting up the second I/O domain to provide virtual network devices to guest domains. Conceptually, the second I/O domain provides I/O services to guest domains, and its I/O configuration is managed by the Logical Domain Manager and administrative commands in the primary domain.

Configure a Second I/O Domain

Setting up a second I/O domain involves creating a new logical domain and then giving it the server's unused PCI bus so that it can access one or more NICs configured in PCI Express expansion slots. There is one very important step to this: you must be sure that you give the second I/O domain the PCI bus that does *not* support the internal disk controller. *If you give away the PCI bus supporting the internal disk controller, and your primary I/O domain or other Logical Domains boot from it, you will lose the ability to boot your server from its internal disks.*

Create a New Logical Domain

An I/O domain is simply a Logical Domain that owns I/O resources and provides I/O services to other guest domains. You can use an existing Logical Domain as an I/O domain, or you can create a new one. This example creates a new guest domain that can then have the Solaris OS installed in it via JumpStart™ software. Another option is to make a copy of another guest domain's boot disk for use by the new I/O domain.

In this example, the I/O domain is named `secondary`, and it is configured with a virtual network interface (supported by the primary I/O domain) that can be used for the network install process.

```
fraser# mkfile 10G /domains/secondary.img
fraser# ldm create secondary
fraser# ldm set-vcpu 4 secondary
fraser# ldm set-mem 4g secondary
fraser# ldm add-mau 1 secondary
fraser# ldm add-vnet vnet0 primary-vsw0 secondary
fraser# ldm add-vdsdev /deomains/secondary.img vol8@primary-vds0
fraser# ldm add-vdisk vdisk0 vol8@primary-vds0 secondary
fraser# ldm bind secondary
```

Make sure that you save the configuration to the service processor so that it will survive a power cycle of the server. This example stores the configuration under the name `fourguests`.

```
fraser# ldm remove-sponfig fourguests
fraser# ldm add-sponfig fourguests
```

At this point the guest domain's operating system can be installed over the network as described in the first article in this series: *LDoms I/O Best Practices: Data Reliability with Logical Domains*.

Only the Solaris 10 OS needs to be installed in the guest domain; do not attempt to install the Logical Domain Manager in the second I/O domain.

Assign I/O Resources

Now you endow the new guest domain with the I/O resources that makes it an I/O domain. To achieve this, you must remove one of the PCI buses from the primary I/O domain and assign it to the second one.

Note – To avoid mistakes that can prevent your server from booting, always follow this procedure even if you think you know which bus supports which devices. For some servers, the internal disk controller is assigned to different PCI buses depending on the server production date.

Determine PCI Bus Configuration

List the primary I/O domain's bindings to verify that the server has two PCI buses, and what their names are:

```

fraser# ldm list-bindings primary
...
IO
  DEVICE          PSEUDONYM      OPTIONS
  pci@780         bus_a
  pci@7c0         bus_b

```

If your server is a Sun SPARC Enterprise T5120 or T5220 server, your output will make it obvious which is the PCI bus and which is the NIU. Since you will be giving away the NIU to the secondary I/O domain, you don't have to worry about giving away the server's ability to boot from its internal disk. For the instructions that follow, use the name of the NIU (for example, `niu@80`) as the PCI bus assigned to the secondary I/O domain.

Locate the Disk Controller

Now find out on which bus the disk controller providing the primary I/O domain's boot disk is configured. First find the disk device that corresponds to your boot disk, in this case this is `/dev/dsk/c1t0d0s0`:

```

fraser# df /
/                (/dev/dsk/c1t0d0s0 ):47729512 blocks  6297665 files

```

Then find the PCI bus to which it is attached:

```

fraser# ls -l /dev/dsk/c1t0d0s0
lrwxrwxrwx  1 root  root          65 Mar 11 15:33 /dev/dsk/c1t0d0s0
-> ../../devices/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@0,0:a

```

The string `pci@7c0` indicates that the disk controller is on `bus_b`. This finding serves as an example of how important it is to go through this procedure. The server on which these commands were tested is an early production Sun SPARC Enterprise T2000 server with its disk controller occupying a PCI-X slot on `bus_b`. Later models of the same server have disk controllers configured onto `bus_a`. This exercise tells us that you can give `bus_a` to the new I/O domain, along with any network devices attached to it.

