ORACLE
VM

Oracle VM Server for SPARC Best Practices

ORACLE WHITE PAPER | APRIL 2018

ORACLE

Table of Contents

.

Introduction

This paper discusses best practices for Oracle VM Server for SPARC (previously called Sun Logical Domains). Oracle VM Server for SPARC virtualization technology allows creation of multiple virtual systems on a single physical system. Deployed on Oracle SPARC servers, this solution provides a powerful, efficient virtualization platform. Readers seeking a high level view of Oracle's SPARC virtualization technologies can consult the whitepaper "Consolidation Using Oracle's SPARC Virtualization Technologies" at http://www.oracle.com/technetwork/server-storage/sun-sparc-enterprise/technologies/consolidate-sparc-virtualization-2301718.pdf. That document provides a methodology for evaluating the features of SPARC virtualization choices including SPARC Physical Domains (PDoms), Oracle VM Server for SPARC, and Oracle Solaris Zones. These are complementary technology solutions that can be used independently or in combination.

SPARC Physical Domains (PDoms), Oracle VM Server for SPARC, and Oracle Solaris Zones can be used independently or in combination to increase isolation, add redundancy, or increase virtualization density.

This paper assumes that Oracle VM Server for SPARC is being used, and describes best practices for production environments with intense resource, performance, and availability requirements running on high-end SPARC servers.

This paper addresses the following Oracle VM Server for SPARC topics:

» **Overview.** Fundamental VM Server for SPARC definitions, concepts and deployment options.
» **Requirements.** Software, hardware, and firmware requirements.
» **Best Practices** for optimal performance.
» **Best Practices** for resiliency and availability.

Oracle VM Server for SPARC Overview

Oracle VM Server for SPARC is a member of the Oracle VM product family that provides highly efficient, enterprise-class virtualization capabilities for supported SPARC servers. Oracle's SPARC servers, with up to 4,096 CPU threads, are powerful systems that can virtualize multiple physical servers onto a single platform.

Oracle VM Server for SPARC leverages the built-in SPARC hypervisor to create virtual systems. Each virtual system is called a *logical domain* or alternatively, a *virtual machine*, and runs its own independent copy of the Oracle Solaris operating system with its own CPU, memory, and I/O resources.

Oracle VM Server for SPARC takes advantage of the massive thread scale offered by SPARC T-Series and SPARC M-Series servers for high-performance virtualization. SPARC M-Series servers can optionally be divided into physical domains ("PDoms"), each of which can host independent Oracle VM Server for SPARC environments. Up to 128 logical domains can be created on a single physical server or M-series physical domain.

Oracle VM Server for SPARC Requirements

Oracle VM Server for SPARC software is based on the Logical Domains Manager, SPARC server firmware, and Oracle Solaris versions. At this writing, the current Logical Domains Manager version is 3.5, which is delivered with Solaris 11.3 and provides features for enhanced performance, availability, function, and manageability.

Best Practice: Use Solaris 11.3 or later, especially in control and service domains, to get the most recent performance and functional improvements. Starting with Oracle VM Server for SPARC 3.3, Solaris 10 cannot be used for the control domain. Solaris 11 service domains should be used even if guest domains run Solaris 10. You can "mix and match" Solaris OS levels in the different domains. That said, the most recent versions of Solaris are recommended for the best performance and feature availability. Solaris 11.3 or later is required for control, service, and I/O domains on SPARC M7, T7, S7 and later systems. Solaris 10 1/13 plus required patches can be used as guest domains on those servers.

Best Practice: Keep firmware levels up to date, based on the platform and the version of Oracle VM Server for SPARC. The Oracle VM Server for SPARC 3.5 Installation Guide shows firmware versions for each supported platform at https://docs.oracle.com/cd/E80106_01/html/E80108/ldomsrequiredsoftwarepatches.html#scrolltoc

Oracle Solaris 10 domains and Oracle Solaris 11 domains can run on the same server. It is recommended to use Solaris 11.3 or later for the control domain and all I/O and service domains even if guest domains run Solaris 10. Starting with Oracle VM Server for SPARC 3.3, the control domain cannot be Solaris 10. Applications certified for Solaris 10 can run in Solaris 10 branded zones in Solaris 11 domains. This leverages the Solaris 11 kernel that has been optimized for the latest servers while still appearing to be Solaris 10.

Oracle VM Server for SPARC Domain Roles

Logical domains have one or more roles that depend on how they are configured, which resources they own and what services they provide:

» **Control Domain.** There is one control domain per server or physical domain, and it is the first domain to boot up when a server is powered up. The control domain has a privileged connection with the hypervisor which allows it to manage resources and administer domains. The Logical Domains Manager runs in this domain and uses this hypervisor connection to create and manage other logical domains. The control domain is a root complex domain and usually a service domain. The control domain is named `primary`.

» **Service Domain.** A service domain provides virtual device services to other domains, such as a virtual switch, a virtual console concentrator, and a virtual disk server. You can have more than one service domain, and any domain can be configured as a service domain. Service domains are typically root domains.

» **I/O Domain**. An I/O domain has direct access to a physical I/O device, such as a network card in a PCI Express (PCIe) controller. An I/O domain can own the following:

  » One or more PCIe root complexes, in which case the domain is also called a *root domain.*

  » PCIe slots or on-board PCIe devices by using the direct I/O (DIO) feature.

  » Single-root I/O virtualization (SR-IOV) virtual functions.

  The maximum number of I/O domains on a single server is determined by the server-specific number of PCIe buses, PCIe devices, and SR-IOV virtual functions.

  A service domain is almost always an I/O domain, but the opposite is not necessarily true. An I/O domain may have I/O devices for its own applications rather than to provide virtual I/O for other domains. An I/O domain that is also a service domain shares physical I/O devices with other domains in the form of virtual devices. An

I/O domain can use physical I/O devices to eliminate dependency on service domains and to ensure optimal I/O performance for its own applications.

Running the Oracle Solaris operating system in an I/O domain is very similar to running Oracle Solaris on a non-virtualized system. The operating system running in an I/O domain uses regular (non-virtualized) device drivers to access physical I/O devices. An I/O domain may also use virtual I/O devices.

» **Root Domain.** A root domain is an I/O domain that has a PCIe bus (also known as a PCIe root complex) assigned to it, with direct access to all physical I/O devices on its buses. The maximum number of root domains that can be created on a single server is determined by the number of PCIe buses on the server platform. For example, a maximum of four root domains can be created on the SPARC T4-4 server, and a maximum of 16 on a T5-8 server. The control domain is a root domain.

» **Guest Domain.** A guest domain exclusively uses virtual devices provided by service domains, and does not have physical I/O devices. Applications run in guest domains or in I/O domains depending on whether you want to use virtual I/O or physical I/O.

Best Practices for Domain Configuration

This section describes configuration guidelines and best practices recommended when setting up Oracle VM Server for SPARC, emphasizing the control domain and service domains.

» Service domains consume CPU, memory and I/O to provide virtual I/O services, and must be allocated sufficient resources to effectively drive I/O. This applies to the control domain when it is used as a service domain. The importance of providing sufficient resources to service domains cannot be overstated.

  » A good starting point for a service domain is a minimum of 2 CPU cores (`ldm set-core 2 primary`) and 16 GB of memory (`ldm set-mem 16g primary`).

  » Large environments require additional CPU and memory resources, especially for I/O intensive workloads. An exception can be made when applications run in I/O domains and use physical I/O instead of virtual I/O. In that case smaller service domain allocations may be sufficient.

  » Small test environments can have smaller allocations, but never less than 1 core and 4GB of RAM.

  » Additional RAM may be required if ZFS or NFS are being used for virtual disk backends, as they may have increased memory requirements for buffering.

  » A 10GbE network device can consume an entire CPU core, so add a core to drive that effectively.

  » There is no fixed sizing because workloads differ in I/O intensity or use physical I/O instead of virtual I/O.

  » Measure CPU and memory consumption in service domains to ensure that they are allocated appropriate resources. Use `ldm list` for a quick view of CPU utilization, along with standard Solaris tools like `pgstat`, `vmstat` and `mpstat`. `pgstat` is recommended since it provides accurate core and pipeline utilization. Check for high memory consumption by looking at `vmstat -p` output: a non-zero scan rate (“`sr`”), low free storage, or swapping indicates that more memory is needed. Occasional checks may not detect peak load periods, so monitor proactively.

  » Service domains, including the control domain, can be dynamically and non-disruptively reconfigured to change resource allocations to meet requirements.

  » CPU resources should be allocated on a core boundary to minimize false cache sharing in which multiple domains are allocated the same CPU core.

Service domains use CPU and memory resources to provide virtual I/O and must be sized accordingly. A good starting point is a minimum of 2 CPU cores and 16GB of RAM, but actual resource consumption must be measured.

» Configure multiple service domains to eliminate single points of failure. Guest domains can be provisioned with redundant paths to virtual network and disk devices.

