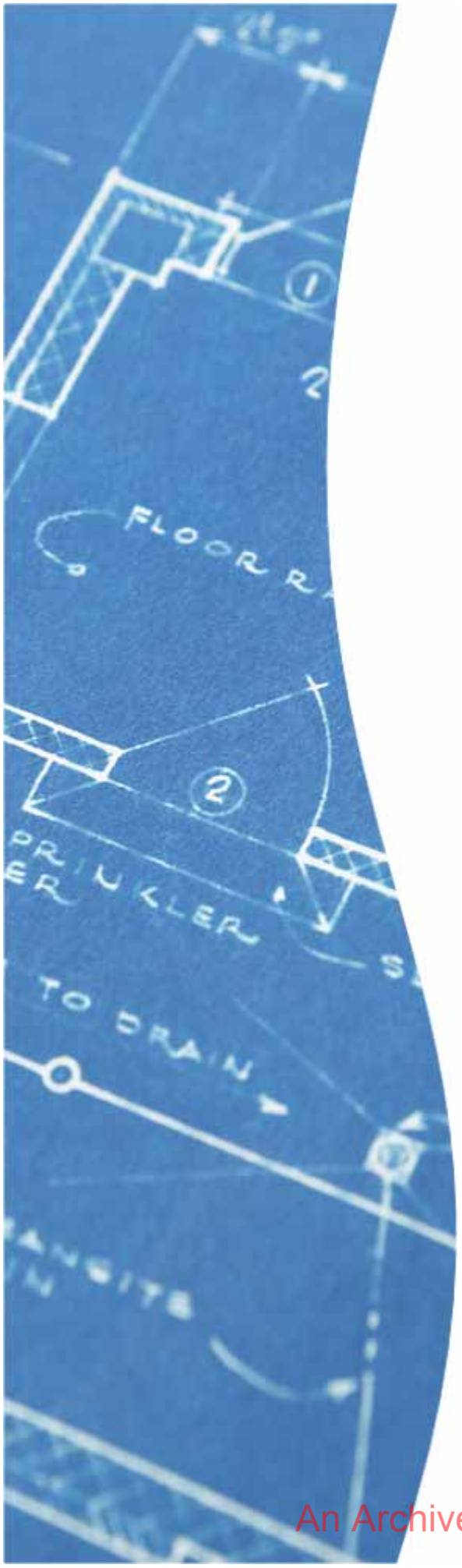
An Archived Oracle Technical Paper
April 2008

USING SOLARIS CLUSTER AND SUN CLUSTER GEOGRAPHIC EDITION WITH VIRTUALIZATION TECHNOLOGIES

Important note: this paper was originally published before the acquisition of Sun Microsystems by Oracle in 2010. The original paper is enclosed and distributed as-is. It refers to products that are no longer sold and references technologies that have since been re-named.



USING SOLARIS™ CLUSTER AND SUN™ CLUSTER GEOGRAPHIC EDITION WITH VIRTUALIZATION TECHNOLOGIES

Tim Read, Solaris Cluster Engineering

Sun BluePrints™ On-Line

Part No 820-4690-10
Revision 1.0, 4/29/08

Table of Contents

Chapter 1
Introduction

Virtualization technologies have become more prevalent over the past few years as enterprises have striven to drive up the utilization of datacenter assets in order to maximize the return on capital investments. These technologies also allow organizations to manage datacenter assets more efficiently and effectively. Consequently, virtualization now permeates all aspects of the enterprise IT environment — servers, storage, networking, and software. In many cases, the goal is to consolidate older business or mission-critical systems onto newer, virtualized environments. However, resulting deployments must achieve the same levels of availability and disaster protection as the previous incarnations.

Today, Sun offers several virtualization technologies to address the needs of organizations: Dynamic System Domains, Logical Domains, and Solaris™ Containers. Additionally, open source and third party products such as the Sun xVM hypervisor[1] and VMware are available. Each of these technologies has its own set of benefits and restrictions when combined with the Solaris Cluster software or its Open High Availability Cluster open source equivalent.

As there are several other Sun Blueprints™ and white papers available [4, 6, 11] that cover the aforementioned virtualization technologies, they are not explained in detail here. Instead, this document considers the merits of each virtualization technology, or conversely, the restrictions it might impose on potential solution stacks.

Before proceeding, it is appropriate to set some expectations regarding the contents of this Sun Blueprint. After reading this the reader should have a clear understanding of what the intersection of Sun's virtualization technologies, the Solaris Operating System (Solaris OS), and Solaris Cluster technologies can offer. There is no *magic* solution that solves all problems or satisfies all user requirements. Instead, it is necessary to deploy a pragmatic mix of alternatives to meet particular datacenter constraints.

Readers might also notice what appears to be inconsistent naming of Sun's cluster product throughout this document. The nomenclature is, in fact, correct due to the name change from Sun Cluster to Solaris Cluster with the release of version 3.2 in January 2007.

Finally, as this is a rapidly evolving technology area, this Sun Blueprint gives a snapshot of the state of affairs at the point of publication. It is anticipated that this document will be revised when changes in circumstances warrant an update.

1.First available in Solaris Express Developer Edition 1/08.

Chapter 2

To Implement or Not To Implement?

Before launching into the complex discussions of how clustering and virtualization intersect, it is worth stepping back and considering the reasons for virtualizing and clustering. From this analysis it might become clear that, in some circumstances, the aims of the technologies actually conflict.

The two fundamental motivations here are efficiency, in general measured as maximized resource utilization, and availability as demanded by the service levels agreed with business users or end users. By consolidating services onto fewer platforms, the impact of an individual platform failure increases, because the fault containment offered by multiple separate systems correspondingly diminishes[1]. Does this mean that systems that demand high availability should never be virtualized and consolidated? The general answer is no. There are some cases where the effect is negligible, however there are other scenarios where this incurs greater disruption when viewed from an organizational level.

First, consider a multitiered service, such as SAP, or more generally Service Oriented Architectures (SOA) that consists of Web, application, identity, portal, and database servers. The service as a whole might well be rendered unavailable if any of these layers fails, regardless of the platform on which the tier resides. Consequently, (vertically) consolidating or virtualizing the tiers onto fewer physical servers is unlikely to worsen the failure modes of the service. The only remaining consideration is the recovery time of the service when all of the various server layers are competing for resource when the services are restarting.

In contrast, a (horizontal) consolidation of multiple separate services, e.g., test and development with engineering or human resources (HR) with sales and order entry means that multiple services can be affected by a single event. So when measuring availability from an overall organizational level, this can lead to a lowering of this metric. This might be undesirable, but it is a business decision rather than a technical one.

It is this aspect of fault containment that you should be remembered as the various virtualization technologies are covered. A single Solaris 10 OS instance can host multiple containers and while the failure of one container might not affect another, the failure of the Solaris instance impacts all of the containers hosted. In contrast, a hypervisor can host multiple operating systems, where each operating system can fail without affecting the others hosted. Only if the hypervisor or the hardware itself fails are all of operating systems impacted.

1.This is only partly true of servers with the Dynamic System Domain capability. Only virtualized services within a Dynamic System Domain are affected if the domain fails. Other domains within the platform are unaffected.

While it is more common for mean time between failure (MTBF) statistics to be available for hardware components, since the components wear out, it is less common for that data to be available for software or operating systems. Maturity and the ability to detect and recover from faults also plays a major part in the reliability of a piece of software. Therefore, it is difficult to provide any firm guidance here as to the relative likelihood of a software versus a hardware failure.

Chapter 3

How Solaris™ Cluster Benefits Virtualized and Non-Virtualized Environments

Readers with a knowledge of some virtualization products on the market might wonder why there is any need to use what might be termed as the traditional clustering that Solaris Cluster offers. So it is valuable to describe some of the features that Solaris Cluster offers and show how these address a broader range of requirements than simply the ability to migrate live workloads to other nodes in the cluster [14].

One of the most important features of Solaris Cluster is its resource group manager, which maintains tight control of the starting, stopping, and monitoring of services, held in entities known as resource groups. By providing strong control, it helps ensure that a service is never started elsewhere before it is stopped in its former location. This helps to ensure data integrity — a key constituent of achieving high levels of availability. This becomes more important in a virtualized environment, where services and their containing OS environments are deployed more dynamically and share underlying storage more widely.

However, the resource group manager alone is not particularly valuable without resource types to orchestrate the control of these services. Sun provides more than 40 supported agents (resource types for particular data services) for applications such as Oracle[1], SAP, NFS, Web services (HA and scalable), application servers, etc. [13]. The key point is that the application probes contained in these agents address the issue of application monitoring that the black-box approach of simple workload migration, e.g., VMotion, do not address. Here, the black-boxes are only monitored as an operating system instance without any knowledge of the service contained within.

Broad application coverage is achieved through the Solaris Cluster generic data service (GDS) agent, which allows organizations to provide start, stop, and probe scripts for any custom application and bring it into the same general framework. This enables components within the entire software stack to be constantly monitored to ensure that the end-to-end service continues to function.

Service outages are not just the result of application or server hardware failure, they can also stem from networking or storage problems. Solaris Cluster has the built-in capability to continuously probe the health of the networks and storage on which the system and services depend. That's why Solaris Cluster uses IP MultiPathing (IPMP) to monitor and recover from public network failures, Solaris I/O Multipathing software

1.While the Sun HA-Oracle and 9i RAC agents probe the Oracle database for problems, the Oracle 10g RAC agent does not. Instead the 10g agent's manageability features provide additional coordination with Oracle Clusterware. This is because from Oracle 10g onwards, Oracle Clusterware (running on top of Solaris Cluster) is responsible for managing the availability of its own resources: database, listeners, logical hosts, etc.

(aka MPxIO) and disk path monitoring to manage and recover from storage connection failures, and Solaris Volume Manager (or Veritas Volume Manager) and zpool mirroring to manage and recover from storage failures. All of these technologies allow the resource group manager and kernel-based cluster membership monitor (CMM) to make informed decisions about the automatic failover of services, or in more serious cases, the rebooting of one or more cluster nodes.

Solaris Cluster has the capability to handle failover containers (see "Failover containers" on page 20) and other guest operating systems that reside in virtual containers as black-boxes. The benefit of this is greater flexibility in the range of objects that can be made highly available, e.g., operating system other than the Solaris OS. However, the down side of this approach is that there is a lack of fine grain control within these objects.

Solaris Cluster can achieve the near continuous service availability offered by live migration under certain circumstances. This is accomplished either through specialized applications, such as Oracle RAC (9*i*, 10*g*, or 11*g*), or using scalable services.

A scalable service, e.g., a Sun Java™ System Web Server service, has an instance of the Web server running on multiple cluster nodes concurrently. Web requests for http or https pages come in through a global IP address hosted on one of the cluster nodes. Requests are then dynamically distributed to the appropriate node, using one of several load balancing policies, to be serviced by the Web server. It then sends the reply out through the local network connection. This architecture not only scales throughput, something that is not achieved by live migration alone, but also offers an almost continuous service. The only pause occurs should the node hosting the global IP address fail, requiring a failover of the IP address to a new node. This only takes a matter of seconds, after which the service continues.

Solaris Cluster also has the advantage of working on both SPARC® and x86/x64 systems, although it is restricted to Solaris OS only.

The final advantage of using Solaris Cluster is that it provides seamless integration for disaster recovery with Sun™ Cluster Geographic Edition. The latter provides a homogeneous interface to a number of replication technologies that enable services to be migrated from a primary site to a remote disaster recovery site under the control of the Sun Cluster Geographic Edition framework.