Locate Network Devices

Now locate on which bus your network devices are configured. In the case of the Sun SPARC Enterprise T2000 sever used in this example, two built-in interfaces are on `bus_a` and two built-in interfaces are on `bus_b`. For all other servers, you will need to install a PCI Express NIC into one of the server's expansion slots to provide some NICs on the PCI

Express bus that does not have internal NICs. There is no central source of information describing which PCI Express slot is connected to which of the server's PCI buses. You can often deduce this by looking at the architecture block diagram in the server white papers, however there is no substitute for verifying the device location through this procedure.

This example shows us that the interface `e1000g0` and `e1000g1` are connected to `bus_a`, so they can be used by the second I/O domain:

```
fraser# ls -l /dev/e1000g0
lrwxrwxrwx  1 root  root          48 Mar 11 15:32 /dev/e1000g0 ->
../devices/pci@780/pci@0/pci@1/network@0:e1000g0
fraser# ls -l /dev/e1000g1
lrwxrwxrwx  1 root  root          50 Mar 11 15:32 /dev/e1000g1 ->
../devices/pci@780/pci@0/pci@1/network@0,1:e1000g1
```

Give Away a PCI Bus

At this point you know that you can't give away the primary I/O domain's `bus_b`; you must give away its `bus_a`. Use the `ldm remove-io` variant to remove `bus_a` from the primary I/O domain. Be sure that you determine the correct PCI bus for your server:

```
fraser# ldm remove-io pci@780 primary
```

Reboot the primary I/O domain, and then assign `bus_a` to the secondary I/O domain. If the domain is running, it must first be stopped and then unbound to allow you to add the new device. Then it can be rebound and restarted with its new capabilities:

```
fraser# ldm add-io pci@780 secondary
```

Boot the secondary I/O domain with the `-r` option so that it discovers its new devices.

```
{0} ok boot -r
```

Save the new LDoms configuration in the service processor; we use a new name `split`:

```
ldm add-spconfig split
```

Create a New Virtual Switch

Now that the secondary I/O domain owns a PCI bus and two network devices, you can set up a virtual switch that can attach the network device to guest domains. This command creates a virtual switch `secondary-vsw1` in the secondary I/O domain. Issue these commands from the primary I/O domain

```
fraser# ldm add-vsw net-dev=e1000g0 secondary-vsw1 secondary
```

Then reboot the secondary I/O domain to make the change take effect. You don't have to plumb the network interface for the `e1000g0` device in the I/O domain. It is merely acting as a port on a switch.

Attach a Virtual Network Device to a Guest Domain

Now add a `vnet` device to a guest domain. These commands attach `vnet1` to domain `guest1`. When the guest is rebooted, it will see its new device and you can configure IPMP from within the guest domain:

```
fraser# ldm stop guest1
LDom guest1 stopped
fraser# ldm unbind guest1
fraser# ldm add-vnet vnet1 secondary-vsw1 guest1
Initiating delayed reconfigure operation on LDom guest1. All configuration
changes for other LDoms are disabled until the LDom reboots, at which time
the new configuration for LDom guest1 will also take effect.
fraser# ldm bind guest1
fraser# ldm start guest1
LDom guest1 started
```

Configure IPMP in the Guest Domain

With the guest domain configured with two `vnet` devices, now you can boot the domain and configure an IPMP group that uses the two devices.

Configuring IPMP involves providing additional parameters to the `ifconfig` command that configures the virtual network interfaces. You can type `ifconfig` with the correct parameters, or you can modify the `/etc/hostname.device` files and reboot the guest domain. Since you'll want to test to be sure that your configuration survives a reboot anyway, modifying the hostname files and rebooting is the best approach to verify a correct, persistent configuration.

Recall the naming convention described in “Configure Addresses in DNS and Hosts Files” on page 11: the two public addresses are `guest1` and `guest1-sec`; the test addresses are `guest1-0` and `guest1-1`. The example arbitrarily names the IPMP group `blue`. Once you have these entries in `/etc/hosts` and in your name service, type the following commands:

```
guest1# echo >/etc/hostname.vnet0 "guest1 netmask + broadcast + group blue up addif guest1-0 netmask + broadcast
+ -failover deprecated up"
guest1# echo >/etc/hostname.vnet1 "guest1-sec netmask + broadcast + group blue up addif guest1-1 netmask +
broadcast + -failover deprecated up"
```

If you wish to set up the IPMP group without rebooting, you can use the contents of the `/etc/hostname` files as arguments to `ifconfig`, as in:

```
guest1# ifconfig `cat /etc/hostname.vnet0`
guest1# ifconfig `cat /etc/hostname.vnet1`
```