- Reboot or failure of a service domain does not bring down or reset guest domains using that service domain. Instead, if a guest domain's I/O is provided by a single service domain, I/O operations are suspended while the service domain is down, and automatically resume once the service domain is running again. Some applications may time out if they are unable to complete I/O on a timely basis, so this should not be considered a resilient configuration.
- If a redundant service domain has been configured to provide multi-path disk and network access, guest virtual I/O continues using the other service domain without interruption of service. Redundant service domains are a best practice on servers that support them, and are illustrated later in this document.
- **Note:** some server models, such as S7-2 or T8-1, are designed for scale-out deployments and don't have multiple buses and built-in device controllers to cost-effectively create multiple root domains for redundant service domains. Instead, availability and scale are provided by multiple servers sharing the workloads. The best architecture with these servers is to use resilient applications like Oracle Real Application Clusters, and use live migration for planned outages. Consult the document library for each server model.
- Service domains must not use virtual devices provided by other service domains, as that introduces a dependency that compromises availability.
- Service domains should use resilient I/O backends for the virtual devices they export.
- To configure virtual disk redundancy:
  - Define virtual disks as members of `mpgroup` pairs using `ldm add-vdsdev` commands pointing to the same storage device backend accessed by different service domains' virtual disk services. A single `ldm add-vdisk` command names one of the `vdsdev` definitions in the `mpgroup`. This forms an active-passive pair that provides access even if a service domain fails, reboots or loses access to the disk device. The device appears to the guest as a single redundant virtual disk, so no configuration is required within the guest. **Note:** if using NFS for virtual disk backends, use "soft" NFS mounts to ensure that a backend failure is propagated to the service domain, which will determine if the entire backend is unavailable. Starting with Oracle VM Server for SPARC 3.2, the administrator can use the `ldm set-vdisk` command to set the active backend. This can be used to improve performance by balancing load over service domains and their I/O paths. **Note**: mpgroups cannot be used for guest applications that require SCSI reservation. For further details, see https://docs.oracle.com/cd/E80106_01/html/E80109/configuringvdiskmultipathing.html
  - Use Solaris I/O Multipathing (MPxIO) within each service domain to protect against I/O path errors and to permit hotplug for live service. This does not provide resiliency for loss of the service domain, so is not a replacement for using an `mpgroup`. The two methods can be used together.
  - If using virtual HBA (described at https://docs.oracle.com/cd/E80106_01/html/E80109/usingvhbas.html), use MPxIO in the guest domain across virtual HBAs provided by multiple service domains. This supports active-active path management and SCSI reservation.
- To configure virtual network redundancy
  - Use IP multipathing (IPMP) within guest domains. Two or more virtual network devices are defined for the guest using virtual network devices provided by more than one service domain. The guest can use link-based or probe-based IPMP to provide resiliency across its virtual network devices. Guests running Solaris 11 can optionally use transitive probing. The IPMP group continues to work if a network device or service domain fails.
  - Use IEEE 802.3ad link aggregations in service domains. A virtual switch can be a link aggregation for availability and load spreading using the instructions in "Creating a Link Aggregation" at https://docs.oracle.com/cd/E53394_01/html/E54788/gmsaa.html. This does not provide resiliency for loss of the service domain, so it should be combined with IPMP. The two methods can be used together.
- Each service domain should be a root domain. PCI root complexes should be assigned to alternate service domains to provide disk and network resources they need to boot from and to serve out for virtual devices. The buses and devices to use depend on the server model. An example in this paper will illustrate how to determine which buses to assign to a service domain.

Multiple service domains can eliminate single points of failure by providing guest domains with redundant I/O for resiliency. Redundant disk is configured using `mpgroups`*, with MPxIO in service domains, or by using virtual HBA with MPxIO in guest domains . Network redundancy is configured by using virtual switches based on link aggregations, with* `IPMP` *in guest domains.*

» Access to the control domain and other service domains should be carefully controlled to prevent accidental or deliberate interference with domain operations. It is not generally recommended to run applications in the control domain or service domains. An exception can be made for system monitoring and management software, but the security and availability implications of the added software must be fully understood.

Best Practices for CPU and Memory

This section describes CPU and memory configuration guidelines and best practices for all domains. Oracle VM Server for SPARC uses a simple and efficient model for allocating CPU resources to domains. CPU threads or entire cores are allocated directly to each domain, and are not shared and time-sliced as with traditional software-based hypervisors. Applications will see near-native performance.

Oracle VM Server for SPARC eliminates overhead exhibited by software-based time-slicing at the hypervisor level. CPUs can be dynamically and non-disruptively added to or removed from a domain as needed for application requirements.

Operating systems use privileged instructions that change CPU state, like memory mapping and interrupt levels. These instructions must be suppressed in a virtual machine or they would alter actual system state and compromise the shared server environment. Traditionally, hypervisors generate program traps or dynamically replace such instructions, causing a branch into the hypervisor where the native CPU instructions are emulated in software with the context of that virtual machine. This process can occur many thousands of times per second, requiring multiple context switches each time. This process has been improved in recent hardware generations, for example, with VT-x on Intel processors, but there is still overhead for virtual CPUs on physical ones. Oracle VM Server for SPARC eliminates this expensive process since each domain has its own CPUs and can freely change its own CPU state without interfering with other VMs. The entire category of overhead is eliminated. Another benefit is that domains are not affected by other domains CPU activity: the “noisy neighbor” problem in which one domain’s CPU activity reduces CPU capacity available to others is entirely avoided, making CPU performance in a domain consistent and repeatable regardless of what other domains are doing.

Oracle VM Server for SPARC also takes a simple and efficient approach to memory allocation. Memory is not oversubscribed, as it is in some other virtualization technologies. It appears attractive to oversubscribe memory for virtual machines and page or swap "least recently used" (LRU) pages to disk to handle memory demands, but in practice it is problematic with virtual machines. Virtual machines have poor locality of reference since they tend to "touch" all of their memory locations. As a result their working set sizes approach the virtual machine memory size, and performance suffers if all memory pages are not resident. In effect, they typically take up the same amount of real memory anyway.

Another consideration is that modern operating systems use virtual memory, and the combination of a virtual memory OS under a virtual memory hypervisor has been observed to lead to poor selection of pages to swap to disk, leading to “double paging”. VMs in an oversubscribed memory environment risk unpredictable performance due to "thrashing" of virtual memory contents to and from disk. Most VM systems that oversubscribe are deliberately run with low levels of oversubscription to avoid this risk, except for intermittent workloads. There are additional complications mapping guest VM application memory addresses ("guest virtual") to guest VM real addresses ("guest real") which is virtual memory in the hypervisor ("host virtual") and finally to physical memory addresses ("host real"). This requires creation and maintenance of “shadow page tables”, which consumes CPU cycles and memory.

Oracle VM Server for SPARC eliminates this complexity and performance risk by using a simple rule: VMs are given their own dedicated memory, which is not swapped to and from disk. Without virtual memory there is no risk of thrashing, and no need to expend memory for the "shadow page tables" used in other environments. Guest VM memory allocation can be dynamically increased or decreased as needed for application requirements.

Solaris Zones are a very effective way to increase VM density. The Solaris OS can make informed CPU scheduling and memory management decisions for zones under a single resource manager. Solaris 10 branded zones in Solaris 11 domains can host applications certified for Solaris 10 while running on a Solaris 11 kernel optimized for the newest servers. Solaris Zones and Oracle VM Server for SPARC are complementary technologies. Used together, they provide a powerful virtualization platform.

The CPU and memory allocation model for Oracle VM Server for SPARC simplifies administrative effort, but several best practices still apply:.

» Oracle’s SPARC servers supporting logical domains have CPU cores with 8 CPU threads. CPU resources should be allocated on a core boundary for performance reasons. This prevents "split core" situations in which CPU cores are shared by more than one domain (different domains with CPU threads on the same core). That reduces performance by causing "false cache sharing" in which domains compete for a core's cache. The performance impact of split-core situations varies, depending on the domains’ activities
  » The best method is to use whole-core allocation, for example: `ldm set-core 8 mydomain`. An alternative is to allocate virtual CPUs in increments of the number of threads per core.
  » Oracle VM Server for SPARC attempts to avoid split core situations by assigning domains CPUs from unused cores, even when whole-core allocation is not specified.
  » If fine-grain CPU granularity is needed for multiple applications, especially when applications have intermittent CPU needs, it is best to deploy them in zones within a logical domain for finer resource control.
  » When domains have very light CPU requirements, there is no harm in giving them sub-core allocations even if that causes a split-core configuration, since idle workloads are not harmed by added overhead.
» Starting with the T4 processor, SPARC servers can use a critical threading mode that delivers the highest single thread performance by dedicating all of a core's pipeline and cache resource to a software thread. This requires whole-core allocation, described above, and sufficient CPU cores that Solaris can give critical threads their own cores. Depending on the application, this can be several times faster than in the normal "throughput mode".
  » Solaris generally detects threads that benefit from this mode and automatically "does the right thing" without needing administrative effort, whether in a domain or not. To explicitly set this for an application, set its scheduling class to FX with a priority of 60 or more: “`priocntl -s -c FX -m 60 -p 60 -i $PID`". Several Oracle applications, like Oracle Database 11gR2, automatically leverage this capability to get performance benefits not available on other platforms, as described in the section "Optimization #2: Critical Threads" in How Oracle Solaris Makes Oracle Database Fast. http://www.oracle.com/technetwork/articles/servers-storage-admin/sol-why-os-matters-1961737.html
  » This CPU mode formerly could be set at the domain level by issuing the command `ldm set-domain threading=max-ipc <domainname>`. This is not recommended, was removed in Oracle VM Server for SPARC 3.3, and is only mentioned for completeness sake.
» NUMA latency - Servers with more than one CPU socket, such as a SPARC T8-4, have non-uniform memory access (NUMA) latency between CPUs and RAM. "Local" memory access from CPUs on the same socket has lower latency than "remote". This can affect applications with large memory footprints that do not fit in cache, or

are otherwise sensitive to memory latency. This does not apply to single board servers, like a SPARC T8-1, where all memory accesses are local.