Based on the approach Solaris Cluster takes to service level management, the live migration capabilities of some virtualization software should be considered, first and foremost, as a workload management feature, rather than availability management feature. After all, it is impossible to live migrate an operating system that has already failed.

Chapter 4

Virtualization Compatibility

There are three distinct axes to the virtualized technology stacks: the chip architecture, the operating system, and the clustering and disaster recovery software. Each stack exerts a different set of constraints on the combinations that can be used. This is followed by a section covering applications that require special consideration. Finally, there is a discussion on current suitability of these complex, multitechnology, multivendor solutions for business- and mission-critical services.

Sun's virtualization technologies are evolving quickly, so it is important to define the terminology used in the subsequent sections.

Sun xVM Server appliance consists of an integrated management browser user interface (BUI) with Sun hypervisor solutions for SPARC chip multithreading (CMT) and x86/x64 platforms. On SPARC processors implementing the CMT technology, such as the UltraSPARC T1 or T2 processors, Sun xVM Server software is implemented using the Logical Domains (LDoms) technology. On x86/x64, Sun xVM Server software is implemented using the Sun xVM hypervisor, an implementation based on the work of the XEN community project [12].

As xVM Server has not yet been released, both xVM hypervisor (x86/x64) and Logical Domains (SPARC, and CMT) are currently considered separately. To avoid any confusion, the term Logical Domains or LDoms is used to reference the hypervisor on SPARC CMT. Unless otherwise specified, the term Sun xVM hypervisor is used to reference the hypervisor on x86/x64.

Axis One — Instruction Architecture

Although the discussion in this Sun Blueprint is confined predominantly to the Solaris 10 Operating System and SPARC and x86/x64 processor-based systems, it still results in a non-trivial matrix of potential solution stacks. (OpenSolaris and Open High Availability Cluster can also run on third party hardware, but since support is not available for this combination, this option is not be considered.) This expands further when BrandZ zones ("Branded Zones (BrandZ)" on page 20), non Solaris logical domains, and third party products such as VMware are considered. This document restricts consideration to VMware ESX 3.0.1 [2] release or above, as it contains features necessary to support some of the core Solaris Cluster functions, i.e., quorum disks.

It is helpful to begin by listing the virtualization technologies pertinent to each instruction architecture, as illustrated in Table 1. This table only details the instruction set and operating system matrices.

The term *available* means that an implementation of the technology is available to run on that particular chip architecture, while *Sun supported* indicates that the technology is directly supported by Sun.

Table 1. Virtualization support for the SPARC and x86/x64 chip architectures

	Solaris 10 Containers	Dynamic System Domains	Logical Domains	Sun xVM hypervisor	VMware ESX 3.x
SPARC	Available, Sun supported.	Hardware platform specific. Sun supported.	Hardware platform specific. Sun supported.	N/A	N/A
x86/x64	Available, Sun supported.	N/A	N/A	Available as Sun xVM hypervisor on OpenSolaris.	Available, Sun supported.

Because they are hardware specific, the entries for Dynamic System Domains and LDoms require further documentation. The current Sun servers supporting domains are the Sun Fire™ 4900 and 6900 servers, Sun Fire E20K and E25K servers, and Sun SPARC Enterprise M4000, M5000, M8000, and M9000 servers, while those supporting LDoms are the Sun SPARC Enterprise T1000 and T2000 servers, Sun SPARC Enterprise T5120, T5140, T5220, and T5240 servers, and Sun Blade™ T6300 and T6320 Server Modules.

Hardware support for VMware ESX is governed by VMware's support matrices, details of which are available on-line [10].

Axis Two — Operating System

While Sun recommends migrating applications to the latest Solaris 10 OS, it is recognized that not all enterprise applications can be readily moved, either for reasons of cost, ISV support, or software compatibility. Therefore, when considering how virtualization might be of benefit, it is also important to understand the intersection of these technologies with older Solaris releases and, potentially, other operating systems such as Linux and Microsoft Windows.

In Table 2, *available* means that this is a viable technology mix and *Sun supported* means that Sun can provide direct support for that combination. Other options have appropriate qualifiers on their entries, e.g., a *not supported* entry means that it is currently not supported by Sun and there are no plans to support or implement the combination. Those entries marked with an asterisk have hardware restrictions that reference Table 1.

Table 2. Virtualization support for Solaris, Linux and Microsoft Windows operating environments

	Solaris 10 Containers	Dynamic System Domains	Logical Domains	Sun xVM hypervisor	VMware ESX 3.x
Solaris 8	Available, Sun support for branded zone running Solaris 8 binaries	Available, Sun supported	Not supported	Not supported	Not supported
Solaris 9	Available, Sun support for branded zone running Solaris 9 binaries	Available, Sun supported	Not supported	Not supported	Available, Sun supported(*)
Solaris 10	Available (native), Sun supported	Available, Sun supported	Available (native), Sun support with Solaris 10 11/06 onwards (*)	Available as guest on Sun xVM hypervisor with OpenSolaris, no direct Sun support	Available, Sun supported(*)
Linux	Available, Sun support for branded zone running Linux binaries	N/A	Available with Ubuntu Linux, but not supported by Sun	Available as guest on Sun xVM hypervisor with OpenSolaris, no direct Sun support	Available, Sun supported
Microsoft Windows	N/A	N/A	N/A	Available as guest on Sun xVM hypervisor with OpenSolaris, no direct Sun support	Available, Sun supported

Axis Three — Solaris Cluster and Sun™ Cluster Geographic Edition release

The Solaris Cluster software provides applications with a framework to make them highly available. Solaris Cluster is tightly integrated with the Solaris kernel, bringing a number of benefits to services that use this framework [3].

In contrast, Sun Cluster Geographic Edition is a layered product that runs in conjunction with Solaris Cluster to give services a common disaster recovery mechanism, regardless of whether services use storage-based replication technologies, such as EMC's SRDF and Sun StorageTek™ TrueCopy software, or host-based software replication products like Sun StorageTek Availability Suite.

As with the Solaris 10 OS, Solaris Cluster 3.2 2/08 offers the latest set of features for making services highly available, yet enterprises might have good reasons to maintain applications on existing Sun Cluster and Solaris Cluster deployments. As the product develops, the support for virtualization has increased, consequently it is important to understand what options each platform offers (Table 3).

Table 3. Sun and Solaris Cluster support for virtualization technologies

	Solaris 10 Containers	Dynamic System Domains	Logical Domains	Sun xVM hypervisor	VMware ESX 3.x
Sun Cluster 3.1 8/05	Support limited to failover Solaris 10 containers	Solaris 8, 9, and 10 can be clustered between domains, providing the OS release is identical and the platform supports the Solaris OS version	Not supported	No support for Solaris Cluster running inside a control (*domo*) or guest (*domU*) domain	Not officially supported by Sun but see [2]
Solaris Cluster 3.2	Support for both zone nodes and failover containers	Solaris 9 and 10 can be clustered between domains, providing the OS release is identical and the platform supports the Solaris OS version	Supported in I/O domain only using LDom software v1.0.1 (or above)	No support for Solaris Cluster running inside a control (*domo*) or guest (*domU*) domain	Not officially supported by Sun but see [2]
Solaris Cluster 3.2 2/08	Support for both zone nodes and failover containers	Solaris 9 and 10 can be clustered between domains, providing the OS release is identical and the platform supports the Solaris OS version	Supported in I/O domain only using LDom software v1.0.1 (or above)	No support for Solaris Cluster running inside a control (*domo*) or guest (*domU*) domain	Not officially supported by Sun but see [2]

Part of the complexity displayed in Table 3 is due to how and where Solaris Cluster actually runs and what constitutes a cluster node. A more detailed explanation of the possible scenarios is provided in the individual virtualization sections below.

Sun Cluster Geographic Edition is layered on top of Solaris Cluster and provides the facilities to coordinate both the direction of data replication and control the site on which a service is considered the primary instance. The major benefits of Sun Cluster Geographic Edition are its homogeneous interface to the underlying replication technology and its integration with the tight controls Solaris Cluster exerts on starting and stopping applications. In most cases, Sun Cluster Geographic Edition inherits its virtualization support by virtue of its dependence on Solaris Cluster.

Sun Cluster Geographic Edition relies on various replication technologies to provide the underlying data continuity between the clusters participating in the disaster recovery solution. Consequently, virtualization can only be used as the replication technology allows. An important configuration limitation to highlight here pertains to Sun StorageTek Availability Suite configurations using zone nodes (see “Zone nodes (or static non-global zones)” on page 22). Here, the list of cluster nodes on which the service can run (the application's resource group node list) is specified to be something like: nodeA:zoneX,nodeB:zoneY, i.e., a set of zone nodes. However, the node list for the replication storage resource group is: nodeA,nodeB, i.e., a set of physical nodes. The mismatch in node list means that the Sun Cluster Geographic Edition is not able to implement such a configuration at this time. This restriction does not apply to failover containers nor to storage-based replication products (see “Failover containers” on page 20) because there, the resource group node list for these entities consists entirely of physical nodes or zones nodes, but not a mixture.

The current implementation of Sun Cluster Geographic Edition (3.2 02/08) also prevents the use of Solaris ZFS or Solaris Volume Manager. Both restrictions are expected to be lifted in future releases of the product.

Table 4 shows the compatibility of the currently supported replication technologies with Sun Cluster Geographic Edition, running on top of Solaris Cluster, when used with the supported virtualization options.

Table 4. Compatibility of Replication, Virtualization and Solaris Cluster/Solaris Cluster Geographic Edition technology

	Solaris 10 Containers	Dynamic System Domains	I/O Logical Domains
Sun StorageTek Availability Suite	Not supported (see above)	Supported, (but excludes Oracle RAC)	Supported, (but excludes Oracle RAC – see Table 5)
Sun StorageTek TrueCopy	Supported	Supported	Supported, (but excludes Oracle RAC – see Table 5)
EMC Symmetrix Remote Data Facility (SRDF)	Supported	Supported	Supported, (but excludes Oracle RAC – see Table 5)

Special Cases

Some ISVs require additional testing processes that must be passed before a particular clustering or virtualization product is certified. Oracle's product set is a good example — only certified combinations are listed in Oracle's MetaLink support matrix when the combinations have undergone and passed these tests.

Oracle databases can be deployed in one of two ways on a cluster: either as a failover service using the Solaris Cluster HA Oracle agent or as a Oracle Real Application Cluster (RAC) deployment (where a single database is accessed from multiple nodes concurrently). The latter is the more restrictive case because it contains its own built-in clustering technology, called Clusterware (from release 10*g* onwards), that has to co-exist with Solaris Cluster for it to function correctly when Solaris Cluster is being used.

For brevity, only the most recent releases of Oracle are considered in the table below, i.e., Oracle 10*g* R2 and Oracle 11*g*. Datacenters with older releases of software might be constrained primarily by what is supported by the vendor.

As Table 3 showed, there is no official Solaris Cluster support for Sun xVM hypervisor, or VMware yet, so these options have been omitted from Table 5.

Table 5. Oracle support for virtualization technologies on Solaris Cluster

	Solaris 10 Failover Containers	Solaris 10 Zone nodes	Dynamic System Domains	I/O Logical Domains
HA-Oracle 10*g*	Supported by Sun and Oracle on 3.1 8/05 and above	Supported by Sun and Oracle on 3.2 and above	Supported by Sun and Oracle	Not currently supported by Oracle
HA-Oracle 11*g*	Supported by Sun and Oracle on 3.2 2/08 and above	Supported by Sun and Oracle on 3.2 2/08 and above	Supported by Sun and Oracle on 3.2 2/08 and above	Not currently supported by Oracle
Oracle 10*g* R2 and 11*g* RAC	N/A	Not currently supported	Supported by Sun and Oracle, effectively running RAC in the global zone, if the OS is Solaris 10	Not currently supported by Oracle

It is worth noting that neither Oracle 10*g* or 11*g* RAC support the use of Solaris projects to control the resource utilization of the database under their control. This is because the Clusterware framework is started from `/etc/init.d` and inherits the system project resource limits.

