ORACLE

SUN ZFS STORAGE APPLIANCE

An Oracle Technical White Paper
November 2012

Aligning Partitions to Maximize Storage Performance

ORACLE

Table of Contents

Table of Figures

Introduction

This white paper focuses on the importance of correctly partitioning a logical LUN or hard disk.

Configuring disks and storage subsystems for optimal application performance is often perceived both as an art and a science best left to experts. But partition alignment only requires some basic knowledge, and getting the configuration correctly set up will avoid the risk of significant I/O performance loss. Documentation of storage subsystems and best practices often overlooks the very basic step of aligning disk partitions with the block structure of the underlying storage hardware.

For a computer system to be able to use a storage unit, the storage unit must be initialized by a software partitioning tool. This software partitioning tool is part of the particular operating system software installed on the computer system.

The partitioning software creates logical storage units for use by the operating system. Most partitioning software still operates under the standard that a disk uses 512-byte (B) size sectors on physical platters that contain tracks. Partitions are created with start and end boundaries expressed in cylinders using Cylinder, Head, Sector addressing mechanisms. However, the new generation of disk drives and solid state drives (SSDs) use larger-sized sectors and have no fixed number of sectors per cylinder. For logical units presented through Fibre Channel or iSCSI protocols, the concept of sectors and cylinders is not applicable anymore, and Logical Block Addressing (LBA) is used with block sizes far exceeding the original sector size of 512B.

When partitioning tools are still using 512B-sector-based addressing schemes, the start of a partition will most likely *not* be located at the start of a logical block of the underlying disk or LUN. This can have a severe impact on the overall performance of the I/O to and from that disk or LUN. A logical write to one block can translate to a physical read-modify-write operation on two disk blocks. So it is very important that when creating partitions to ensure that the start of each partition created on a disk or LUN is aligned with the start of a physical sector of a disk or logical block of a LUN. Once a partition is created, the start location cannot be changed without deleting the partition and thus losing any data stored within that partition.

This paper describes, in detail, partition structure and methods to ensure the partition’s alignment to disk drive sectors or LUN LBAs. It also explains how to optimally configure a LUN (and its LBA size) as presented by a Sun ZFS Storage Appliance.

Lastly, it shows how to use partitioning tools on a number of vendors' operating systems for optimal use of LUNs served from the Sun ZFS Storage Appliance.

Preparing to Use a Hard Disk

Before operating systems can use hard drives, the drives must be initialized with some basic information. This information consists of partition information and optional boot software locations as well as the boot software itself. Size and locations used in this information are expressed in disk drive hardware characteristics in the form of sector, track, cylinder or LBA entities.

To explain these characteristics, the following sections review the physical side of a disk drive, and how data is organized within a disk drive.

How Disks Work

Computer disk storage systems, particularly the 305 RAMAC, were introduced by IBM in 1956. The 305 RAMAC held 5 MB and had 24-inch-diameter disks. It took till 1980 for the first disk drives for microcomputers to be introduced and disks to become more widely used.

The technology used has not changed much over time. A disk drive consists of three main components: one or more platters, heads, and control electronics.

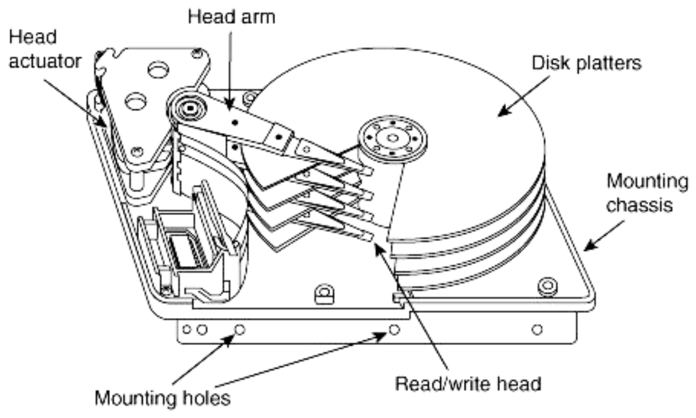


Figure 1. Hard disk drive components

Platters are stacked and held into position by a single spindle. Read/write heads are positioned above each platter. The platters are coated with a magnetic material that is used to store the data. Each platter is divided into many tracks: very thin concentric circles in which the data is stored. Each track is divided into a number of sectors. These sectors have always been 512B in size, but 4KB sector size is also now used, especially with solid state drives.