- The logical domains manager attempts to bind domains to CPU cores and RAM locations on the same CPU socket, making all memory references local. If this is not possible because of the domain's size or prior core assignments, the domain manager tries to distribute CPU and RAM equally across sockets to prevent an unbalanced configuration. This optimization is automatically done at domain bind time.

Best Practices for I/O

Oracle VM Server for SPARC offers alternative ways to configure I/O, with a wide range of results for performance, convenience, flexibility and resiliency. The first decision is whether to use a guest domain based on virtual I/O, or an I/O domain using physical I/O.

Physical I/O

Physical I/O offers native performance and native I/O behavior - devices and device drivers behave the same in non-virtualized environments. However, it does not provide the flexibility and device sharing capabilities of virtual I/O. Other considerations:

- Root complex domains have no restriction on the types of device that can be used.
- Solaris uses its native device drivers for physical devices, with the same configuration and features as in a non-domain environment.
- The number of root complex domains is limited by the number of buses available on a server platform.
- SR-IOV can be used for Ethernet, InfiniBand, and FibreChannel devices. SR-IOV and Direct I/O (DIO) require specific qualifications of the cards used.
- The number of SR-IOV virtual functions and Direct I/O devices is determined by the type and number of qualified cards supporting those functions.
- SR-IOV is preferred to Direct I/O, and provides better resource granularity.
- Live migration is not available for domains using physical I/O, except for domains using SR-IOV for Ethernet devices. This feature became available with Oracle VM Server for SPARC 3.5.
- Direct I/O is not supported on S7, M7, T7 and later systems. On those platforms use root domains, SR-IOV or virtual I/O.
- Starting with Oracle VM Server for SPARC 3.2 and Solaris 11.2 SRU 8, I/O domains using SR-IOV can be configured to avoid single points of failure. Under previous versions, domains using SR-IOV or DIO are subject to a dependency on the domain owning the bus used for the SR-IOV virtual function or DIO card, and may take an outage if the bus-owning domain panics or resets the PCI bus. Now, domains can use SR-IOV virtual functions from different root domains and use IPMP or MPXIO for availability. Resilient I/O domains are described in the Administration Guide section at https://docs.oracle.com/cd/E80106_01/html/E80109/iodomainresiliency.html#scrolltoc.
- SR-IOV is no longer supported for Solaris 10 domains. See SR-IOV requirements at https://docs.oracle.com/cd/E80106_01/html/E80109/sriovhwswreqts.html

Virtual I/O

Virtual I/O offers more flexibility than physical I/O, but does not offer the native performance of physical I/O. However, it is now possible to provide near-native I/O performance for appropriately configured service domains and virtual devices.

- Virtual I/O is restricted to virtual HBA, disk, network, and console devices. Device classes like tape are supported by virtual HBA or provided by a qualified physical I/O device.

» Guest domains using virtual devices can be live migrated between compatible SPARC servers.
» Virtual I/O allows sharing the same physical device as backend for multiple virtual devices and multiple domains.
» Virtual devices consume "Logical Device Channel" (LDC) endpoints in the guest domain and in the service domain that provides the device. A maximum number of LDCs can be associated with a domain, setting an upper bound on the number of virtual devices a domain can have or serve out. The per-domain limit is 1,984 on SPARC T4, T5, T7, T8, S7, M5, M6, M7 and M8 servers from a pool of 98,304 LDCs. The pool is per server on S7, T4, T5 and T7, and per physical domain on M5, M6, M7 and M8. System software and firmware requirements and other details are at https://docs.oracle.com/cd/E80106_01/html/E80109/usingldcs.html#scrolltoc UltraSPARC T2 systems are limited to 512 LDCs per domain. UltraSPARC T2+ and SPARC T3 systems have a limit of 768 LDCs per domain, as do later machines not running the required firmware.
» Virtual I/O requires appropriately sized service domains and defining virtual disk and network services.

There is no single best practice for all situations. If more emphasis is placed on flexibility and device sharing, then virtual I/O is the better choice. That is the typical deployment model, and widely used for workloads consolidated onto domains on shared servers. Recent versions of Oracle VM Server for SPARC permit virtual I/O with near native performance, maximum flexibility, and ability to easily share resources.

If maximum performance and independence from other domains is required, especially for an application that does not require live migration or dynamic reconfiguration, then the best choice may be to deploy it in a root domain. SR-IOV offerd native performance and better resource granularity than root complex domains, but require specific devices and make the I/O domain dependent on the domain owning the physical devices' bus.

Disk Device Best Practices

Physical disk I/O provisioned to a root domain or via a qualified SR-IOV FibreChannel virtual function should have performance characteristics equivalent to a non-virtualized environment. The following practices make it possible to achieve performance and flexibility for virtual disks.

» For best performance, use a whole disk backend: a SAN LUN or full disk. This is the simplest and highest performing virtual disk I/O option. The disk should be presented to multiple hosts if the domain needs to be moved between servers.
» Spread load across multiple disks to reduce latency and queuing, just as in a non-virtual environment.
» Solaris and logical domains software updates often include performance enhancements, so keep software and firmware versions are at current levels to include the latest performance improvements. For example, Solaris 11.1 SRU 19.6 and Solaris 10 patch 150400-13 substantially improved virtual disk performance, as explained in Stefan Hinker's blog “Improved vDisk Performance for LDoms” https://blogs.oracle.com/cmt/entry/improved_vdisk_performance_for_ldoms
» Virtual disks backed by a file in a file system are convenient and easy to set up, but have lower performance. This may be acceptable for test/development or applications with low disk I/O requirements. This disk backend should not be shared between domains or hosts as SCSI reservation is not supported.
» Starting with Oracle VM Server for SPARC 3.3, guest domains can have virtual SCSI host bus adapters (vHBA). This capability is described at https://docs.oracle.com/cd/E80106_01/html/E80109/usingvhbas.html#scrolltoc and permits native Solaris device driver access to LUNs on a SAN. This feature lets guest domains use MPxIO active-active path management, SCSI reservation, and non-disk devices like tape.
» ZFS can be used for disk backends, which provides useful features (clones, snapshots, compression, advanced data integrity) but imposes overhead compared to a raw device, and several restrictions.
    » Local or SAN ZFS disk backends (objects in a ZFS pool mounted to the service domain) cannot be used with live migration, because a zpool can be mounted to only one host at a time. This does not apply to disk backends in ZFS presented to service domains via network using NFS or iSCSI from a system that uses ZFS as a local file system, such as the Oracle ZFS Storage ZS5.

    - Always use ZFS compression. The default algorithm (`zfs set compression=on`) uses very little CPU and can actually speed up I/O while reducing space consumption since fewer blocks are transferred. This setting is applied at the ZFS dataset level and issued on the system hosting the ZFS resource.
    - A ZFS volume "zvol" ("`zfs create -V`") can outperform a flat file in a ZFS dataset.
    - Set the ZFS `recordsize` (for files) or `volblocksize` (for ZFS volumes) to match the expected application use. This is the same recommendation as in a non-virtual environment, and avoids read-modify-write cycles that inflate I/O counts and overhead. In particular, the `recordsize` default of 128K is not optimal for small random I/O. These settings are applied on the system hosting the ZFS resource.
    - ZFS requires more memory than other backends. Ensure that the service domain has sufficient memory, and monitor it under load to make sure it isn't low on memory. A minimal service domain of 4GB RAM will not be sufficient.
- Starting with Oracle VM Server for SPARC 3.1.1, you can use SR-IOV for Fibre Channel devices. See the Oracle VM Server for SPARC Release Notes for qualification details. This can provide high performance for I/O on qualified host bus adapters.
- Virtual disks on NFS and iSCSI can perform well on a fast (10GbE) network
    - iSCSI may provide better performance than NFS, but actual results are subject to performance tuning.
    - Apply the same network tuning you would use for in non-virtual applications. For NFS, specify mount options to disable `atime`, use hard mounts (except as described below with redundant disks using an `mpgroup`), and set large read and write sizes. The following values in the service domains' `/etc/system` can improve NFS performance: `set nfs:nfs3_bsize=1048576` or `set nfs:nfs4_bsize=1048576` (increasing I/O size in NFS version 3 or version 4 clients) and `set rpcmod:clnt_max_conns=8` (to increase concurrency).
    - If NFS or iSCSI backends are provided by ZFS, whether with standard Solaris ZFS or the ZFS Storage Appliance, make sure the server has sufficient RAM for caching, and install write-optimized solid-state disk (SSD) ZFS Intent Logs (ZIL) to speed up synchronous writes.
- **For completely native performance for all devices, use a PCIe root complex domain and use physical I/O. This ensures performance without compromise or complexity.**

Network Device Best Practices

Physical network provided by SR-IOV virtual functions, Direct I/O devices, or native I/O in root complex domains are the same as in a non-virtualized environment. Consequently, network device best practices are concentrated on virtual network devices. The most comprehensive documentation on how to configure for virtual network performance is in the MOS note **Best Practices for Oracle Solaris Network Performance with Oracle VM Server for SPARC (Doc ID 1908136.1)**

- Allocate 8GB of RAM and 2 cores to a service domain for each 10GbE of bandwidth it supports. Guest domains should have 4GB of RAM and a full core for each 10GbE of bandwidth in order to provide sufficient resources to drive the CPU fanout for the virtual network devices.
- Current versions of Oracle VM Server for SPARC and Solaris use Large Segment Offload (LSO) and RxDring support to substantially reduce network latency and CPU utilization. This change, along with the above resource allocation, makes it possible to drive 10GbE at essentially wire speed.
    - The domains must run Solaris 11 SRU 9.6 or Solaris 10 with patch 150031-07 or later.
    - Issue `ldm set-domain extended-mapin-space=on mydomain` for each domain. If this was not previously set, the domain (including the control domain) will require a reboot for the change to take effect. This is now the default for new domains, so the change is needed only for domains created before this support was introduced. To see if any domains need to be changed, log into the control domain and `issue ldm list -l|grep extended`, and see if any are "`extended-mapin-space=off`".