Suitability of virtualization options for business or mission-critical Solaris Cluster implementations

Clearly, systems that are either business- or mission-critical need to be supported by the respective hardware and software vendors to ensure that outages or problems are resolved quickly by the relevant parties. In this case, support equates to the ability to log support *tickets* or calls with the vendors and have them provide advice, procedures, or software fixes to resolve a given issue. (This might also require the vendors to have cooperative support arrangements.)

With the diversity of potential implementations, it is helpful to understand which are suitable for mission- and business-critical deployment, i.e., fully supported by the relevant vendors, and which are only suitable for test and development. Table 6 provides these details.

Table 6. Suitability of various virtualization and availability software stacks for business and mission critical services.

Technology Stack	Suitability
Sun servers and storage, Solaris OS, Solaris Cluster using failover containers, zone nodes, or domains	Mission- and business-critical: configurations supported subject to Sun internal testing matrix
Sun servers, third-party storage, Solaris OS, Solaris Cluster using non-global zones or domains	Mission- and business-critical: configurations supported subject to Sun and third-party testing
Third-party server, third-party storage, Solaris OS, Solaris Cluster using non-global zones	Mission-and business-critical: configurations supported subject to Sun and third-party testing
Sun servers, Sun or third-party storage, Solaris OS, Solaris Cluster running in a LDom	Mission- and business-critical: I/O LDom configurations supported subject to Sun internal testing
Sun servers and storage, Solaris OS, Solaris Cluster under VMware	Test and development only: not officially supported by either vendor
Sun servers and storage, OpenSolaris, Open High Availability Cluster	Test and development only: not officially supported by Sun
Other permutations	Test and development only: not officially supported by Sun

Chapter 5

Virtualization Technologies

Each virtualization technology possesses a different set of benefits, challenges, and constraints, as indicated previously. The next sections consider each of these technologies in turn, examining aspects such as provisioning and patching, as well as providing technical insights into why clustering support might be restricted in certain cases.

It is first helpful to define precisely what a Solaris Cluster node is and how this corresponds to the virtualization technologies described below.

A cluster node is a machine that is running both the Solaris Operating System and Sun Cluster software. A cluster node is either a current member of the cluster (a cluster member), or a potential member.

Cluster nodes are generally attached to one or more multihost devices. Nodes that are not attached to multihost devices use the cluster file system to access the multihost devices. For example, one scalable services configuration enables nodes to service requests without being directly attached to multihost devices.

All nodes in the cluster are grouped under a common name (the cluster name), which is used for accessing and managing the cluster.

Public network adapters attach nodes to the public networks, providing client access to the cluster.

Cluster members communicate with the other nodes in the cluster through one or more physically independent networks. This set of physically independent networks is referred to as the cluster interconnect.

Every node in the cluster is aware when another node joins or leaves the cluster. Additionally, every node in the cluster is aware of the resources that are running locally as well as the resources that are running on the other cluster nodes.

Nodes in the same cluster should have similar processing, memory, and I/O capability to enable failover to occur without significant degradation in performance. Because of the possibility of failover, every node must have enough excess capacity to support the workload of all nodes for which they are a backup or secondary.

Each node boots its own individual *root* (/) file system.

This means that a Solaris Cluster node equates to any of the following entities:

- A dynamic system domain
- A logical (I/O) domain
- A global zone on a Solaris 10 Operating System

- Solaris Cluster running on the Solaris OS as a guest operating system under VMware (although this is not officially supported yet – see [2])

Dynamic System Domains

Dynamic System Domains, often shortened to just *domains*, have been available for more than a decade. One of the first widely deployed systems to include this feature was the Sun Enterprise™ 10000 server[1] introduced in 1997. Many of Sun's current high-end systems use this capability to allow flexible and dynamic allocation of CPU/memory boards and I/O trays, connected to the center-plane, to one or more electrically isolated domains. Current servers with this capability include:

- Sun Fire E4900 and E6900 servers
- Sun Fire E20K and E25K servers
- Sun SPARC Enterprise M4000, M5000, M8000, and M9000 servers

Domains are configured through a system controller. Because domains are electrically isolated from one another, each domain is immune to the failures that might occur in other domains within the same chassis. Furthermore, because the center-planes of these systems are designed with a point-to-point topology, bandwidth between chassis slots (for CPU/memory cards and I/O trays) is not shared, providing some level of guarantee of performance. This approach offers lower complexity than, say, a hypervisor-based system that is sharing the underlying hardware between multiple guest operating systems. In such a configuration, guest operating systems share the memory and I/O channels, making it more difficult to pinpoint performance problems.

Each domain within a chassis runs a completely separate instance of the Solaris OS that is installed manually or through a provisioning system such as: Sun xVM Ops Center, Sun N1™ System Manager, Sun N1 Service Provisioning System (with the OS provisioning plug-in), Solaris JumpStart software, Jumpstart Enterprise Toolkit (JET), etc. With the exception of the Sun SPARC Enterprise MX000 server series and other UltraSPARC VI processor-based systems (Solaris 10 OS only), all of the systems mentioned above support the Solaris 8 OS onwards on UltraSPARC IV 1.35 GHz processors. For higher clock rates, support is limited to Solaris 9 OS and above. However, in all cases, the Solaris OS interacts directly with the underlying hardware rather than with an intermediate hypervisor, as is the case with Sun xVM Server (either on SPARC or x86/x64 processor-based systems).

Because each domain contains an independent instance of the Solaris OS, Solaris Cluster can be used to construct distinct clusters between any two or more domains, whether the domains are in the same or in separate chassis, as illustrated in Figure 1. These clusters have their own resource group manager to control the individual services contained in the resource groups.

However, this mean that each of these Solaris instances need to be patched independently, unlike Solaris 10 Containers. (See “Solaris Containers” on page 18.)

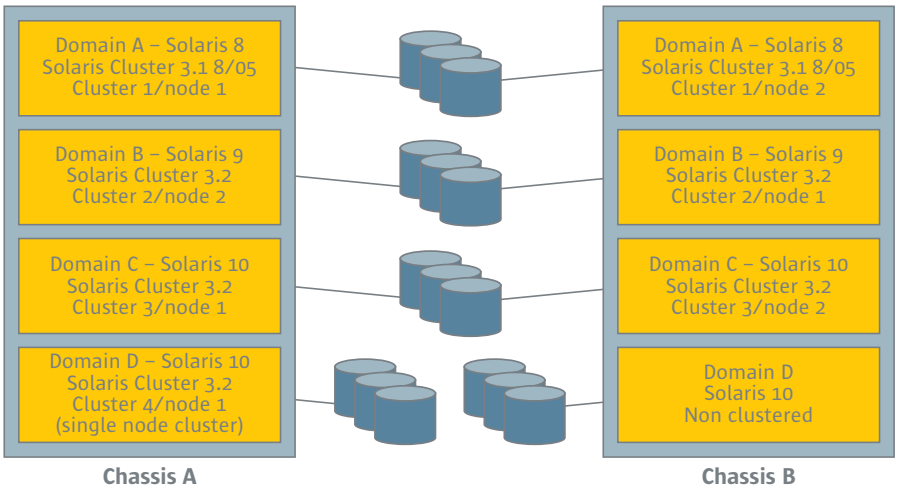


Figure 1. Creating multiple distinct clusters within a single chassis using dynamic system domains (ignoring network connectivity)

If the cluster uses the Solaris 10 OS then Solaris Containers can be used to isolate services, from both a security, resource, and fault perspective. Additional hardware resources: CPU/memory boards and I/O trays can be dynamically reconfigured into and out of the domains with the exception of any CPU/memory boards holding portions of the kernel memory cage.

Logical Domains

Logical Domains, or LDoms, is a facility provided by Sun servers that use the CoolThreads™ technology, allowing one or more virtualized instances of Solaris 10 OS (11/06 or above) to run on the same underlying hardware. This capability is part of the broader Sun xVM Server family of hypervisors. Systems that support this feature include:

- Sun SPARC Enterprise T1000 and T2000 servers
- Sun SPARC Enterprise T5120 and T5220 servers
- Sun SPARC Enterprise T5140 and T5240 servers
- Sun Blade T6300 and T6320 Server Modules

LDoms technology differs from Dynamic System Domains insofar as it accesses the underlying hardware through a thin layer of software, held in firmware, known as a hypervisor [4]. The physical hardware is then presented to the logical domains as a set of virtual resources. With the LDom v1.0.1 software release, only virtual CPU resources can be dynamically added and removed from a domain running the Solaris OS. Other resources, such as memory and I/O, can only be reconfigured at the next reboot of the

domain. However, in contrast to a non-CPU constrained zone, i.e., one that does not have a CPU cap in place, an LDom can only run on the CPUs that are allocated to it.

The *Beginners Guide to LDoms: Understanding and Deploying Logical Domains* [4] defines the different roles for logical domains:

> There are several different roles for logical domains, and these are mainly defined by context; their usage defines them. A domain may have one or more of these roles, such as combining the functions of an I/O and service domain:

- Control domain —Creates and manages other logical domains and services by communicating with the hypervisor.
- Service domain — Provides services, such as a virtual network switch or a virtual disk service, to other logical domains.
- I/O domain — Has direct ownership of and direct access to physical input/output devices, such as a PCI Express card or a network device. Can optionally share those devices to other domains by providing services.
- Guest domain — Presents a virtual machine that subscribes to services provided by service domains, and is managed by the control domain.

The control domain runs the Logical Domain Manager software that creates and configures other logical domains on the system. Currently, only I/O domains are supported by Solaris Cluster 3.2 2/08 [5], as illustrated in Figure 2, though this restriction is expected to be relaxed in the future. To comprehend why this is the case, it is important to understand some of the key components of each technologies to recognize how they work together in order for the products to function correctly.

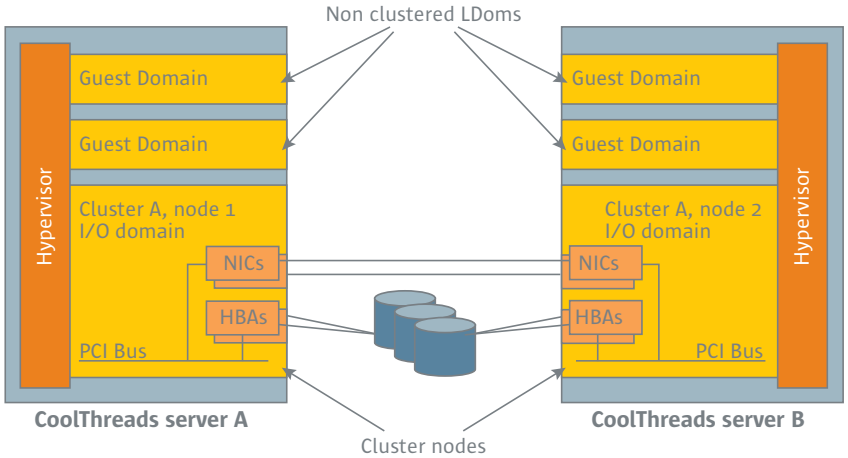


Figure 2. Clustering two I/O domains between two CoolThreads technology-based servers

An I/O domain is a logical domain where the devices on a PCI bus are directly under the control of the Solaris kernel. In these circumstances, the entities that interact with the kernel disk device drivers represent *real* storage devices, however the logical units

(LUNs) are presented, i.e., a just a bunch of disks (JBOD) disk or a hardware RAID LUN, etc. The important point is that they interact through the native Solaris `sd` device driver. In contrast, the storage devices mapped into a guest domain are not always *real* disks, indeed they could be files allocated in some file system (UFS, Solaris ZFS, or VxFS) in a service domain. The crucial point is that there is an intermediate layer of software that is intercepting the guest domain's I/O requests to these virtual devices that ultimately converts them into physical reads, writes, and ioctl calls.