Disk Addressing Methods

All tracks at the same position on each platter are called cylinders. Data is located by a cylinder number, head number and sector number. The controlling electronics in the disk drive position the heads to the right cylinder, activate the right head and wait for the right sector to pass to be able to read data from a sector or write data to a sector. This form of location addressing is called CHS (Cylinder, Head, Sector), or geometry-based access.

Next to this data storage organization structure, the disk drive also contains a structure, put in place by the drive manufacturer, to correctly position the heads over the data track. Details of that structure are not relevant here.

In initial designs, each track held the same number of sectors. The density of information would be the highest on the inner tracks of the disk, determining the maximum number of sectors to be used.

By introducing variable data rate recording techniques to increase a disk’s capacity, more sectors could be used on the outer tracks than on the inner tracks. The disk electronics used a translation schema to make the disk appear as having a fixed-sector-per-track schema. This way the method of geometry-based disk access continued to work.

In a lot of computer architectures a low-level I/O interface software layer is used to abstract hardware-specific interfaces from the computer OS. This software, known as the BIOS (Basic Input/Output Software), resides in a chip of the computer's hardware. The BIOS determines the geometry of the disk drive and offers a set of functions to the OS to manipulate the drive based on geometry-type disk access.

Given the increasing disk capacity and more complex disk internal data addressing structures, the BIOS in current computers no longer uses fixed CHS mapping schema. The disk size is determined at boot time and a Logical Block Addressing (LBA) schema is used. Instead of referring to Cylinder, Head and Sector numbers, the disk is seen as a unit with a continuous number of blocks, with each block being 512B in size. Current disk drives all support LBA – a form of addressing already known in the SCSI specification. If needed, the BIOS I/O functions perform the translation between LBA and CHS addressing.

Most current operating systems use their own disk I/O layer in which a direct LBA access method is used to access the disk.

Hard Disk Interfaces

Hard disk drives come with a number of different interfaces, or type of bus connection technology used, to connect a disk to a computer system. This technology has evolved from separate data and control connections to a single serial type of connection.

Currently, the common interfaces are ATA, Serial ATA (SATA), FC, SAS, SCSI and iSCSI.

Advanced Technology Attachment (ATA)

ATA is also known by the marketing name IDE. This interface is commonly used in the PC x86 world. The bus supports two devices: master and slave. It does not support command queuing. This technology is rapidly being replaced by SATA. Older drives use CHS addressing schema, while newer drives use LBA.

Serial ATA (SATA)

SATA is the serial version of the ATA disk format. A SATA bus does not have a master/slave relationship. The connections are point-to-point with a small cable. SATA transfer speed is 150MB/s, with second-generation SATA-II boosting this to 300MB/s. SATA drives use the LBA schema.

Small Computer System Interface (SCSI)

SCSI is a technology used in the high-end server market, as well as for I/O devices like tape drives and CD-ROM drives. By using command tag queuing, SCSI can achieve higher performance than ATA.

Serial Attached SCSI (SAS)

SAS is an improved version of the parallel SCSI technology. SAS differs from SCSI in that the connection between initiator and target is point to point and supports higher data transfer rates. Disk access is still according to SCSI LBA schema.

Fibre Channel (FC)

Fibre Channel is a serial optical or copper interface. A serial SCSI protocol is used on top of the physical FC packet frame transport layer. FC is mainly used in high-end storage solutions. Since SCSI is employed, the SCSI LUNs use an LBA access schema. Using FC requires a dedicated Storage Area Network (SAN) infrastructure.

All of the preceding interface technologies require a Host Bus Adapter (HBA) to handle the specifics of the bus technology transfer commands.

iSCSI

Using SCSI protocol over IP interconnect, iSCSI is a lower-cost alternative for a SAN-based solutions. No special hardware is needed to build a storage infrastructure, but performance may be impacted by other systems sharing the same network infrastructure. As with FC type storage, iSCSI LUNs are accessed using LBA-type access schema.