    - This option requires an additional 4MB of RAM per guest.
    - Validate that the `dring_mode` is correct by issuing `kstat -p|grep dring_mode` in each domain. The value 4 indicates the proper mode is set.
    - LSO provides performance benefits similar to what jumbo frames provide at the link layer, and it is not necessary or recommended to use jumbo frames. To validate that LSO is turned on, issue the command `kstat -p|grep lso`. The value `tcp:0:tcpstat:tcp_lso_disabled` should be set to 0, and the value `lso_enabled` should be 1.
- Configure IP network multipathing (IPMP) groups in the guest domain using virtual network devices from different service domains. This provides resiliency against loss of a device or service domain without loss of access. Either link-based or probe-based IPMP can be used, based on site preferences just as with non-virtual systems. For link-based IPMP, specify `linkprop=phys-state` when defining the virtual network device to propagate physical link state to the virtual network link.
- Configure link aggregation in the service domain to make use of IEEE 802.3ad features for bandwidth, load balancing and network link failover. Link aggregation is enabled by using an aggregation for a virtual switches backend device rather than a physical network link. Link aggregation is complementary to using IPMP in the guest domain, and the two methods can be combined.
- Use multiple network links, use jumbo frames (if not using LSO), create VLANs, adjust TCP windows and other systems settings the same way and for the same reasons as you would in a non-virtual environment.
- Set the property `inter-vnet-link=off` on the virtual network switch if excessive logical domain channels are being used. That will have a slight increase in the time to transmit packets between domains on the same virtual switch as all guest-to-guest communications traffic will go through the virtual switch rather than directly from one guest domain to another guest domain.
- Use network bandwidth controls if combining multiple domains on the same virtual switch and concerned about VMs consuming an excessive portion of the shared network resource, for example: `ldm set-vnet maxbw=200M net0 mydomain`.
- To use Solaris 11 network virtualization in a guest domain, for example with Solaris zones, you must configure virtual network devices with multiple MAC addresses. This is done by creating virtual network devices with the alt-mac-addr parameter, as in `ldm add-vnet alt-mac-addrs=auto,auto,auto,auto vnet1 primary-vsw0 mydomain1`. See https://docs.oracle.com/cd/E80106_01/html/E80109/vnicsonvnets.html#scrolltoc.

Best Practices for Live Migration

Live Migration is useful for moving guest domains from one server (the source server) to another (the target) while domains continue to run. It can be used to improve operation in a domain environment by making it possible to evacuate a server for maintenance, to load balance virtual machines across a pool of servers, or to free up resources so other domains can be assigned additional CPUs or memory.

There are several requirements for live migration:

- Live migration requires servers running compatible versions of Oracle VM Server for SPARC and firmware. As a best practice, customers should standardize on the same version of Oracle VM Server for SPARC across their servers, and use the same firmware level on matching server types (for example, all the T7 systems should run the same firmware level). An exception is when live migration is used for non-disruptive upgrades to new software and firmware levels: in this case, check the Release Notes for compatible versions.
- Server and target servers must have identical CPU chip and clock frequencies, unless cross-CPU live migration is used. Setting a domain's `cpu-arch` property enables you to migrate between systems with different processor types as described in the Oracle VM Server for SPARC Administration Guide at Domain Migration Requirements for CPUs. The default setting is `native`. Other settings may disable some advanced CPU features such as instruction-based crypto offload, so use `native` if all the servers are the same type. Note that `cpu-arch` can only be changed when the domain is not running, so cross-CPU migration must be planned in advance.

» The target machine must have sufficient CPU and memory to accommodate the resources in use by the domain to be migrated.

» Only guest domains (domains exclusively using virtual devices) can be live migrated. A domain with physical I/O (dedicated PCIe bus, direct I/O card, Infiniband or Fibre Channel SR-IOV) cannot be live migrated. An exception is that domains using Ethernet SR-IOV can be live migrated – see https://docs.oracle.com/cd/E80106_01/html/E80109/migratedomainwithethernetvf.html.

» Guest domains must use networked storage, and each virtual disk backend in the domain to be migrated must be defined on the target machine. This shared storage can be a SAN disk, or storage provided by NFS or iSCSI. The virtual disk back end must have the same volume and service names as on the source machine. Paths to the backend can be different on the source and target machines, but they must refer to the same backend.

» Each virtual network device in the domain to be migrated must have a corresponding virtual network switch on the target machine. Each virtual network switch must have the same name as the virtual network switch to which the device is attached on the source machine.

» Common network accessibility between the source and target servers.

Domains can be migrated without being shutdown once these requirements are met it. Several best practices apply:

» Live migration is CPU intensive in the control domain of the source (sending) host. Additional CPU cores in the control domain can substantially reduce domain migration and suspend times. CPU cores can be added before live migration and removed afterwards.

» On older SPARC servers, such as SPARC T3 and previous models, add a crypto accelerator to the control domain for each CPU core it owns (`ldm set-crypto 2 primary`). This speeds up migration using the SPARC processor's hardware acceleration. Hardware crypto acceleration is always available without an administrative step on T4 and later processors.

» Live migration is also CPU intensive in the domain being migrated, so try to migrate domains during periods of low activity. CPU cost is substantially reduced on the M7, T7 and later servers running Solaris 11.3 or later.

» Guests that heavily modify their memory take more time to migrate since memory contents have to be retransmitted, possibly several times. The overhead of tracking changed pages also increases guest CPU utilization, though this cost is also substantially reduced on M7 and T7 and later servers running Solaris 11.3. In general, live migration takes more time for large guest domains, and is not always appropriate for large memory applications. In those cases, application-level clustering is often more efficient and convenient.

» Live migration should use a high bandwidth network.

» Guest domains that are migrated should use Network Time Protocol (NTP) to ensure consistent clock behavior.

Live migration is not as operationally important in Oracle VM Server for SPARC as in some other virtualization technologies. Some virtualization platforms lack the ability to update system software without an outage, so they require "evacuating" the server via live migration. With Oracle VM Server for SPARC an alternate service domain makes it possible to do "rolling upgrades" in place without having to evacuate a server. For example, you can `pkg update` Solaris in the control domain and service domains at the same time during normal operational hours, and then reboot them one at a time into the new Solaris level. While one service domain reboots, virtual I/O proceeds through the alternate, and you can upgrade Solaris in all the service domains without a loss in application availability. This requires a server configured with multiple root domains, with disk and network resources available for each domain. As noted previously, some SPARC servers like the S7-2 family are designed for horizontal scale (as are many other platforms) and don’t have built-in network and disk resources to permit redundant service domains

Oracle VM Server for SPARC also permits increasing and reducing the CPU, memory, and I/O resources of a domain. Rather than migrate a domain to a less-utilized server to give it more capacity, it can be easier and faster to reduce the CPU or memory allocation of another domain on the same server for the time those resources are needed, and not migrate any domain at all. It’s a best practice to reserve some headroom in order to conveniently meet a sudden peak in demand, As a result, Oracle VM Server for SPARC reduces the number of use cases in which live migration is the best or only answer.

Live migration is not suitable for all applications. In particular, applications that use real- time network heartbeats to determine application state may not behave correctly during migration due to variable latency and the suspension period at the end of migration. It’s also possible that network packets may be dropped during migration. Live migration should be considered in context of application requirements.

Application frameworks like Oracle Real Applications provide multi-node applications that often do not need to be migrated at all. It can be simpler and faster to shutdown a node in the cluster and optionally start another one on a different host. Live migration should not be considered a substitute for high availability designs, as one cannot live migrate a domain from a failed system.

Best Practices for Availability

Availability is a broad topic, and this section will focus on aspects specific to Oracle VM Server for SPARC. The goal is to improve Reliability, Availability and Serviceability (RAS) for guests and applications running in domains.

Let's state some guiding principles for availability that apply to Oracle VM Server for SPARC:

» **Avoid Single Points Of Failure (SPOFs).** Systems should be configured so a component failure does not result in a loss of application service. The general method to avoid SPOFs is to provide redundancy so service can continue without interruption if a component fails. Oracle VM Server for SPARC makes it possible to configure systems that avoid SPOFs.

» **Configure for availability** at a level of resource and effort consistent with business needs. Effort and resource should be consistent with business requirements. Production has different availability requirements than test/development, so it's worth expending resources to provide higher availability. Even within the category of production there may be different levels of criticality, outage tolerances, recovery and repair time requirements. A simple design may be more understandable and effective than a complex design that attempts to "do everything".

» **Design for availability** at the appropriate tier or level of the platform stack. Availability can be provided in the application, in the database, or in the virtualization, hardware and network layers they depend on - or using a combination of all of them. It may not be necessary to engineer resilient virtualization for stateless web applications where availability is provided by a network load balancer, or for enterprise applications like Oracle Real Application Clusters (RAC) and WebLogic that provide their own resiliency

» **It is often the same architecture whether virtual or not:** For example, providing resiliency against a lost device path or failing disk media is done for the same reasons and may use the same design whether in a domain or not.