One of the strongest features of Solaris Cluster is its resilient quorum and fencing algorithms that protect against split brain (partition in space), amnesia (partition in time), and data corruption [3, 9]. For these to function correctly within a guest domain, the ioctl calls made to quorum devices and device groups need to be handled correctly. In turn, this requires any intermediate drivers to accurately transform these calls into the appropriate actions, in most cases this necessitates additional engineering development work and testing at Sun.

The *I/O domain only* constraint (see Table 3 on page 9) bestows an architectural limit on the number of logical domains that can be used as a cluster node, per chassis. This applies even when considering the most minimal of cluster node configuration, i.e., only two physical network interface cards (NICs), each of which has two tagged virtual local area networks (VLANs) configured (one for public and one for heartbeat) networks, and one dual port Fibre Channel host bus adapter (HBA), which is used for both storage are network (SAN) boot and shared storage. This is detailed in Table 7.

Table 7. Maximum number of I/O domains that can be used as cluster nodes per chassis

System	Maximum number of I/O LDoms	Number of I/O LDoms for a usable Solaris Cluster configuration
Sun SPARC Enterprise T1000 server	Two	One (due to limited PCI slots)
Sun SPARC Enterprise T2000 server	Two	Two
Sun SPARC Enterprise T5120 server	One	One
Sun SPARC Enterprise T5220 server	One	One
Sun SPARC Enterprise T5140 server	Two	Two
Sun SPARC Enterprise T5240 server	Two	Two
Sun Blade T6300 Server Module	Two	One
Sun Blade T6320 Server Module	One	One

As with Dynamic System Domains, this virtualization technique does not decrease the number of operating systems that need to be managed. Each individual Solaris OS deployment needs to be provisioned and patched separately.

While it is possible to configure an I/O domain used as a cluster node to act as a service domain for other guest domains, it is important to consider the unpredictable load the guest can generate and overall complexity involved. The latter is often a key cause of decreased availability. Furthermore, if the I/O domain fails, it not only brings down the cluster node, but also causes the dependent guest domains to appear to hang. This is because the guest domains lose access to the root file systems and cannot become responsive again until the I/O domain becomes available.

A similar issue arises should guest domains be supported as cluster nodes. In order to avoid the service domain supporting the guest domain from becoming a single point of failure, the guest should be constructed in such a way as to balance its constituent components over two separate service domains. Thus, the guest domain's root disk is mirrored across two virtual LUNs from separate service domains. Similarly, IPMP groups should be constructed using virtual adapters from separate service domains.

Solaris Containers

Solaris Containers combine the Solaris 10 zones feature with resource management. Solaris Zones are configured using the `zonecfg` command, which allows devices and file systems to be configured into a zone and allows resource constraints, e.g., memory caps and processor pools, etc. The latter is very valuable when *hard* CPU limits are enforced, as they might be required as part of the ISV software licensing term, i.e., the maximum number of CPUs on which the software can be run. Furthermore, unlike Dynamic System Domains or LDoms, zone are available on any platform that runs the Solaris 10 Operating System.

When a system runs one or more Solaris Containers there is only one Solaris kernel active and this controls the *global zone*. The additional *virtual* Solaris instances, termed *non-global zones*, are represented by a set of extra Solaris processes running in the global zone, e.g., `zoneadmd`, etc. This means that booting a Solaris Container is much faster than booting a full operating system, whether it is in a Dynamic System Domain, an LDom, or a guest domain. Figure 3 illustrates two nodes in a cluster that are configured for container failover.

The performance of applications in a non-global zone is almost identical to the performance in the global zone on a machine without non-global zones. However, it is significantly better than the same application running in an identically configured (with respect to CPU and memory resources) LDom or VMware guest domain, simply because of the additional virtual device drivers and hypervisor software layers that must be traversed in those configurations. The precise impact varies from service to

service, so the only way to measure it is to conduct suitably constructed benchmarks before deployment. The same is also true of Dynamic System Domains when compared with LDoms or VMware guests.

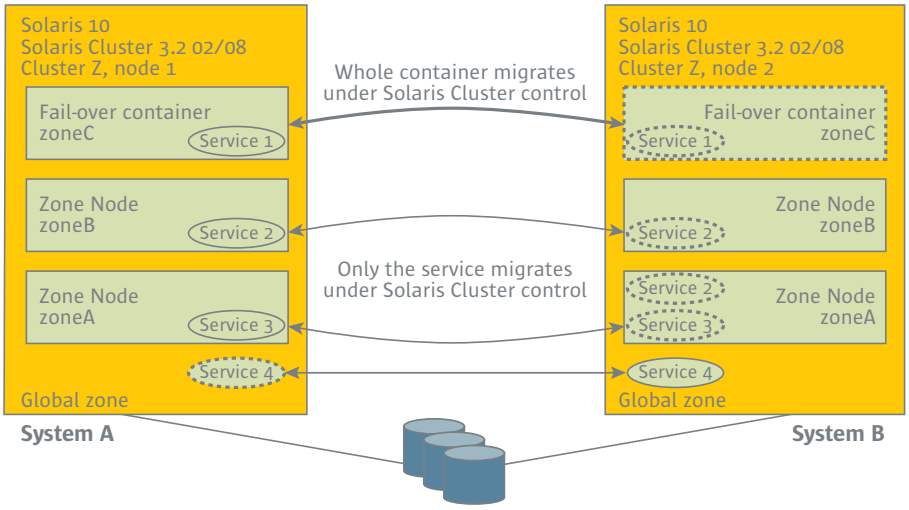


Figure 3. Zone nodes and failover containers participating in a Solaris Cluster

The root directory of a non-global zone is simply a subdirectory of the global zone's root file system, e.g., `/zoneroot/ngz_zone_a`, which contains `./usr`, `./etc`, `./dev` subdirectories as usual, so applications inside a zone are effectively running in a environment where the Solaris root directory is altered. (See the manual page entry for `chroot`(1)).

There are two types of zone that can be configured: *sparse root* zones and *whole root* zones. The difference between the two types is the amount of the global zone installation they share. A sparse root zone shares, among other things, a read-only copy of the global zone's `/usr` and `/lib`, while a whole root zone contains a separate copy of all of the global zone's directories and, by implication, packages. This impacts the memory usage of the two approaches. Where the binaries in a shared root zone point to the same inodes as the copy in the global zone, the full root zone binaries do not. Consequently, the kernel cannot share the memory pages for those binaries in the latter case and thus has a larger memory footprint.

While there is no difference between the patching mechanism of sparse and whole root zones, there is a constraint on how failover zones must be patched. (See "Failover containers" on page 20.) While, in theory, some packages in a whole root zone could be patched to a different level from that of the global zone, it is not considered a best practice and can in some cases cause problems for failover containers. Any requirement for the zone to be booted or shutdown is documented in the patch *README* file.

In general, containers can support a large proportion of the applications that are normally capable of running in the global zone [11]. Three notable exceptions to this are Oracle RAC (9*i*, 10*g* and 11*g*), DHCP[1], and NFS. It is possible that the restriction on running Oracle RAC in Solaris Containers could be lifted in a future release of Solaris Cluster.

The important point to remember is that by deploying Solaris Containers, the number of Solaris instances that need to managed can be minimized.

For further information on support for zones on Solaris OS and Solaris Cluster nodes see the section entitled *Support for Solaris Zones on Sun Cluster Nodes* in [16].

Solaris Live Upgrade can also be used with Solaris Cluster to minimize the disruption caused by patching. Check the Solaris Cluster manual for restrictions.

Branded Zones (BrandZ)

By default, a non-global zone has the same characteristics as the operating system in the global zone, which is running the Solaris 10 Operating System or later release. These native non-global zones and the global zone share their conformance to standards, runtime behavior, command sets, and performance traits in common.

It is also possible to run a different operating system inside of a non-global zone. The branded zone (BrandZ) framework extends the Solaris Zones infrastructure to include the creation of brands, or alternative sets of runtime behaviors. Brand can refer to a wide range of operating system. For example, the non-global zone can emulate another version of the Solaris Operating System, or an operating system such as Linux. Or, it might augment the native brand behaviors with additional characteristics or features. Every zone is configured with an associated brand.

Zones that provide an environment for Solaris 8 (SPARC only) or Linux (x86/x64 only) applications, can be made highly available through the Sun Cluster Data Service for Solaris Containers. Although neither the standard Solaris Cluster data service nor Solaris Service Management Facility is available within these containers, services can be made highly available using the `sczsh` script provided with the Solaris Cluster Data Service for Solaris Containers. This allows developers to write start, stop, and probe scripts.

Failover containers

Sun Cluster 3.1 8/05 was the first Sun Cluster product release to ship with support for Solaris Containers. The Solaris Cluster Data Service for Solaris Containers treats the container as a *black box*.

1.DHCP can run in a zone if it uses the exclusive IP mode.

A failover container has an identical Solaris Zone configured on multiple Solaris Cluster nodes and the *root* directory is stored on a failover file system. Once the zone is installed, the failover file system can be switched between nodes under the control of Solaris Cluster. The start and stop methods simply boot and shut down the container using the appropriate `zoneadm attach` and `detach` commands. The probe script for the agent checks that a container is running the `zsched` process and that the container reaches a suitable milestone boot point.

The Solaris Cluster Data Service for Solaris Containers uses a shell script (`sczbt_register`) and configuration file to define and register a specific set up with the Solaris Cluster framework. When a container fails over, the *virtual* instance of Solaris is stopped on the first node and restarted on another. This should not be confused with a live migration of a running operating system instance that might occur with VMware's VMotion functionality. The services are unavailable for the time it takes to shut the services and container down, switch over the control of the required storage, and then restart the container and the services. Depending on the nature of the services, this can take from as little as a few seconds to several minutes for large databases.

The initial 3.1 8/05 release of Sun Cluster only supported standard Solaris Containers, but allowed both failover and multimaster configurations. Although most of the standard Sun Cluster data services did not support containers at the initial release in 2005, the Samba, WebSphere MQ, PostgreSQL, and Tomcat agents were extended to use the `sczsmf` script supplied with the container agent. This allowed these data services to integrate with the Solaris 10 Service Management Facility feature inside the container. This restarted the service when the container was restarted or when it failed, and check the service's health through a probe script so that it could be restarted if it was considered *unhealthy*.

The current Solaris Cluster 3.2 2/08 release extends support through the HA Container agent for non-global zones of brand type *solaris8* and *lx* (Linux). It also provides the possibility for failover containers configured with the *ip-type=exclusive*, as opposed to *ip-type=shared*, configuration setting. An exclusive IP zone has a dedicated adapter assigned to it. Instead of setting the IP address through the net address properties in the `zonecfg` command, it is configured using the `ipconfig` command from within the zone itself. This approach does not allow the use of a *SUNW.LogicalHostname* or *SUNW.SharedAddress* resource because resources of these types run in the global zone, whereas the exclusive IP address is configured in the local zone itself. Note that Solaris Cluster does not monitor the state of IPMP groups configured within zones that have *ip-type=exclusive* set.