Storage Natural Block Sizes

As mentioned before, the smallest unit of storage used in disk technology is a sector. Traditionally, sectors were always 512B in size, but with disk technology's increasing data storage densities, this became inefficient. In response to the recognized need for larger data sectors, the International Disk Drive Equipment and Materials Association (IDEMA) led an industry-wide initiative to transition from a 512- to 4096-byte-per-sector standard – known as Advanced Format. So the natural block size (NBS) of a disk is its physical sector size. However, some disk drive vendors still report block size of 512B when physically using a different sector size. This makes it hard for disk management software to create an optimal partition layout for such drives.

Currently there are three types of disk drives:

- 512n disks – Disks using 512B sectors, meaning a natural block size, or NBS, of 512B.
- 4Kn disks – Disks using 4096B sectors and reporting an NBS of 4096B.
- 512e disk – Disks using a sector size of 4096B but emulating or reporting sectors of 512B NBS. These disks are problematic for disk labeling tools, which cannot determine the real NBS from the information reported by the disk.

For storage volumes exported by storage subsystems, the NBS does not have a fixed/standardized value. As the LBA addressing method is used in these applications, the LUN NBS does not have restrictions and is defined during the LUN's creation. The storage subsystem software manages the mapping of LUN blocks to physical disk sectors within the storage subsystem. This enables you to select a block size that is optimal for the I/O characteristics of the application and match the block size of the database or file system of the application host. The rule of thumb for selecting block sizes is to choose small block sizes (4-8KB) for online transaction processing (OLTP) applications and large block sizes (64-128KB) for sequential/streaming-type workloads.

Applying Partitions to Disk Drives

Most operating systems use some form of logical storage unit concept on a disk drive to organize and store information. These logical units are referred to as partitions or, with some operating systems, slices. This section explains the general concept of storage partitions and different techniques used by the major operating systems for their creation.

Partition management software is used to create, delete, resize partitions and manipulate the characteristics of partitions. Each operating system usually contains its own version of such a tool. In most cases, the partitioning software also adds boot information and software onto the disk.

Changing Standards for Partitioning

The most commonly used partition format, called Master Boot Record (MBR), stems from the PC DOS-based computer architecture. This partitioning scheme consists of a boot record and up to four partitions. The MBR is located at the start of the disk and contains the boot code and a table of four partition definitions. Each partition is defined by its length and start location which, in turn, are defined by a Cylinder, Head and Sector (CHS) field – known as a CHS address. The sector count starts at 1 (rather than zero) and the number of heads is limited to 255. Given the head limit and size limit of the cylinder field, this schema works for disk sizes smaller than 2 terabytes (TB).

Because of the disk size limitation with MBR partitions, a newer standard was developed: the GUID Partition Table (GPT) standard. This standard uses LBA instead of CHS addressing to define the location of a partition. The GPT definition is part of a wider Unified Extensible Firmware Interface (UEFI) specification. GPT-based partition schemas are often referred to as EFI System Partition-based systems. With the GPT-standard partitions, the LBA count starts at 0 rather than 1, as with CHS addressing.

Disks over 2TB in capacity require the GPT partition scheme. For an operating system to read from a GPT-formatted disk it needs to support the GPT fdisk labeling structure. Some operating systems require hardware using EFI firmware (BIOS) in order to boot from a GPT-labeled disk.

UNIX-based systems started with their own disk labeling structure that is different from the PC MBR-based partitioning scheme. In that partitioning scheme the term slices is used for disk partitions. Information about the disk's geometry and slice layout are stored in a label area. This information is referred to as the Volume Table of Contents (VTOC). Writing VTOC information on a disk is often referred to as “labeling” a disk. When UNIX was ported to the x86 platform, the PC DOS MBR partitioning schema was kept in place, and the UNIX VTOC schema was layered on top of the MBR partitioning schema, so that the VTOC partitioning schema is written within a MBR partition.