» **It is often the same technique whether using domains or not:** Many configuration steps are the same. For example, configuring IPMP or creating a redundant ZFS pool is pretty much the same within the guest whether you're in a guest domain or not. There are configuration steps and choices for provisioning the guest with the virtual network and disk devices, which we will discuss.

» **Sometimes it is different when using domains:** There are new resources to configure. Most notable is the use of alternate service domains, which provides resiliency in case of a domain failure, and also permits improved serviceability via "rolling upgrades". This is an important differentiator between Oracle VM Server for SPARC and traditional virtual machine environments where all virtual I/O is provided by a monolithic infrastructure that is a SPOF. Alternate service domains are widely used to provide resiliency in production logical domains environments.

» **Some tasks are done via logical domain commands**, and some are done in the guest: For example, with Oracle VM Server for SPARC we provide multiple network connections to the guest, and then configure network resiliency in the guest via IP Multi Pathing (IPMP) - essentially the same as for non-virtual systems. On the other hand, we configure virtual disk availability in the virtualization layer and service domains, and the guest sees an already-resilient disk without being aware of the details.

» **Live migration is not "high availability**" in the sense of "continuous availability": If the server is down, then you don't live migrate from it, and a cluster or VM restart elsewhere would be used. However, live migration can be part of the RAS solution by improving Serviceability - you can move running domains off of a box before planned service or maintenance.

» **Save your domain configuration:** after making a change to your domains configuration that you want to persist over a powercycle, save it to the service processor by issuing an `ldm add-config` *`configname`* command. 'bootsets' containing the Service Processor (SP configuration and constraints are saved on the control domain when a configuration is saved to the SP, and are used to reload the constraints database after a power cycle. A copy is saved on the control domain `in /var/opt/SUNwldm/autosave-<configname>` and you can set an auto-recover policy for situations where the copy on disk is newer than the one saved on the service processor. This is documented at https://docs.oracle.com/cd/E80106_01/html/E80109/configurationmanagement.html

» **Backup your domain configuration:** save the domain configuration by exporting the configuration to an XML file, which can be saved on a separate host. Using `ldm list-constraints -x mydom > mydom.xml` saves the information for one domain, and `ldm list-constraints -x > everything.xml` saves the information for the entire system. If the domain configuration needs to be restored on a separate server, and individual domain can be restored by using a sequence like `ldm add-dom -I mydom.xml; ldm bind mydom; ldm start mydom` and the entire domain configuration can be restored from factory default mode by `using ldm init-system -i everything.xml`.

» **Configure domain dependencies:** if domains depend on one another, whether for service domains or application needs, you can express those relationships and set failure policies by defining their relationships as documented in https://docs.oracle.com/cd/E80106_01/html/E80109/configuredomaindependencies.html

Network Availability

Network connectivity is made resilient in guest domains by using redundant virtual network devices, just as redundant physical network devices are used with non-virtualized environments. Multiple virtual network devices are defined to the guest domain, each based on a different virtual switch ("vswitch") associated with a different physical backend. The backend can be either a physical link or an aggregation, which can provide additional resiliency against device failure and may improve performance.

Solaris IP Multipathing (IPMP) is configured within the guest much the same as in a physical instance of Solaris. If a link in an IPMP group fails, the remaining links provide continued network connectivity. The visible differences within the guest are the device names, since the "synthetic" device names for virtual network links (`vnet0`, `vnet1`, etc) are used instead of physical link names like `nxge0` or `ixgbe0`.

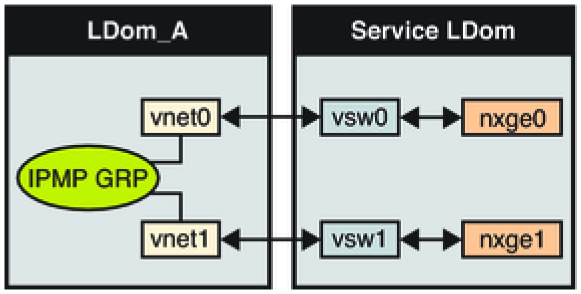

Figure 1 Two Virtual Networks Connected to Separate Virtual Switch Instances

This provides protection against failure of a switch, cable, NIC, and so on. Assuming that there are virtual switches in the control domain named `primary-vsw0 and primary-vsw1`, each using a different physical network device (that is, `primary-vsw0` could be based on the control domain's `net0`, and `primary-vsw1` based on `net1`) this is as simple as:

```
# ldm add-vnet linkprop=phys-state vnet0 primary-vsw0 ldg1
```

```
# ldm add-vnet linkprop=phys-state vnet1 primary-vsw1 ldg1
```

The optional parameter `linkprop=phys-`state indicates that the guest will use link-based IP Multipathing, so Solaris can detect link status changes. This parameter can be omitted if Solaris is going to use probe-based IPMP with a test IP address. For a full discussion, see **Using Virtual Networks and Configuring IPMP in a Logical Domains Environment**. https://docs.oracle.com/cd/E80106_01/html/E80109/configuringipmpinldomsenv.html The remainder of the exercise is to configure IPMP in the guest domain, which is essentially the same as in a non-domain environment.

There is additional protection if virtual network devices are provisioned from different service domains. If any of the service domains fails or is shutdown, network access continues over virtual network devices provided by the remaining service domains. Use of separate service domains is transparent to the guest Solaris OS, and the only configuration difference is that virtual switches from different service domains are used in the above command sequence.

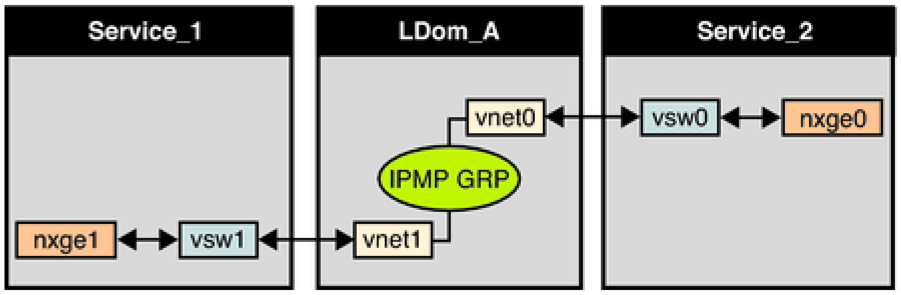

Figure 2 Virtual Network Devices Each Connected to Different Service Domains

Disk Availability

Configuring for redundancy is essential in order to tolerate media or path failure without data loss or outage. Oracle VM Server for SPARC uses methods similar to physical systems, either by provisioning virtual disks with backends that are themselves resilient, and/or by mirroring non-resilient disks.

Every virtual disk has a backend, which can be a physical disk or LUN, an iSCSI target, a disk volume based on a ZFS volume ("zvol") or volume manager such as Solaris Volume Manager (SVM), or a file on a file system (including NFS). The following graphic shows the relationship between the virtual disk client ("`vdc`") device driver in the guest domain, and the virtual disk server ("`vds`") driver in the service domain that work together, connected by a Logical Domain Channel (LDC) to associate the virtual disk to the physical backend:

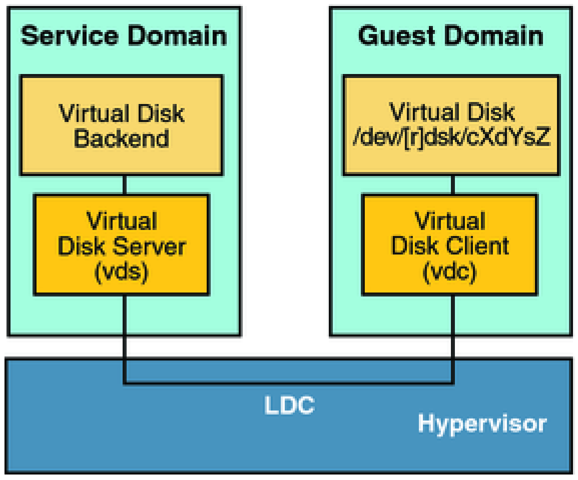
Service Domain
Virtual Disk Backend
Virtual Disk Server (vds)
Guest Domain
Virtual Disk /dev/[r]dsk/cXdYsZ
Virtual Disk Client (vdc)
LDC
Hypervisor

Figure 3 Virtual Disks with Oracle VM Server for SPARC

One of the most important domain configuration choices is whether a disk backend should be a physical device, such as a LUN or local disk, or a file in a file system mounted to a service domain. This has implications for performance, availability, and functionality such as the ability to perform live migration or support SCSI reservation. Physical devices (like a LUN) have the best performance, while file-based disk backends offer convenience, since they can be created as needed without needing to allocate physical devices from a storage array. Here are examples of using a physical device backend, a file-based backend, and a ZFS volume:

Virtual disk based on a LUN

```
# ldm add-vdsdev /dev/rdsk/c0t5000C5003330AC6Bd0s2 myguestdisk0@primary-vds0
# ldm add-vdisk my0 myguestdisk0@primary-vds0 mydom
```

Virtual disk based on a file (in this case, on NFS)

```
# mkfile -n 20g /ldomsnfs/ldom0/disk0.img
# ldm add-vdsdev /ldomsnfs/ldom0/disk0.img ldom0disk0@primary-vds0
# ldm add-vdisk vdisk00 ldom0disk0@primary-vds0 ldg0
```

Virtual disk based on a ZFS volume

```
# zfs create -V 20g rpool/export/home/ldoms/ldom1-disk0
# ldm add-vdsdev /dev/zvol/rdsk/rpool/export/home/ldoms/ldom1-disk0 \
              ldg1-disk0@primary-vds0
# ldm add-vdisk vdisk10 ldg1-disk0@primary-vds0 ldg1
```

The first example takes a device visible to the service domain (in this case, the control domain) and presents it to the guest. The service domain does not format it or mount a file system on it. That is the simplest backend: the service domain does a "passthrough" of all I/O activity. This is the recommended best practice for performance.