Failover containers have some drawbacks with respect to availability during the patch process. When patches are applied to a system, all configured containers must be available on that node for patching. This requirement means that any failover

containers must be failed over to the node to be patched and the service must be shut down prior to beginning the patch process. Since the service is wholly contained within this single container, there is no other host for it to run on, thus the service is unavailable for the entire duration of the patch process. Furthermore, the whole process must be repeated again when the other node is patched. Although this can be mitigated to some extent by detaching the container from the configuration using the `zoneadm -z <zone> detach` command. For this reason, it is recommended that the zone node approach documented below be used whenever possible.

Zone nodes (or static non-global zones)

Solaris Cluster 3.2 is the first release to provide more comprehensive support for Solaris Containers and most of the data services are container aware[1], with the obvious exception of Oracle RAC, DHCP, and NFS as noted above. With this release, zone names can appear in resource group node lists in a similar fashion to nodes themselves. In this document these entities are termed *zone nodes* to distinguish them from *failover containers*. So, where previously a service might have been defined with a node list equal to *nodeA, nodeB*, it can now have a node list consisting of a mix of full nodes and containers, e.g., *nodeA, nodeB:zoneA* or *nodeA:zoneA, nodeB:zoneA*, etc.

With Solaris Cluster 3.2 and above, it is possible to create a resource group that uses containers and to make a service highly available within the containers. The code below demonstrates how a failover Oracle service, as opposed to an Oracle RAC service, might be configured between two containers named *zonex* and *zoney* on physical cluster nodes *nodea* and *nodeb* respectively using the command line interface (CLI). This can also be performed using the Solaris Cluster Data Services Configuration wizard, which can guide the user through the configuration of a number of data services including Oracle, SAP, and NFS.

1.5. Siebel is another exception, as is Sun Grid Engine, which relies on NFS.

```
Register the resource types as needed
# clresourcetype register -N SUNW.oracle_server
# clresourcetype register -N SUNW.oracle_listener
# clresourcetype register -N SUNW.HAStoragePlus

Create the resource group
# clresourcegroup create -n nodea:zonex,nodeb:zoney oracle-rg

Add the logical host name resource to the Oracle resource group
# clreslogicalhostname create -g oracle-rg oracle-lh

Add a resource of type SUNW.HAStoragePlus to the resource group
# clresource create -g oracle-rg -t SUNW.HAStoragePlus \
-p FileSystemMountPoints=/oracle:/failover/oracle \
-p AffinityOn=TRUE oracle-data-store-rs

Add the Oracle server resource to the resource group
# ORACLE_HOME=/oracle/oracle/product/10.2.0/db_1 export ORACLE_HOME
# clresource create -g oracle-rg -t SUNW.oracle_server \
-p ORACLE_HOME=${ORACLE_HOME} \
-p Alert_log_file=${ORACLE_HOME}/admin/acme/bdump/alert_acme.log \
-p ORACLE_SID=acme -p Connect_string=hamon/H@M0nPwd \
-p Resource_dependencies=oracle-data-store-rs oracle-svr-rs

# clresource create -g oracle-rg \
-t SUNW.oracle_listener -p ORACLE_HOME=${ORACLE_HOME} \
-p LISTENER_NAME=LISTENER_acme oracle-lsnr-rs
```

The *SUNW.HAStoragePlus* resource shows that the file system `/failover/oracle` is mounted in the global zone and then is *loop back* mounted onto `/oracle` in the zone node. The same syntax can be used for a global file system (instead of a failover file system) as long as the `global` flag is present in the `/etc/vfstab` files in the global zones of each of the nodes mentioned in the resource group node list, in this case *nodea* and *nodeb*.

The current release of Solaris Cluster 3.2 2/08 does not support branded (Solaris 8 or LX) zone nodes. However, these types of zones can co-exist alongside other native zones that participate in a resource group's node list. For example, a branded zone called *testlx* can be present on cluster node *node1*, as long as no Solaris Cluster resource groups have *node1:testlx* in their node list. The cluster can use other non-branded zones for the resource group's node list, e.g., *node1:prod – a 'native'*, Solaris Zone.

Unlike failover containers, resource groups that use zone nodes can be failed over to an alternate zone node on another node while its current container is patched. Although this requires a brief service outage, it is limited to approximately the sum of the service stop and start time.

Configuring file systems in Solaris Cluster zone nodes

This section described the how highly available local (failover) file systems and global file systems can be configured into zone nodes used by Solaris Cluster. In cases where there is more than one way to apparently achieve the same result, the recommended approach is noted together with reasons for that choice.

Highly available local file (failover) systems

Highly available local file systems (also sometimes called failover file systems) can be configured into non-global zones, as illustrated in the code below, whether based on UFS, Veritas File System (VxFS), Sun StorageTek SAM-QFS (QFS), or Solaris ZFS. Unlike UFS, VxFS, and QFS file systems that are mounted in the global zone and then loop back mounted into the non-global zone, Solaris ZFS file systems are only mounted directly into the non-global zone.

```
Create a resource group that can holds services in nodea zonex and nodeb
zoney
nodea# clresourcegroup create -n nodea:zonex,nodeb:zoney test-rg

Make sure the HAStoragePlus resource is registered
nodea# clresourcetype register SUNW.HAStoragePlus

Now add a UFS [or VxFS] fail-over file system: mount /bigspace1 to fail-
over/export/install in NGZ
nodea# clresource create -t SUNW.HAStoragePlus -g test-rg \
-p FilesystemMountPoints=/fail-over/export/install:/bigspace1 \
ufs-hasp-rs

Now add a QFS fail-over file system: mount /bigspace2 to fail-over/
export/share in NGZ
nodea# clresource create -t SUNW.HAStoragePlus -g test-rg \
-p FilesystemCheckCommand=/bin/true \
-p FilesystemMountPoints=/fail-over/export/install:/bigspace2 \
qfs-hasp-rs

Now add a zpool (not a file system), the mountpoint is govern by the zfs
file system property
nodea# zfs set mountpoint=/extraspace myzpool/extraspace
nodea# clresource create -t SUNW.HAStoragePlus -g test-rg \
-p Zpools=myzpool zfs-hasp-rs
```

There are several additional restrictions pertaining to failover file systems:

- HA-ZFS is only available with Solaris Cluster 3.2 or above.
- HA-ZFS works at the granularity of *zpools*, so all files associated with a zpool are failed over when the zpool moves. This is equivalent to failover file systems associated with a *metaset* or device group.

- HA-SAM (Sun StorageTek Storage Archive Manager) is only supported with Solaris Cluster 3.2 or above when used in conjunction with global file systems. Even then, there a number of other constraints on the supported configuration.

It is worth noting that failover file systems cannot be configured into zone nodes using `zonecfg` since that would only result in attempts to mount the file system whenever the zone boots.

Global file systems

The preferred method to make a global file system available to zone nodes is by using an *HAStoragePlus* resource in a scalable resource group as shown in the code below.

```
Create a resource group that can holds services in nodea zonex and nodeb
zoney
nodea# clresourcegroup create -p Nodelist=nodea:zonex,nodeb:zoney \
-p Desired_primaries=2 -p Maximum_primaries=2 test-rg

Make sure the HAStoragePlus resource is registered
nodea# clresourcetype register SUNW.HAStoragePlus

Make sure there is an entry in /etc/vfstab in the global zones on all nodes
nodea# grep bs1 /etc/vfstab
/dev/md/myset/dsk/d200 /dev/md/myset/rdsk/d200  /bs1  ufs  2 no
global,logging

Now add a UFS [or VxFS] fail-over file system: mount /bs1 to global/install
in NGZ
nodea# clresource create -t SUNW.HAStoragePlus:6 -g test-rg \
-p FilesystemMountPoints=/global/install:/bs1 \
ufs-hasp-rs
```

This approach is preferred for several reasons:

- It is dynamic — *HAStoragePlus* resources can be configured and removed while the zone is running without requiring a zone reboot.
- It is synchronized with zone boot and shut down — If a zone that should have the file system mounted is initially unavailable, and then is booted, the *HAStoragePlus* resource mounts the file system automatically.
- It includes checks to avoid misconfiguration — This prevents a global file system from mounting at an inappropriate location in the zone, e.g., `/bin`.

The global mount is achieved by using a *loopback* mount from the global zone into the zone node. Because this mount is performed from the global zone, it cannot be unmounted by a *root* user in the zone node.

It is worth highlighting that only global file systems based on UFS and VxFS can be used in this configuration. Shared QFS cannot currently be used as a global file system within a set of zone nodes.

While it is strongly discouraged, it is possible to mount a global file system into the namespace of a set of zone nodes, either directly or by using a loopback mount, as illustrated in the code below. The former requires that the path name for the desired mount be the same for all non-global zones considered.

```
Mounting directly requires a consistent target path name
nodea# mount -g /dev/md/webset/dsk/d1000 /zoneroot/zonea/root/global/webset

Mounting via lofs does not needs a consistent target path name
nodea# mount -F lofs /global/webset /zoneroot/zonea/root/global/webset
nodeb# mount -F lofs /global/webset /zoneroot/zoneb/root/global/webset
```

The primary reason this is not recommended is that there is no coordination with start up and shut down of the zone. Consequently, these operations can have undesirable results, such as preventing a zone from booting or not having the loopback mount present in all zone nodes. Unlike the *HAStoragePlus* resource, no checks are made on where the resulting mount might take place, thus there is a higher possibility of misconfiguration or unanticipated behavior.

Furthermore, none of these options offers the ability to coordinate the primary I/O path of the global mount with a service running in the non-global zones. This is because the control of the primary path would need to be performed by a resource group with a node list consisting of physical nodes: *nodea,nodeb*, while the application in the non-global zone would have a node list consisting of the relevant zones: *nodea:zonea,nodeb:zoneb*. Since the node lists do not match, there is currently no way to specify resource group affinity between these groups using the *RG_affinities* resource group property. For more details on RG_affinities see [15].

Finally, global file systems can be mapped into zones using the `zonecfg` command and the loopback file system as outlined in the code below. Once again, this method is not recommended because of its static nature, i.e., it cannot be modified without rebooting the zone. It can also suffer from boot ordering problems where the zone boots before the global file system provider is available.

```
zonecfg:oracle-zone> add fs
zonecfg:oracle-zone:fs> set dir=/global/oracle
zonecfg:oracle-zone:fs> set special=/global/oracle
zonecfg:oracle-zone:fs> set type=lofs
zonecfg:oracle-zone:fs> end
```

Any attempt to use a `metaset` devicename, e.g., `/dev/md/nfs/dsk/d100` (or a VxVM equivalent) in conjunction with UFS or VxFS in the place of *lofs* in the code above will fail.

Zone nodes and networking

Zone nodes can also be used for scalable services, i.e., services that run on multiple nodes at once. Good example services are Web server or application servers. A global (service) IP address is configured in a resource group that spans the global zones of the relevant nodes, while the Web services are configured in a scalable resource group that comprise the appropriate zones nodes. At present, such configurations can only use the *ip-shared* network option.

The code below shows how to configure a shared address into a resource group that spans multiple zone nodes. Subsequent scalable resource groups that span the same node list can the have resources that bind to the shared address.