Test IPMP in the Guest Domain

Now reboot the guest domain and use the `ifconfig` command to verify that your IPMP group is running. This example shows the two public and the two test addresses up and running:

```
guest1# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
vnet0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 192.168.0.31 netmask ffffffff broadcast 192.168.0.255
    groupname blue
    ether 0:14:4f:fb:58:90
vnet0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 2
    inet 192.168.0.33 netmask ffffffff broadcast 192.168.0.255
vnet1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
    inet 192.168.0.32 netmask ffffffff broadcast 192.168.0.255 groupname blue
    ether 0:14:4f:f9:e:3b
vnet1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 3
    inet 192.168.0.34 netmask ffffffff broadcast 192.168.0.255
```

Use the command `ping -s guest1` from another system to monitor the connectivity that the IPMP group provides. Then simulate some failures by pulling network cables or disabling switch ports.

If you pull the network cable attached to the device through which `vnet0` is supported, you'll observe the console message:

```
Jun 26 16:50:26 guest1 in.mpathd[158]: NIC failure detected on vnet0 of
group blue
Jun 26 16:50:26 guest1 in.mpathd[158]: Successfully failed over from NIC
vnet0 to NIC vnet1
```

If you observe the interface state from the guest, you'll note that the `vnet0` and `vnet0:1` devices are flagged as `FAILED`, and the two public IP addresses are now associated with `vnet1:2`:

```
guest1# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
    inet 127.0.0.1 netmask ffffffffvnet0: flags=19000842<BROADCAST,RUNNING,MULTICAST,IPv4,NOFAILOVER,FAILED>
mtu 0 index 2
    inet 0.0.0.0 netmask 0
    groupname blue
    ether 0:14:4f:fb:58:90
vnet0:1: flags=19040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER,FAILED> mtu 1500 index 2
    inet 192.168.0.33 netmask ffffffff broadcast 192.168.0.255
vnet1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
    inet 192.168.0.32 netmask ffffffff broadcast 192.168.0.255
    groupname blue
    ether 0:14:4f:f9:e:3b
vnet1:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500 index 3
    inet 192.168.0.34 netmask ffffffff broadcast 192.168.0.255
vnet1:2: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
    inet 192.168.0.31 netmask ffffffff broadcast 192.168.0.255
```

Restore the external connection and observe the console message indicating that IPMP has failed back the repaired interface:

```
Jun 26 16:59:03 guest1 in.mpathd[158]: NIC repair detected on vnet0 of
group blue
Jun 26 16:59:03 guest1 in.mpathd[158]: Successfully failed back to NIC
vnet0
```

Likewise, you can simulate failures and repair of the physical link connected to the `vnet1` interface.

Chapter 5

Configuring IPMP in a Single I/O Domain

Once you've taken care of the prerequisites described in Chapter 3, setting up IPMP through a single I/O domain involves creating a second virtual switch, assigning it to a network device, and setting up IPMP in the guest domain.

Configure the I/O Domain

The first step is to configure the guest domain with two network devices. This step assumes that you've already configured one of the primary I/O domain's network devices, created a virtual switch, and attached a virtual network device to the guest domain. For this example, you have already attached the device `e1000g2` to the virtual switch `primary-vsw0`, and you have attached a `vnet` device on the switch connected to the guest domain named `guest1`.

Configure the Second Network Device

If you haven't already, configure a second network interface in the I/O domain. Add a hostname entry to `/etc/hosts` and add the new hostname to `/etc/hostname.interface` file. For the network device `e1000g3`. This example uses the hostname `switch2`.

This step isn't absolutely necessary because the virtual switch you create in the next step can access the device directly. Configuring the second network device allows you to test to be sure it is working before you use it to provide services to the guest domain.

```
fraser# echo switch2 >/etc/hostname.e1000g3
```

Reboot the I/O domain to make the changes take effect. Although you can manually execute the `ifconfig` command to configure the interface, rebooting the domain tests the configuration change when you make it. If there are any errors, you catch them now rather than at some time in the future when you reboot the domain.