The other two examples create a virtual disk from a file or ZFS volume in a file system visible to the service domain. There is a layer of indirection and file buffering that increases latency, but the guest can leverage features like ZFS mirroring or RAIDZ, compression and cloning provided in the service domain's file system.

There are functional differences. For example, a local (non-NFS) ZFS disk backend cannot be used with live migration because a ZFS pool can only be imported to one Solaris instance (in this case, one control domain) at a time.

Disks can be made resistant to media failure using methods consistent with non-virtual environments: such as using disk backends on RAID or redundant ZFS environments, or combining virtual disks within the guest domain into a redundant ZFS pool.

An equally important aspect is ensuring redundant path availability to the device. Both Fibre Channel Storage Area Network (FC SAN) and NFS devices support multiple access paths. An FC SAN controller can have multiple host interfaces, preferably using more than one FC switch, to provide redundant access to multiple servers. Each server can have multiple Host Bus Adapter (HBA) cards to connect to the FC fabric. Together, they provide resiliency against the loss of an individual HBA, cable, FC switch, or host interface on the FC controller. This is similar to configuring multiple channels and control units as has long been done with mainframe DASD.

The same concept is available for virtual disks based on NFS or iSCSI. In those cases, the physical transport is Ethernet, and resiliency can be provided by multiple network devices, which may be network devices, aggregates, or IPMP groups.

Redundant disk paths are expressed in logical domains by using virtual disks defined with multipath groups ("`mpgroup`") as described in the Oracle VM Server for SPARC Administration Guide section “Configuring Virtual Disk Multipathing”. https://docs.oracle.com/cd/E80106_01/html/E80109/configuringvdiskmultipathing.html This creates an active/passive pair that continues to operate if one of the service domains is down. Note that this cannot be used for applications that require SCSI reservation. If SCSI reservation is needed, consider using virtual HBA (vHBA) instead.

The logical domains administrator defines multiple paths to the same disk backend (emphasized because this is crucial: the same device, accessed by multiple paths) and then adds the disk to the guest domain using one of the paths. This is illustrated below using an NFS backend available to multiple service domains.

```
# ldm add-vdsdev mpgroup=mpgroup1 /ldoms/myguest/disk0.img disk@primary-vds0
# ldm add-vdsdev mpgroup=mpgroup1 /ldoms/myguest/disk0.img disk@alternate-vds0
# ldm add-vdisk mydisk disk@primary-vds0 myguest
```

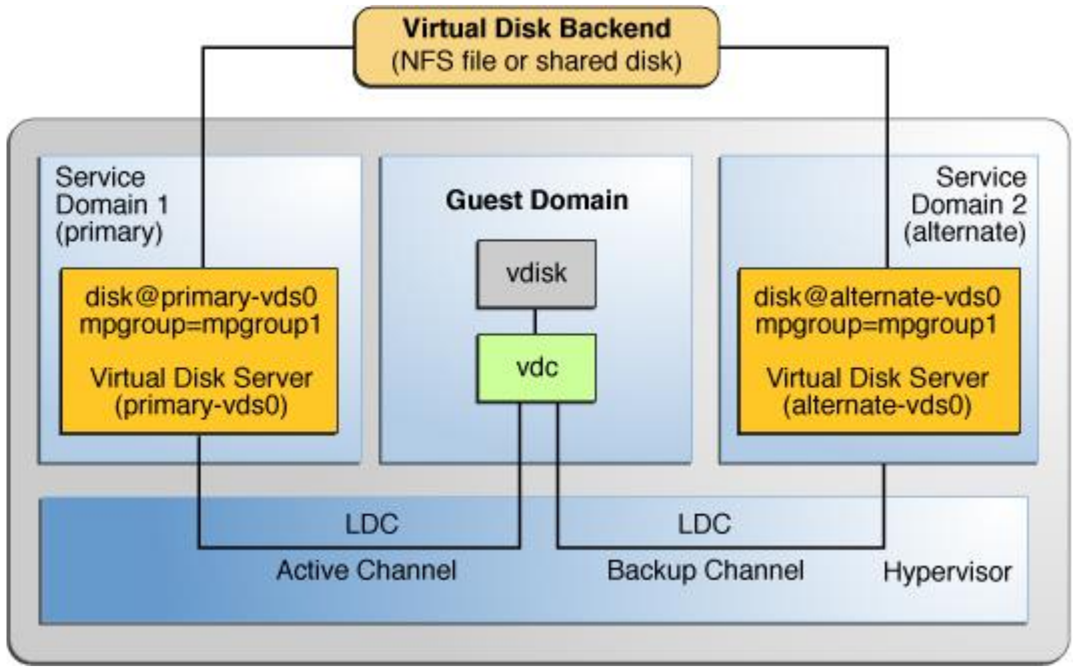
Virtual Disk Backend
(NFS file or shared disk)
Service Domain 1 (primary)
Guest Domain
Service Domain 2 (alternate)
vdisk
vdc
disk@primary-vds0 mpgroup=mpgroup1
Virtual Disk Server (primary-vds0)
disk@alternate-vds0 mpgroup=mpgroup1
Virtual Disk Server (alternate-vds0)
LDC
LDC
Active Channel
Backup Channel
Hypervisor

Figure 4 Configuring Virtual Disk Multipathing

Service Domain Availability Using Multiple Service Domains

A service domain can be a single point of failure. If it fails or is taken down for maintenance, then the devices it provides become unavailable and the guest domain access to I/O is suspended until the service domain is restored. The widely used best practice is to use multiple service domains to avoid this type of interruption. This can also be used for "rolling upgrades" in which service domains are taken down one at a time for maintenance. This avoids loss of service without requiring that a server be "evacuated" by live migration.

This section discusses details for configuring alternate service domains. The reference material is the Oracle VM Server for SPARC Administration Guide section “Creating a Root Domain by Assigning PCIe Buses”. https://docs.oracle.com/cd/E80106_01/html/E80109/configurepciexpressbusesacrossmultipleldoms.html

» Ensure your system is physically configured with network and disk devices on multiple PCIe buses so the control domain and I/O domains can operate using their own physical I/O. Note that built-in network and storage cards may reside on a single bus and cannot be shared by multiple root domains. Properly laying out redundant service domains on a given server requires careful consideration of the actual I/O connectivity of the server. Refer to the appropriate system service manual for details.

» Ensure that service domains do not depend on other service domains for their disk or network devices. That would create a single point of failure that negates the value of multiple service domains.

» Identify which PCIe buses are needed by the control domain. At a minimum, that includes buses with the control domain's boot disk and network link used to log into it. If the control domain is also a service domain, as is typical, it will also include the backend devices used for virtual devices it serves to guests.

» Identify which PCIe buses are needed by the root and alternate service domains. As with the control domain, this includes devices needed so the domain can boot, plus any devices used for virtual network and disk backends.

» Define the root domain to be used as an alternate service domain.

» Move selected buses from the control domain to the service domain. This will require a delayed reconfiguration and reboot of the control domain.

» From the control domain define virtual disk and network services using both control and service domains.

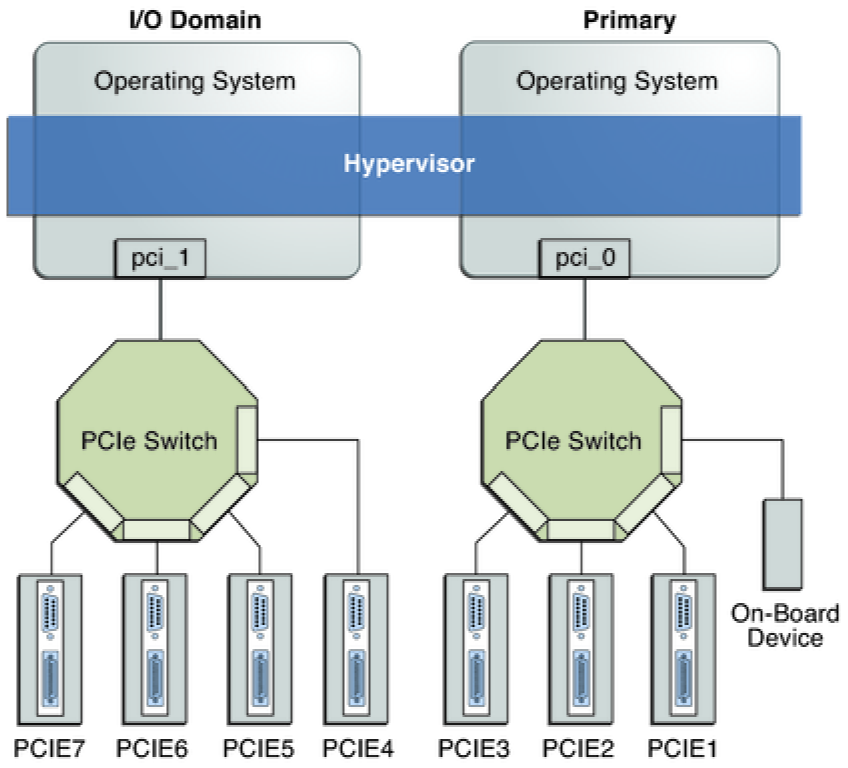

Figure 5 Assigning a PCIe bus to an I/O Domain

Example Configuration Using Multiple Service Domains

In this example, a server is configured with the control domain and a root complex domain. Both domains are root domains and are used as service domains, providing virtual disk and virtual network services to a guest domain.