IP filtering can be used with zone nodes. The method for setting up the rules is the same as it is for the global zones. Currently the only restriction is that it cannot be used with scalable services. Any dynamic state held by the filter is lost on fail over regardless of whether the service is scalable or not.

```
Create a resource group that can holds services in nodea zonec and nodeb
zoned
nodea# clresourcegroup create -n nodea:zonec,nodeb:zoned shared-addr-rg

Now add a shared address resource to the resource group with a host name
of my-sa-host which has the address 10.11.114.62
nodea# clresource create -t SUNW.SharedAddress -g shared-addr-rg \
-h my-sa-host shared-addr-rs

Bring the resource group on-line
nodea# clresourcegroup manage shared-addr-rg
nodea# clresourcegroup on-line shared-addr-rg

Check the address is plumbed up, on non-primary nodes it will appear on
the loop back interface
-bash-3.00# zlogin -C zonec
[Connected to zone 'zonec' console]

bash-3.00# ifconfig -a
lo0:1:
flags=20010008c9<UP,LOOPBACK,RUNNING,NOARP,MULTICAST,IPv4,VIRTUAL> mtu
8232 index 1
        inet 127.0.0.1 netmask ff000000
bge0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
index 2
        inet 10.11.114.60 netmask ffffff00 broadcast 10.11.114.255
bge0:2: flags=1040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4>
mtu 1500 index 2
        inet 10.11.114.62 netmask ffffff00 broadcast 10.11.114.255
```

No global device support

Global devices, that is Solaris Volume Manager (SVM) metasets, Veritas Volume Manager (VxVM) diskgroups, or the clustered counterparts (SVM/Oban and VxVM/CVM), are not available within a zone node. Applications that are normally configured with raw devices, e.g., database log files, should be configured to use file system storage instead. As long as the file system is mounted with the `forcedirectio` flag, there should be little noticeable performance degradation, though it should be noted that other applications that benefit from file system page caching should be set up to use different file systems.

Sun xVM hypervisor

Sun xVM Server on the x64 architecture is a high efficiency, open source hypervisor capable of hosting multiple guest operating systems, such as Microsoft Windows, Linux, and Solaris. It is built using technology from the Xen open source project.

The hypervisor, or Virtual Machine Monitor (VMM), runs one or more of these virtual machine instances or domains. There is a privileged domain called *dom0* that controls the configuration of other unprivileged guest domains (*domU*), such as OpenSolaris, Solaris, Linux, or Microsoft Windows.

Although Solaris Cluster 3.2 2/08 does not support Sun xVM hypervisor, a project (illustrated in Figure 4) has been proposed to make this functionality available first on the Open High Availability Cluster [7]. Any future Solaris Cluster support for the project is primarily dependent on the official release of Sun xVM hypervisor, i.e., Sun xVM Server for x86/x64, itself.

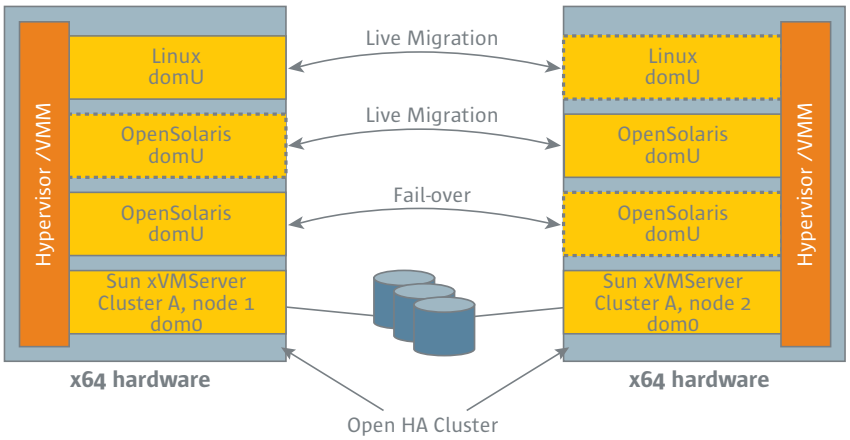


Figure 4. Proposed Open High Availability Cluster project for Sun xVM Server

In contrast to the VMware option described in the next section, the project proposes to run Open High Availability Cluster on OpenSolaris installed in *dom0* on two or more servers. These domains constitute the cluster nodes. The resource group manager within Open High Availability Cluster would then control the fail over and live migration of *domU* guest domains. The implementation is intended to benefit from all the standard features found in Solaris Cluster and Open High Availability Cluster, such as resource group affinity, which could be used to govern which *domUs* are co-resident on the same physical server. The concept is similar to that used for failover containers described "Failover containers" on page 20.

The availability goals of this approach differ somewhat from those for services running on Solaris Cluster. Unlike the latter, which benefits from probe, start, and stop scripts to manage the availability of the service, in this case only the encapsulating operating system is restarted (or migrated) should it fail. No effort is made to probe the services inside the external operating system instance, however, the project is expected to provide hooks to allow users to supply their own probe mechanism.

As each of the *domU* guest environments could have a different operating system, the provisioning and patching procedures for each can be fairly different. Thus, although virtualization might bring benefits in resource utilization, it does not necessarily address management costs, since it does not reduce the number of operating system instances.

While, in theory, it should be possible to create an Open High Availability cluster or Solaris Cluster between cluster nodes using guest domains for the cluster nodes, it is not yet a Sun supported configuration.

VMware ESX 3.x

Solaris Cluster is not officially supported on VMware ESX, but the combination has been shown to be feasible and documented in a 2007 Sun blog entry [2]. Many of the issues preventing automatic certification of this combination are common to VMware, Sun xVM hypervisor, and LDom guest domains, i.e., confirming that the drivers function correctly (see the discussion at the end of "Logical Domains" on page 15).

Unlike the Open High Availability Cluster project for Sun xVM hypervisor, which uses one instance of Open High Availability Cluster running (in *dom0*) on the entire system, the implementation on VMware is different because Solaris (or OpenSolaris) is not running on the Service Console, as illustrated in Figure 5. Instead, subject to hardware configuration and resource constraints, multiple, separate Solaris Cluster installations can be created on a single set of x64 nodes. Each installation has its own resource group manager and is isolated from the other instance by the VMkernel (see Illustration 7 below). Resource groups can then be protected within their individual clusters. Indeed, such a configuration could be used in conjunction with Sun Cluster

Geographic Edition to try out various disaster recovery scenarios, using the individual clusters as primary and secondary sites.

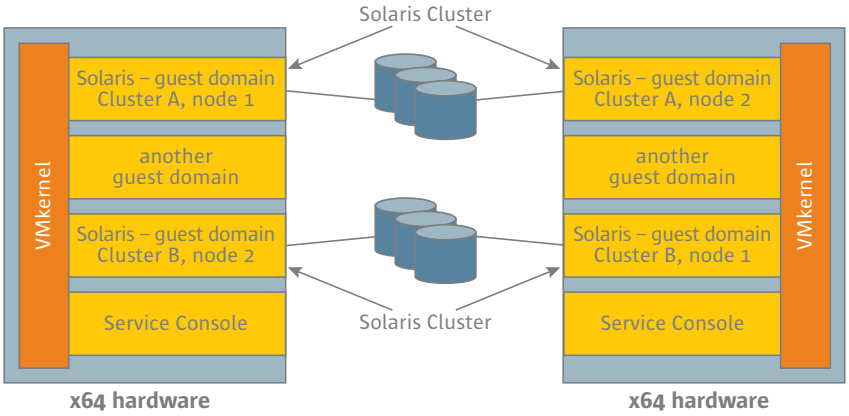


Figure 5. Potential configuration for Solaris Cluster on VMware

As the blog entry indicates, this configuration needs to have the shared storage devices mapped in using Raw Device Mapping (RDM). This helps ensure that key Solaris Cluster features, such as fencing and quorum, work correctly.

When running as a guest under VMware, Solaris interacts with the underlying hardware under the control of the hypervisor. This can lead to problems if the Solaris kernel is not scheduled to run in timely fashion. Delays can keep the cluster heartbeat packets from being sent out. Without the cluster heartbeat packets, the cluster membership monitor determines that one or more of the nodes is dead and forces a reconfiguration to avoid split-brain. Similarly, trying to use VMotion on a running Solaris Cluster is likely to cause the same outcome because of the time it can take to switch over from the old instance to the newly migrated one. A more concrete reason why using VMotion with Solaris Cluster will not work is that the SCSI reservations held on the shared storage are not migrated with the operating system.

Chapter 6

Fault Detection, Isolation, and Containment

High availability is fostered[1] through a combination of fault detection (determining that there is a problem), fault isolation (determining what the cause is), and fault containment (preventing the fault from spreading). The Solaris 10 Operating System includes a in-built fault management facility that can, for example, retire memory pages or CPUs in response to (soft) fault events. This prevents the errors from turning into hard errors that could otherwise cause a kernel panic and bring down the system. An agent can detect the fault, isolate it to a particular component, and in the face of repeated errors, contain it by disabling further operations using that entity. The Logical Domain hypervisor also benefits from the same capabilities. The hypervisor itself participates in handling the traps, presented by the CPUs, to the hyper-privileged software that constitutes the hypervisor. By preventing *dom0* from failing, the hypervisor can contain the fault and prevent its propagation to the other (guest) *domUs*. This capability is not currently available on Sun xVM hypervisor, i.e., the x64 hypervisor platform.

Solaris Cluster includes similar capabilities to detect, isolate, and contain failing cluster nodes. It accomplishes this using a robust, kernel-based membership monitor. Every node sends out low-level data link provider interface (DLPI) packets once per second to each of its peers on each of the private networks. These packets are sent in the kernel interrupt context, making them very resilient to the peaks in system load. A network, or path, between two nodes is only declared down if a heartbeat messages does not complete the round trip between nodes, over that specific path, within the time-out period. The default time-out is 10 seconds and messages are normally sent every second. Both of these times are tunable parameters, but should only be adjusted to a lower values if the cluster load characteristics are bounded and well understood.

In the event that the cluster partitions—one subset of nodes cannot communicate with another subset of nodes (i.e., all the private interconnect paths are down)— the membership monitor goes through a series of lockstep stages to compute a new membership. The process by which Solaris Cluster determines the new membership is described in detail in the section titled *Quorum Devices and Three-Site Configurations* in [8]. When a node finds that it is not a member of a partition with sufficient votes, the node panics itself because there must be only one active cluster. Once the new membership is established, the registration keys of the nodes that do not form part of the new cluster are removed by a node from the surviving portion of the cluster. Additionally, all of the disks that are shared between the two partitions are reserved for sole use by the surviving members of the cluster. This is achieved through SCSI reservation ioctls. It is at this point that the failfast driver on the nodes in the failed

1. High availability in general is achieved through a combination of people, process and products.

partition, which have not already panicked, find that access rights are revoked and panics the node with a SCSI reservation conflict message.

What are the ramification of using Solaris Cluster together with the previously discussed virtualization technologies for consolidation?

Systems consolidated with Dynamic System Domains can benefit from the complete electrical isolation provided by the systems' backplanes. This means that multiple independent clusters can be run without any hardware or software failure from one domain propagating to another. Furthermore, the use of Solaris Containers within those clusters can further serve to allow more fine-grain control with only the instance of Solaris itself as the single point of failure within the domain. Each Solaris Container can fail independently without affecting other containers controlled by that instance of Solaris.

When LDoms-capable systems are used for consolidation, all I/O domains depend on the same internal server infrastructure, so unlike the Dynamic System Domains-capable systems, the I/O domains are not electrically isolated. Therefore, a hardware fault could affect more than one I/O domain. Within the I/O domain, the same considerations apply to subsequent use of Solaris Containers as did with Dynamic System Domains.