Create a New Virtual Switch

Create a new virtual switch named `primary-vsw1` in the I/O domain. This case assumes that you're using network device `e1000g3` in the I/O domain named `primary`:

```
fraser# ldm add-vsw net-dev=e1000g3 primary-vsw1 primary
```

You can verify that your I/O domain is configured with two virtual switches:

```
fraser# ldm list-services primary
...
VSW
  NAME                MAC                NET-DEV  DEVICE  MODE
  primary-vsw0        00:14:4f:fb:ae:95  e1000g2  switch@0  prog,promisc
  primary-vsw1        00:14:4f:f8:6a:40  e1000g3  switch@1  prog,promisc
```

Save the configuration in the service processor so it survives a power cycle, and reboot the primary I/O domain so that the deferred changes take effect. This example replaces the existing active configuration `fourguests` with the new one:

```
fraser# ldm remove-spconfig fourguests
fraser# ldm add-spconfig fourguests
fraser# reboot
```

Attach a Virtual Network Device to a Guest Domain

After the reboot, connect the virtual switch to the logical domain; this example uses the domain named `guest1`.

```
fraser# ldm stop guest1
LDom guest1 stopped
fraser# ldm unbind guest1
fraser# ldm add-vnet vnet1 primary-vsw1 guest1
Initiating delayed reconfigure operation on LDom guest1. All configuration
changes for other LDoms are disabled until the LDom reboots, at which time
the new configuration for LDom guest1 will also take effect.
fraser# ldm bind guest1
fraser# ldm start guest1
LDom guest1 started
```

Connect to the guest domain's console port and note at the `ok` prompt that two `vnet` devices are now available to the guest:

```
{0} ok show-nets
a) /virtual-devices@100/channel-devices@200/network@1
b) /virtual-devices@100/channel-devices@200/network@0
q) NO SELECTION
Enter Selection, q to quit: q
```

Configure IPMP in the Guest Domain and Test

Now you can configure IPMP in the guest domain and test following the instructions in “Configure IPMP in the Guest Domain” on page 17

Chapter 6

Summary

The virtual world enabled by Logical Domains on Sun Servers with CoolThreads technology opens up a broad range of possibilities for organizations needing to implement network availability mechanisms to match those on physical servers. Network availability can be increased by establishing separate physical paths to network resources, making them available to guest domains through one or two I/O domains:

- On servers with two PCI buses, or a PCI bus and an NIU, a combination of internal network interfaces and NICs used in expansion slots allows two redundant paths to be established. In these configurations, even the failure or rebooting of an I/O domain does not interrupt network availability.
- On servers with one PCI bus, where 10 Gigabit Ethernet through the NIU is not feasible, two physical network paths can be established and routed through a single I/O domain. Although this configuration is susceptible to a failure or rebooting of an I/O domain, it provides protection from the most common network failures. It protects against human errors including switch misconfiguration, cables being pulled inadvertently, and cables plugged into the wrong switch port or server. It also protects against failures in NICs, cables, and switch ports themselves.

Once at least two paths are established to the network, the virtual network devices that guest domains see can be configured into IPMP groups that provide continuous network access despite the failure of one of the two links. Because IPMP is set up in a virtual environment, the guest domain's IPMP configuration is independent of whether its network access is provided through one or two I/O domains. The beauty of this simple solution is that both configurations can be used on the same physical server, and guest domains even can be switched back and forth between using one or two I/O domains without them being aware of the underlying change.

About The Author

Peter A. Wilson has more than 16 years of industry experience, 12 of which have been with Sun, serving in a wide variety of hardware, software, systems and product marketing roles. Peter moved from the United Kingdom to the US in 2000 to lead the customer tests of Sun's Netra™ and fault-tolerant servers. Peter is currently a technical marketing engineer responsible for all aspects of Sun's servers with CoolThreads technology. Peter holds a M.Eng (Master of Engineering) degree in Microelectronics and Software Engineering from the University of Newcastle-upon-Tyne, U.K.

Acknowledgments

The author would like to thank Steve Gaede, an independent technical writer and engineer, for working through the issues and configurations described in this paper and developing this article based on them. Thanks also to Alexandre Chartre and Narayan Venkat for their help in understanding some of the nuances of Logical Domains.

References

References to relevant Sun publications are provided in sidebars throughout the article.

Ordering Sun Documents

The SunDocsSM program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is

`http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at:

`http://www.sun.com/blueprints/online.html`

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN (9786) **Web** sun.com



© 2008 Sun Microsystems, Inc. All rights reserved. Sun, Sun Microsystems, the Sun logo, BluePrints, CoolThreads, JumpStart, Netra, OpenBoot, Solaris, and SunDocs are trademarks or registered trademarks of Sun Microsystems, Inc. or its subsidiaries in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc. Information subject to change without notice. Printed in USA 9/2008