Each domain is allocated the following set of resources, listed in Table 1:

Domain	CPU and memory	Devices
Control Domain	2 CPU cores (16 virtual CPUs) 16 GB	PCI buses (internal disks, on-board NICs, HBA connected to storage arrays)
Secondary service domain	2 CPU cores (16 virtual CPUs) 16 GB	PCI buses (internal disks, on-board NICs, HBA connected to storage arrays)
Guest domains (variable number)	As needed for application requirements	Virtual disk and network devices provided by the control domain and secondary service domain

This configuration does not use all resources of the system. Unallocated resources can be used to create additional domains or to reconfigure the existing domains if more resources are required, for example to handle a more important workload. `The ldm list-devices` command can be used to display available resources. A valuable feature of Oracle VM Server for SPARC is that resources can be dynamically allocated as needed.

To act as a service domain, the control domain will have the following virtual device services:

» One virtual disk service (`primary-vds0`) used to export physical disks as virtual disks to guest domains.

» One virtual console concentrator service (`primary-vcc0`) used to provide access to the virtual consoles of the guest domains. This service will use the port range 5000-5100 for virtual console access.

» At least one virtual switch service, in this `example primary-vsw0` associated with the network `net0`.

The secondary service domain has similar services, with names adjusted to indicate the domain name.

Software Installation and Configuration

The following tasks are part of server software installation:

» Oracle Solaris installation

» Oracle VM Server for SPARC installation

» Control domain configuration

» Secondary service domain configuration

» Guest domain configuration

Additional configuration tasks, including network and storage related tasks, are covered in later sections of this document. Oracle installation occurs as a separate step, after the network and storage devices are configured

Oracle Solaris Installation

Oracle SPARC systems come pre-installed with Oracle Solaris and Oracle Solaris 11 comes with Oracle VM Server for SPARC pre-installed. Ensure that the appropriate releases of Oracle Solaris OS and Oracle VM Server for SPARC are installed for your server, and check that required patches are present.

You may also choose to re-install the entire system so that it conforms to your installation policy and matches your requirements. Refer to the Oracle Solaris OS installation documentation for more information on how to install the Oracle Solaris OS on SPARC servers.

After Oracle Solaris OS is installed, the system can be configured to use logical domains.

Oracle VM Server for SPARC Installation

This example assumes that the Oracle Solaris 11 OS and required patches are already installed. Oracle VM Server for SPARC software is included by default with the Oracle Solaris 11 OS so it doesn't need to be installed. For a complete procedure on how to install logical domains refer to the Oracle VM Server for SPARC Installation Guide https://docs.oracle.com/cd/E80106_01/html/E80108/index.html

Control Domain Configuration

This example uses a SPARC T5-8 to illustrate creation of a root complex domain to be used as a control and service domain, and a second root complex domain to be used as a secondary service domain. The installed Solaris instance is first reconfigured to become the control domain. That includes defining default services and resizing the control domain to release CPU and memory resources to use for other domains to be defined later. We also identify buses to keep for the control domain, and buses to assign to the root complex domain.

The system is delivered with a single control domain owning all CPU, I/O and memory resources as shown below: Some output is snipped out for brevity.

```
primary# ldm list -l
NAME        STATE      FLAGS   CONS    VCPU  MEMORY    UTIL  NORM  UPTIME
primary    active     -n-c--  UART    1024  1047296M 0.0%  0.0%  2d 5h 11m
----snip----
CORE
    CID    CPUSET
    0      (0, 1, 2, 3, 4, 5, 6, 7)
    1      (8, 9, 10, 11, 12, 13, 14, 15)
    2      (16, 17, 18, 19, 20, 21, 22, 23)
    3      (24, 25, 26, 27, 28, 29, 30, 31)
----snip----
    124    (992, 993, 994, 995, 996, 997, 998, 999)
    125    (1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007)
    126    (1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015)
    127    (1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023)
VCPU
    VID    PID    CID    UTIL NORM STRAND
    0      0      0      4.7% 0.2%   100%
    1      1      0      1.3% 0.1%   100%
    2      2      0      0.2% 0.0%   100%
    3      3      0      0.1% 0.0%   100%
----snip----
    1020   1020   127    0.0% 0.0%   100%
    1021   1021   127    0.0% 0.0%   100%
    1022   1022   127    0.0% 0.0%   100%
    1023   1023   127    0.0% 0.0%   100%
----snip----
IO
    DEVICE                           PSEUDONYM        OPTIONS
    pci@300                          pci_0
    pci@340                          pci_1
    pci@380                          pci_2
    pci@3c0                          pci_3
    pci@400                          pci_4
    pci@440                          pci_5
    pci@480                          pci_6
    pci@4c0                          pci_7
    pci@500                          pci_8
    pci@540                         pci_9
    pci@580                         pci_10
    pci@5c0                         pci_11
    pci@600                         pci_12
    pci@640                         pci_13
    pci@680                         pci_14
    pci@6c0                         pci_15
----snip----
```

Basic domain configuration

The following commands are basic configuration steps to define virtual disk, console and network services and resize the control domain. They are shown for completeness but are not specifically about configuring for availability. In this example we assume that `net0` will be used for virtual switch. Since this is a Solaris 11 control domain, we use the vanity name for the network interface, `net0`, rather than the device driver.

```
primary# ldm add-vds primary-vds0 primary
primary# ldm add-vcc port-range=5000-5100 primary-vcc0 primary
primary# ldm add-vswitch net-dev=net0 primary-vsw0 primary
primary# ldm set-core 2 primary
primary# svcadm enable vntsd
primary# ldm start-reconf primary
primary# ldm set-mem 16g primary
primary# shutdown -y -g0 -i6
```

This is standard control domain configuration. After reboot, we have a resized control domain, and save the configuration to the service processor so it will persist over a power cycle.

```
primary# ldm list
NAME          STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
primary       active     -n-cv-  UART    16    16G      3.3%  2.5%  4m
primary# ldm add-spconfig initial
```

Remember to save the configuration to the SP so it persists across a powercycle, and save it as XML for export to other systems. See https://docs.oracle.com/cd/E80106_01/html/E80109/managingldomsconfigurations.html

Determine which buses to assign to a root complex domain

We next determine which buses must be kept on the control domain and which can be assigned to another root domain. This procedure is documented at “Creating a Root Domain by Assigning PCIe Buses” in the Oracle VM Server for SPARC Administration Guide.
https://docs.oracle.com/cd/E80106_01/html/E80109/configurepciexpressbusesacrossmultipleldoms.html

First, identify the PCIe bus used for the root pool disk (in a production environment this would be mirrored) by getting the device name and then using the `mpathadm` command to obtain the initiator port name, and then determine its device path.

```
primary # zpool status rpool
  pool: rpool
 state: ONLINE
  scan: none requested
config:
        NAME                       STATE     READ WRITE CKSUM
        rpool                      ONLINE       0     0     0
          c0t5000CCA01605A11Cd0s0  ONLINE       0     0     0
errors: No known data errors
primary # mpathadm show lu /dev/rdsk/c0t5000CCA01605A11Cd0s0
Logical Unit:  /dev/rdsk/c0t5000CCA01605A11Cd0s2
----snip----
        Paths:
                Initiator Port Name:  w508002000145d1b1
----snip----
primary # mpathadm show initiator-port w508002000145d1b1
Initiator Port:  w508002000145d1b1
        Transport Type:  unknown
        OS Device File:
/devices/pci@300/pci@1/pci@0/pci@4/pci@0/pci@c/scsi@0/iport@1
```

That shows that the boot disk is on bus `pci@300`, which corresponds to the pseudonym `pci_0`.

Next we determine which bus is used for network. Interface `net0` (based on `ixgbe0`) is our primary interface and hosts a virtual switch, so we need to keep the bus it is attached to.

```
primary# ls -l /dev/ix*
lrwxrwxrwx   1 root     root          31 Jun 21 12:04 /dev/ixgbe ->
../devices/pseudo/clone@0:ixgbe
lrwxrwxrwx   1 root     root          65 Jun 21 12:04 /dev/ixgbe0 ->
../devices/pci@300/pci@1/pci@0/pci@4/pci@0/pci@8/network@0:ixgbe0
lrwxrwxrwx   1 root     root          67 Jun 21 12:04 /dev/ixgbe1 ->
../devices/pci@300/pci@1/pci@0/pci@4/pci@0/pci@8/network@0,1:ixgbe1
lrwxrwxrwx   1 root     root          65 Jun 21 12:04 /dev/ixgbe2 ->
../devices/pci@6c0/pci@1/pci@0/pci@c/pci@0/pci@4/network@0:ixgbe2
lrwxrwxrwx   1 root     root          67 Jun 21 12:04 /dev/ixgbe3 ->
../devices/pci@6c0/pci@1/pci@0/pci@c/pci@0/pci@4/network@0,1:ixgbe3
```

This should also be repeated for the device `/dev/usbecm` which is used for Fault Management Architecture (FMA) fault proxying. Normally it is on the same bus so there is no difference in bus assignment, but customers should check the manuals for their servers' models. In this case, disk and network are on bus `pci@300` (`pci_0`), and there are network devices on `pci@6c0` (`pci_15`) that we can give to an alternate service domain.