Clusters that use Solaris Containers, either as failover containers or zone nodes, benefit from the isolation of the containers. An individual zone can fail and not impact any other zone on that system. Thus the fault is contained to a single zone.

Consolidating on the Sun xVM server is currently limited by the inability of the hypervisor to trap and respond to developing faults, which can lead to the failure of *dom0* and thus to all the managed *domU* domains. However, since Solaris Cluster is not officially supported on this technology, this is a moot point.

Chapter 7

Service Life Cycle Management

A single service can consist of multiple components: a database, application server, Web server, etc. Each component can be deployed in separate Solaris instances. The different virtualization methods have varying requirements when it is necessary to patch these Solaris instances. These difference, in turn, are important when the service is required to be highly available.

When considering the process of provisioning the basic operating system, there is little difference between the virtualization mechanisms when booting from a local disk. In each case, the operating system is provisioned either from DVD, a Solaris JumpStart server, Sun xVM Ops Center, N1 System Manager, or any of a number of other mechanisms. Only Solaris Containers deviate from this model since they are cloned from the packages installed in the global zone of the hosting Solaris instance. Only when the Solaris instance is running can the service component — database, Web server, etc. — be deployed.

To maintain high levels of service, planned downtime (as well as unplanned downtime) need to be kept to a minimum. Typically, the most common reason for planned downtime is for remedial software patching, either to the operating system or to the application software. When the software to be patched is the only copy through which the service can execute then, by definition, the application must be stopped during the patch or upgrade process. Therein lies the problem for services using virtualization that do not also include clustering. When the single container that hosts the software needs to be patched, the service must be stopped. By contrast, an application running on a Solaris Cluster can be switched to a separate Solaris instance, while either the operating system or application software is patched, before the service is switched back to its original, upgraded location. In Figure 6, the boxes on the top show the service availability over time of *SvcA* and *SvcB* on a single node, virtualized platform. While *SvcB* is being patched, it is unavailable. The systems on the bottom illustrate the same services running in a clustered environment, where services *Svc A* and *Svc B* are switched to *node 2* while *node 1* is patched. The services can be switched back to *node 1* if *node 2* needs to be patched as well. In the clustered example, the services are still available while the first node is patched.

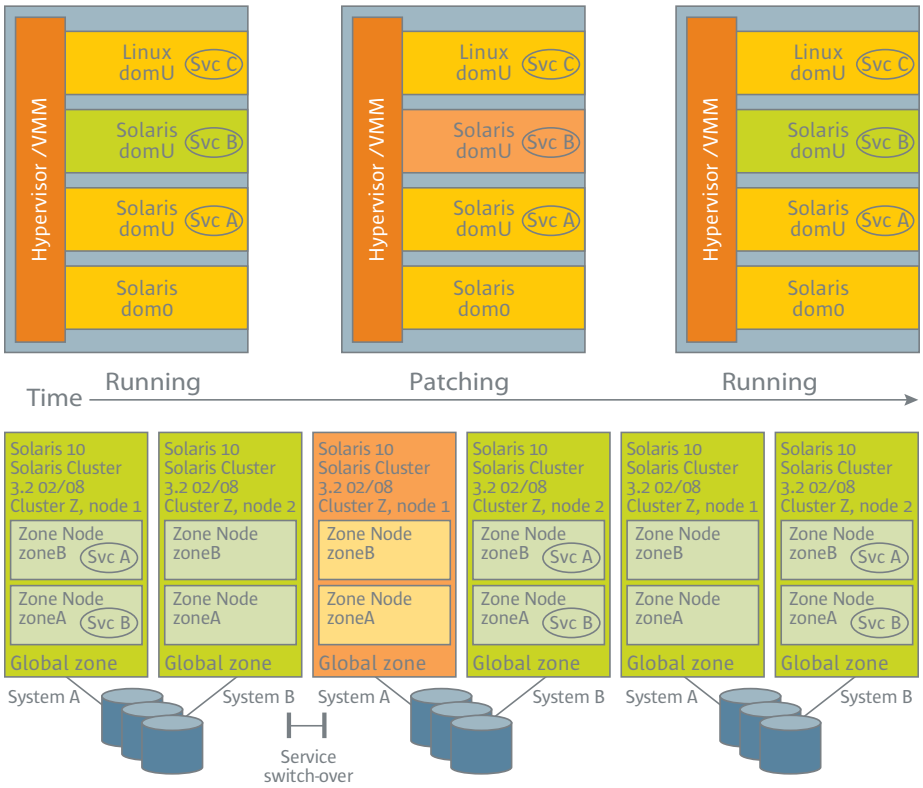


Figure 6. Patching single node virtualization compared to clustered virtualization

Of all of the virtualization technologies covered, Solaris Containers is the only mechanism that actually reduces the number of operating system that need to be managed. All of the other approaches simply collapse down the original number onto fewer physical servers.

Besides consolidation, the other key goal of many virtualization technologies is to optimize resource utilization. A service can be assigned resources from a pool of compute resources to achieve optimal levels of utilization of all the available resources. Here, it is important to highlight that Solaris Cluster can participate in this type of environment. It is possible to dynamically add and remove nodes in Solaris Clusters, allowing resources to be added to, or removed from, the pool. Services hosted on this pool can be changed, on-line, to include the new node(s) as needed. Furthermore, Solaris Cluster allows resource groups and resource to be associated with Solaris projects. Therefore, when the fair share scheduler (FSS) is enabled, the two features can be used together to achieve the desired service level objectives. The corresponding telemetry available from Solaris Cluster can then be viewed via the Solaris Cluster Manager.

Chapter 8

Example Consolidations

This section explores the possible virtualization options for two sample configurations that require high availability.

Multiple Oracle 9*i*, 10*g*, and 11*g* RAC databases

Imagine a number of smaller SPARC clusters running a range of Oracle RAC databases. What consolidation options are available?

The decision process is governed primarily by three factors:

- Available virtualization options
- Available storage
- Whether to have test and development or user acceptance databases on the same cluster as production

The most restrictive of these factors is the virtualization technologies supported by Oracle RAC and Solaris Cluster (Table 5 on page 11). Currently, this configuration is restricted to Dynamic System Domains or, effectively, any global zone on Solaris 10 or Solaris 8/9, i.e., not LDoms or zone nodes (failover containers are irrelevant).

If SAN or supported NAS based storage is available, then RAC clusters of up to 16 nodes can be constructed. Databases can then be consolidated onto this platform using as many nodes as each of the databases require, e.g., sales = nodes 1 through 4, HR = nodes 5 and 6, etc. There is nothing to prevent several database sharing the same nodes, other than resource availability to ensure the configuration performs well.

While different versions of the Oracle software can co-exist on the same set of nodes, as can multiple installation of the same version (using different *ORACLE_HOME* locations), it is advisable to separate 9*i* database from 10*g*/11*g* databases simply for reasons of complexity. Oracle 9*i* does not have an integrated clusterware product, while 10*g* and 11*g* do, so separating the versions maintains the management processes for any one cluster.

Similarly, test and development databases can either be consolidated onto separate virtualized clusters or restricted to nodes not used by the production instances on the cluster. Again, the preferred option is to separate these databases onto separate clusters. As a side note, Oracle 9*i*, 10*g*, and 11*g* RAC can all be used with IP filtering on Solaris Cluster.

Figure 7 shows how multiple RAC databases might be consolidated onto Dynamic System Domain-capable systems. The overall effect is to reduce the number of systems from 24 to 4 and the operating system instance count from 24 to 12, while increasing

the flexibility of power and resource distribution. Furthermore, additional service layers could be consolidated onto the *DSD1* cluster nodes in *cluster A* using zones nodes. This could achieve isolation and resource controls for the service tier (Web, application server, etc.), while further increasing the potential for resource utilization.

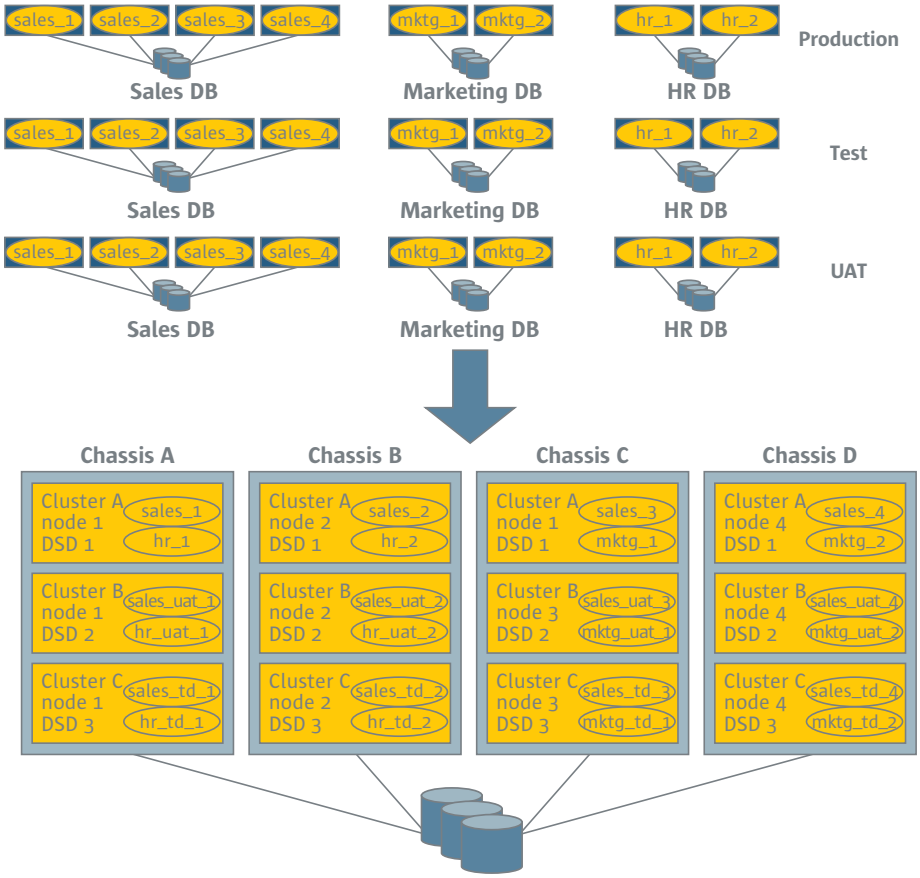


Figure 7. Consolidating Oracle RAC databases onto Dynamic System Domain-capable systems

An engineering prototype has already demonstrated the ability to run Oracle RAC 9*i* in one set of zones, Oracle RAC 10*g* in another set of zones, and Oracle RAC 11*g* in a third set of zones on a cluster. This configuration could reduce the number of operating systems to one per machine, while at the same time supporting simultaneously multiple databases.

Thus, the number of options available could change substantially should support become available for Oracle RAC in LDoms (guest or I/O LDoms) or zone nodes.

Multitiered services

Figure 8 depicts how logical domains and Solaris Containers could be used to support one or more Multitiered services consisting of a mixture of databases, application servers, and Web servers, together with a number of other non-business-critical services. Solaris Cluster is used to provide the highly available infrastructure to those services that need or can benefit from it.

The approach uses guest domains to host the less critical applications that do not require clustering. There is no reason why multiple, separate guest domains cannot be configured per Sun Fire T2000 server, or a combination of guest domains and guest domains with Solaris Containers.