We also determine which buses are needed to give that service domain disk access. Previously we saw that the control domain's root pool was on `c0t5000CCA01605A11Cd0s0` on `pci@300`. The control domain currently has access to all buses and devices, so we can use the format command to see what other disks are available. There is a second disk, and it's on bus `pci@6c0`:

```
primary# format
Searching for disks...done
AVAILABLE DISK SELECTIONS:
       0. c0t5000CCA01605A11Cd0 <HITACHI-H109060SESUN600G-A244 cyl 64986 alt 2 hd 27
sec 66>
          /scsi_vhci/disk@g5000cca01605a11c
          /dev/chassis/SPARC_T5-8.1239BDC0F9//SYS/SASBP0/HDD0/disk
       1. c0t5000CCA016066100d0 <HITACHI-H109060SESUN600G-A244 cyl 64986 alt 2 hd 27
sec 668>
          /scsi_vhci/disk@g5000cca016066100
          /dev/chassis/SPARC_T5-8.1239BDC0F9//SYS/SASBP1/HDD4/disk
Specify disk (enter its number): ^C

primary# mpathadm show lu /dev/dsk/c0t5000CCA016066100d0s0
Logical Unit:  /dev/rdsk/c0t5000CCA016066100d0s2
----snip----
        Paths:
                Initiator Port Name:  w508002000145d1b0
----snip----
primary# mpathadm show initiator-port w508002000145d1b0
Initiator Port:  w508002000145d1b0
        Transport Type:  unknown
        OS Device File:
/devices/pci@6c0/pci@1/pci@0/pci@c/pci@0/pci@c/scsi@0/iport@1
```

At this point we have all the information needed to reassign buses.

`pci@300` (`pci_0`) has control domain's boot disk and primary network device.

`pci@6c0` (`pci_15`) has unused disk and network devices, and can be used for another domain.

We can now define a secondary service domain, then remove the above buses from the control domain and assign them to a root complex domain to be used as an alternate service domain. Removing the buses will require a second reboot, which could be avoided by obtaining the bus information before rebooting to resize the control domain's memory.

```
primary# ldm add-dom secondary
primary# ldm set-core 2 secondary
primary# ldm set-mem 16g secondary
primary# ldm start-reconf primary
primary# ldm rm-io pci_15 primary
```

```
primary# init 6
```

After rebooting the control domain, give the unassigned `bus pci_15` to the secondary domain. At this point you can install Solaris in this domain using a network install server. Alternatively, Solaris could be installed from boot media in a `.iso` file exported by the control domain's virtual disk service. Normally you do not use virtual I/O devices in a service domain because that introduces a dependency on another service domain, but this is acceptable in this case because the virtual device will be removed after Solaris is installed.

```
primary# ldm add-io pci_15 secondary

primary# ldm bind secondary

primary# ldm start secondary
```

Note that network devices on `pci@6c0` are enumerated in the service domain starting as `ixgbe0`.

```
secondary# ls -l /dev/ixgb*

lrwxrwxrwx   1 root     root          31 Jun 21 10:34 /dev/ixgbe ->
../devices/pseudo/clone@0:ixgbe

lrwxrwxrwx   1 root     root          65 Jun 21 10:34 /dev/ixgbe0 ->
../devices/pci@6c0/pci@1/pci@0/pci@c/pci@0/pci@4/network@0:ixgbe0

lrwxrwxrwx   1 root     root          67 Jun 21 10:34 /dev/ixgbe1 ->
../devices/pci@6c0/pci@1/pci@0/pci@c/pci@0/pci@4/network@0,1:ixgbe1
```

Defining Redundant Services

We now have a root I/O domain that can boot and run independently on its own PCIe bus. All that remains is to define redundant disk and network services to pair with the ones defined above in the control domain.

```
primary# ldm add-vds secondary-vds0 secondary

primary# ldm add-vsw net-dev=net0 secondary-vsw0 secondary

primary# ldm add-spconfig mysecondary
```

Guest Domains

You can create guest domains that use services provided by the control and secondary service domains. To illustrate this, a guest domain will be defined that has two network devices so it can use IP Multipathing (IPMP) and a virtual disk with a path from both the control and alternate domains.

```
primary# ldm add-dom ldg1

primary# ldm set-core 16 ldg1

primary# ldm set-mem 64g ldg1

primary# ldm add-vnet linkprop=phys-state ldg1net0 primary-vsw0 ldg1

primary# ldm add-vnet linkprop=phys-state ldg1net1 secondary-vsw0 ldg1

primary# ldm add-vdsdev mpgroup=ldg1group /dev/dsk/xxxxx ldg1disk0@primary-vds0

primary# ldm add-vdsdev mpgroup=ldg1group /dev/dsk/xxxxx ldg1disk0@secondary-vds0

primary# ldm add-vdisk ldg1disk0 ldg1disk0@primary-vds0 ldg
```

Substitute the device paths to the LUN for `/dev/dsk/xxxxx` above - they should be the path from each domain pointing to the same LUN.

Note the `linkprop=phys-state` on the virtual network device definition. This indicates that changes in physical link state should be passed to the virtual device so it can detect link failure in a service domain and perform a failover. Also note the `mpgroup` on each of the virtual disks. They create two paths to each disk so each disk can be accessed from either the control domain or the secondary domain.

At this point, Solaris can be installed in the guest domain without any special procedures.

Configuring and Testing Resiliency

Multipath disk I/O is transparent to the guest domain. This was tested by serially rebooting the control domain or the secondary service domain, and observing that disk I/O operation proceeded without noticeable effect.

Network redundancy requires configuring IP Multipathing (IPMP) in the guest domain. The guest has two network devices, `net0` provided by the control domain, and `net1` provided by the secondary domain. The following commands are executed in the guest domain to make a redundant network connection:

```
ldg1# ipadm create-ipmp ipmp0
ldg1# ipadm add-ipmp -i net0 -i net1 ipmp0
ldg1# ipadm create-addr -T static -a 10.134.116.224/24 ipmp0/v4addr1
ldg1# ipadm create-addr -T static -a 10.134.116.225/24 ipmp0/v4addr2
ldg1# ipadm show-if
IFNAME      CLASS    STATE    ACTIVE OVER
lo0         loopback ok       yes    --
net0        ip       ok       yes    --
net1        ip       ok       yes    --
ipmp0       ipmp     ok       yes    net0 net1
ldg1# ipadm show-addr
ADDROBJ            TYPE     STATE        ADDR
lo0/v4             static   ok           127.0.0.1/8
ipmp0/v4addr1      static   ok           10.134.116.224/24
ipmp0/v4addr2      static   ok           10.134.116.225/24
lo0/v6             static   ok           ::1/128
```

Resiliency was tested by rebooting the secondary service domain and control domain one at a time, and noting that guest network sessions were intact. The guest’s console displayed messages when a link failed and was restored:

```
Jul  9 10:35:51 ldg1 in.mpathd[107]: The link has gone down on net1
Jul  9 10:35:51 ldg1 in.mpathd[107]: IP interface failure detected on net1 of group
ipmp0
Jul  9 10:37:37 ldg1 in.mpathd[107]: The link has come up on net1
```

While one of the service domains was down, `dladm` and `ipadm` in the guest domain showed link status:

```
ldg1# ipadm show-if
IFNAME     CLASS    STATE    ACTIVE OVER
lo0        loopback ok       yes    --
net0       ip       ok       yes    --
net1       ip       failed   no     --
ipmp0      ipmp     ok       yes    net0 net1
ldg1# dladm show-phys
LINK              MEDIA                STATE      SPEED  DUPLEX    DEVICE
net0              Ethernet             up         0      unknown   vnet0
net1              Ethernet             down       0      unknown   vnet1
ldg1# dladm show-link
LINK                CLASS     MTU    STATE     OVER
net0                phys      1500   up        --
net1                phys      1500   down      --
```

When the service domain finished rebooting, the "`down`" status returned to "`up`". There was no outage at any time, and no manual intervention was needed to re-enable the restored path

Conclusion

Oracle VM Server for SPARC is a virtualization technology that allows the creation of multiple virtual systems on a single physical system. Multiple logical domains can be configured on the same physical server to reduce costs and improve operational agility and efficiency.

This paper presents best practices on how to configure Oracle VM Server for SPARC to achieve high performance and availability. Multiple options are available, depending on the application and system requirements. Configuration guidelines and software requirements are also included, to help administrators plan their deployments.

For More Information

Title	URL
Oracle Virtualization	http://oracle.com/virtualization
Oracle VM OTN (Oracle Technology Network)	http://www.oracle.com/technetwork/server-storage/vm/overview/index.html
Oracle VM Server for SPARC Technical White Paper	http://www.oracle.com/technetwork/server-storage/vm/documentation/logical-domains-articles-160640.html
Best Practices for Data Reliability with Oracle VM Server for SPARC	http://www.oracle.com/technetwork/articles/systems-hardware-architecture/vmsrvrsparc-reliability-163931.pdf
Best Practices for Network Availability with Oracle VM Server for SPARC	*http://www.oracle.com/technetwork/articles/systems-hardware-architecture/vmsrvrsparc-availability-163930.pdf*

How to Get the Best Performance from Oracle VM Server for SPARC	*http://www.oracle.com/technetwork/articles/servers-storage-admin/solaris-network-vm-sparc-2275380.html*
SPARC Servers System Documentation	*http://www.oracle.com/technetwork/server-storage/sun-sparc-enterprise/documentation/index.html*
Oracle SPARC Servers White Papers	*http://www.oracle.com/technetwork/server-storage/sun-sparc-enterprise/documentation/whitepapers-2158892.html*

.

Oracle VM Server for SPARC Best Practices
April 2018
Author: Jeff Savit