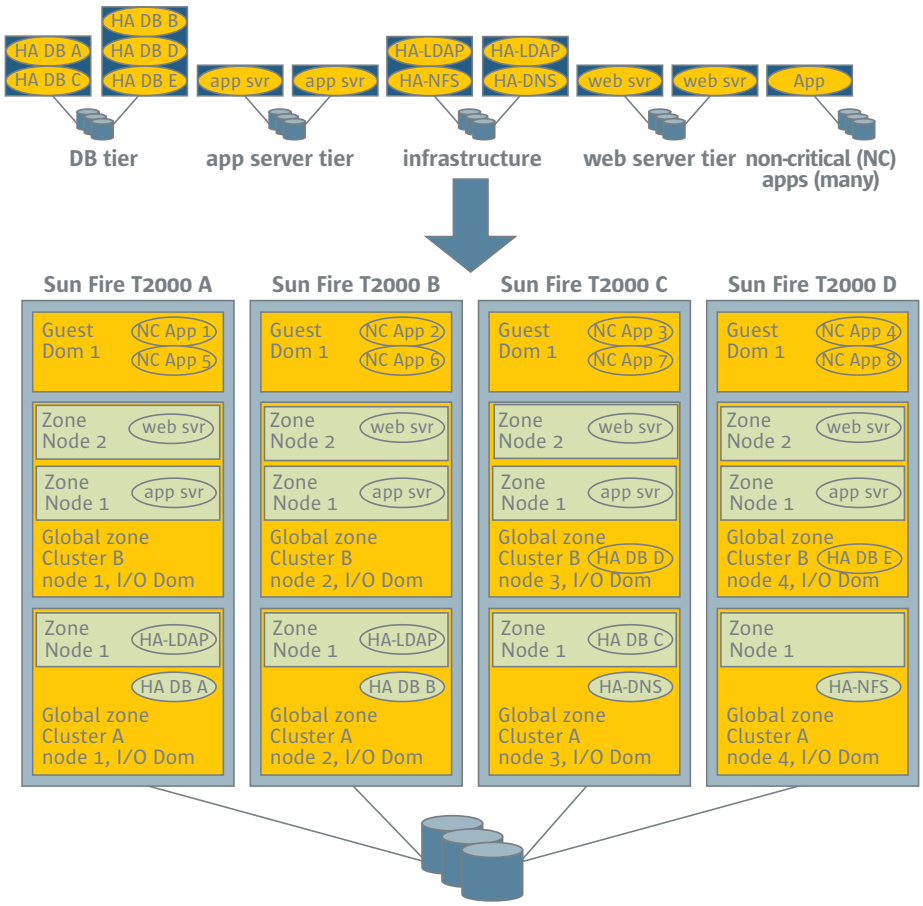


Figure 8. Consolidating multi-tier services onto a system using logical domains, Solaris Containers, and Solaris Cluster 3.2

Services that require clustering are restricted to using I/O domains (see Table 3 on page 9) as cluster nodes because guest domains cannot currently be clustered. Each I/O domain is clustered with its peer I/O domains in the other chassis to form two four-node clusters. The primary motivation is that if any middle tier services require a

highly available NFS service, the services must be hosted on a separate cluster since a cluster cannot be a client of its own HA-NFS service. Furthermore, an NFS server can only run in the global zone. Both of these constraints are imposed by Solaris rather than Solaris Cluster.

The databases in the example are distributed between the global zones and zone nodes of the two clusters. The databases in the global zones, together with the NFS services, could be replicated using one of the replication technologies supported by Solaris Cluster Geographic Edition (see Table 4 on page 10).

The Web and application server tiers both benefit from the isolation of zone node containers. The Web servers can be deployed as a scalable service, allowing external users to connection to one logical address and yet have any of the Web servers service the request. Currently, *ipFilters* can not be used with this configuration (where scalable services are used).

The directory servers can be configured in zones and used as the basis of a multimaster configuration.

The same basic approach can be used to deploy an SAP system, which also involves multitiered applications.

Chapter 9
Summary

Virtualization technologies alone do not address the need of services requiring high availability. With the addition of the Solaris Cluster software and data service agents, services deployed in virtualized environments can also benefit from the fault monitoring and failover capabilities.

Solaris Cluster also allows high service levels to be achieved by providing alternate *containers* (Solaris instance, zone node, etc.) for services, while their original container is patched or upgraded. This is something that cannot be achieved where the service resides in a single virtualized *black box*.

While Solaris Cluster currently imposes some restrictions on the technology combinations supported, e.g., lack of support for LDom or VMware guest domains, Sun is working to investigate and addressed there restrictions with more testing and performance characterizations. Thorough performance and quality testing before supporting a technology helps provide a greater level of assurance that a particular combination functions correctly and enables the services deployed on it to achieve the expected service levels.

About the author

Tim Read has worked in the UK computer industry since 1985. He joined Sun in 1990, in a pre-sales role, as a Systems Engineer. He was responsible for the first Sun Cluster HA 1.0 deployments in UK and now works as Staff Engineer for Sun's Solaris Cluster Engineering group. Tim has authored a number BluePrints and white papers including co-authoring “Designing Enterprise Solutions with Sun Cluster 3.0",“Architecting Availability And Disaster Recovery Solutions”, and “Sun Cluster 3.2 Software: Making Oracle Database 10G RAC Even More Unbreakable”. Tim holds a B.Sc. in Physics, with Astrophysics from Birmingham University in the UK.

Acknowledgements

This Sun Blueprint spans the intersections of many Sun technologies. The author is grateful for the support he received while writing this document from a large number of colleagues including (in no particular order): Terry Smith, Ashutosh Tripathi, Alexandre Chartre, Neil Garthwaite, Detlef Ulherr, Thorsten Frueauf, Honsing Cheng, Steve McKinty, Binu Jose Philip, Michael Haines, Larry Kriewald, Andy Hisgen, Gia-Khanh Nguyen, Jonathan Edwards, Venkateswarlu Tella, Duncan Hardie, Jon Aimone, Oren Teich, Eve Kleinknecht, Kumar Loganathan, Richard Elling, Burt Clouse, Ellard Roush, Hartmut Streppel, Chris Dekanski, Charles Debardeleben, Joost Pronk, Liam Merwick, and Jeff Victor. He apologizes to anyone he missed.

References

Table 8 lists documents referenced in the paper.

Table 8. References.

Reference Number	Description	URL
1	Sun Enterprise 10000 Server: Dynamic System Domains	http://www.sun.com/servers/white-papers/domains.html
2	Clustering Solaris Guests That Run on VMware with Sun Cluster 3.2 Software	http://blogs.sun.com/SC/entry/clustering_solaris_guests_that_run
3	Sun Cluster 3.2 Software: Making Oracle Database 10G R2 RAC Even More Unbreakable	http://www.sun.com/software/whitepapers/solaris10/solaris_cluster.pdf
4	Beginners Guide to LDoms: Understanding and Deploying Logical Domains for Logical Domains 1.0 Release	http://www.sun.com/blueprints/0207/820-0832.html
5	Announcing Solaris Cluster support in LDoms I/O domains	http://wikis.sun.com/display/SunCluster/Read+All+About+It%21+-+Sun+Cluster+Announcements
6	The Sun BluePrints Guide to Solaris Containers: Virtualization in the Solaris Operating System	http://www.sun.com/blueprints/1006/820-0001.pdf
7	OpenSolaris Project: Cluster Agent: OpenSolaris xVM	http://opensolaris.org/os/project/ha-xvm/
8	Architecting Availability and Disaster Recovery Solutions	http://www.sun.com/blueprints/0406/819-5783.pdf
9	Zones and Containers FAQ	http://www.opensolaris.org/os/community/zones/faq/
10	VMware Infrastructure 3 Documentation	http://www.sun.com/software/vmware/support.jsp
11	Solaris Operating System Hardware Virtualization Product Architecture	http://www.sun.com/blueprints/1107/820-3703.pdf
12	Sun xVM Chalk Talk in San Francisco	http://www.sun.com/software/products/xvm/media/talk_vid.xml
13	Sun Cluster 3.2 Data Services Collection for Solaris OS	http://docs.sun.com/app/docs/coll/1574.2?l=en
14	VMware HA: Concepts and Best Practices	http://www.vmware.com/files/pdf/VMwareHA_twp.pdf
15	Resource Group Properties	http://docs.sun.com/app/docs/doc/817-6564/6mlunk5ci?l=en&a=view&q=RG_Affinities
16	Sun Cluster Concepts Guide for Solaris OS	http://docs.sun.com/app/docs/doc/820-2554/x-17efa?l=en&a=view

Ordering Sun Documents

The SunDocs℠ 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/`

Appendix A
Feature Compatibility

The table below provides an indication of the compatibility of certain combinations of features.

Table 9. Feature compatibility

Feature	Dynamic System Domains	Logical Domains	Failover Solaris Containers	Solaris Zone nodes	Solaris Branded containers
Sun Cluster 3.1 8/05: Solaris 8 support	Yes, but not on all DSD capable systems	No	No, N/A	No, N/A	No, N/A
Solaris Cluster 3.1 8/05 and above: Solaris 9 support	Yes, but not on all DSD capable systems	No, N/A	No	No, N/A	No, N/A
Sun Cluster 3.1 8/05 and above: Solaris 10 support	Yes	No	Yes	No	Yes
Sun Cluster 3.2 and above: Solaris 10 support	Yes	Yes, I/O domains only	Yes	Yes	Yes, Solaris 8 or Linux (LX)
Support for dynamic reconfiguration of CPU	Yes, but not boards with kernel cage memory on	Yes, for virtual CPUs in domains running Solaris 10	Capability inherited from system hosting the global zone and resource mapping	Capability inherited from system hosting the global zone and resource mapping	Capability inherited from system hosting the global zone and resource mapping
Support for dynamic reconfiguration of memory	Yes, but not boards with kernel cage memory on	Not supported. Reboot required	Capability inherited from system hosting the global zone and resource mapping	Capability inherited from system hosting the global zone and resource mapping	Capability inherited from system hosting the global zone and resource mapping
Support for dynamic reconfiguration of I/O	Yes	Not supported, reboot required	Capability inherited from system hosting the global zone and resource mapping	Capability inherited from system hosting the global zone and resource mapping	Capability inherited from system hosting the global zone and resource mapping

Feature	Dynamic System Domains	Logical Domains	Failover Solaris Containers	Solaris Zone nodes	Solaris Branded containers
Solaris Cluster 3.2 and above with Solaris 10: ipFilter support	Yes, for failover services on Solaris Cluster 3.2 and above	Yes, for failover services on Solaris Cluster 3.2 and above	Yes, for failover services on Solaris Cluster 3.2 and above	Yes, for failover services on Solaris Cluster 3.2 and above	Yes, for failover services on Solaris Cluster 3.2 and above
Solaris Cluster 3.2 and above with Solaris 10: exclusive-IP mode	N/A	N/A	Yes	No	Yes
Oracle RAC 9i, 10g, 11g	Yes	Pending joint Sun/ Oracle certification	No, N/A	Pending joint Sun/ Oracle certification	No, N/A
Failure mode	Failure of the DSD or an OS panic results in that OS and any other virtualization it supports becoming unavailable	Failure of an LDom results in it being unavailable. Any dependent guest domains might become unavailable if they depend solely on it for root disks and networking	Failure of a (HA) container only impacts that container. No other container is directly impacted by the failure	Failure of a zone node only impacts that zone. No other zone is directly impacted by the failure	Failure of a branded container only impacts that container. No other container is directly impacted by the failure

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

Printed in USA 04/08

USING SOLARIS CLUSTER AND SUN CLUSTER GEOGRAPHIC EDITION WITH VIRTUALIZATION TECHNOLOGIES
April 2008

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together