For example, in the Oracle Solaris operating system, an Oracle Solaris-type partition is created in an MBR schema. Oracle Solaris treats this MBR partition as an Oracle Solaris disk volume on which it writes the UNIX VTOC information containing the defined slices for that volume. This Oracle Solaris-based disk label may be called an SMI (Sun Microsystems Inc.) label.

Note that the terms “slices” and “partitions” are often intermixed and may cause confusion. When referring to a specific partition, you must also clearly identify which level in the partition hierarchy the partition resides upon. Failure to understand this concept can introduce possible misalignment on two levels: the start of the MBR partition containing the UNIX volume, and the start of the slices within the UNIX volume.

The Linux operating system uses the DOS partitioning schema directly and does not use any extra slicing mechanism. Oracle Solaris uses the extra level of slicing in the SMI-type partitioning schema, while the newer EFI labeling schema is native to the disk and uses no extra layering.

How Changing Standards Affect Partition Tools and Alignment

As disk drives’ storage capacity and storage block organization continue to change, the tools used to prepare storage devices for operating system addressing have not been keeping up.

Today's storage devices and storage subsystems presenting storage units have moved away from the original hardware sector size of 512 bytes. The new generation of SATA drives and Solid State Devices (SSDs) use an NBS of 4KB. LUNs presented by a Sun ZFS Storage Appliance have a default NBS of 8KB.

For optimal performance of the storage units, it is important that the block structure used by the file system is aligned with the block structure of the storage units. Proper alignment is achieved if the first block of a file system file structure starts at the begining of a block from the storage subsystem and the size of a file system block is equal or is an even multiple of the NBS of a storage device.

The following diagram shows a misalignment of file system blocks and the storage subsystem blocks caused by a partition misalignment. This type of ‘classic’ misalignment occurs because the disk partitioning tools still default to 512B-sized blocks for the storage devices. This model for creating partitions stems from the PC DOS environment and is still used on various operating systems running on x86 platforms.

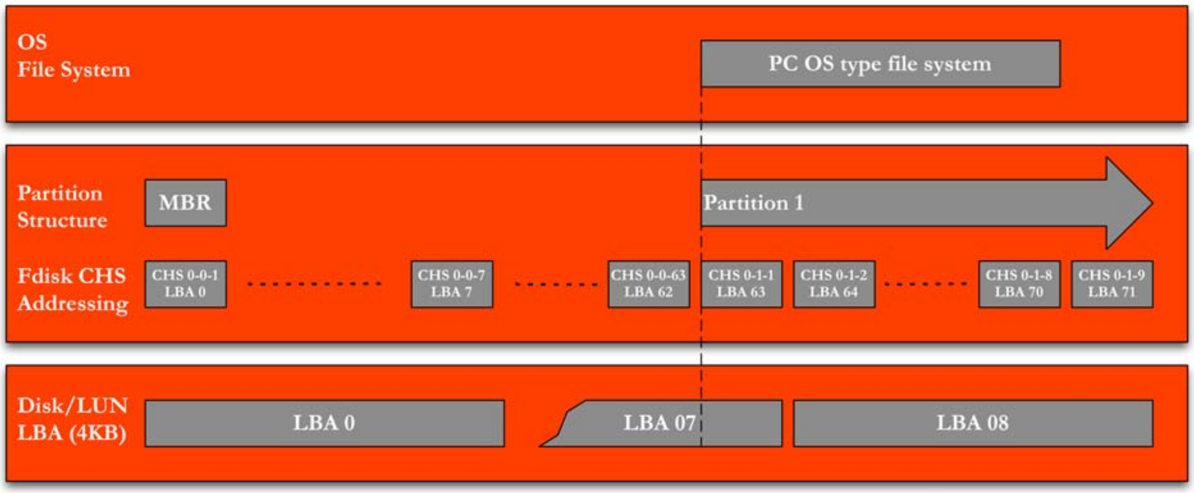


Figure 2. Misaligned partition example

Figure 2 shows three layers: the LBA block structure used on a disk or LUN, a view of the disk/LUN from the partitioning tool fdisk and the partition it created, and a Microsoft Windows-based file system structure called New Technology File System (NTFS) created on the first partition.

Fdisk is not aware of the type of block structure and the size of its blocks of the underlying disk/LUN. Fdisk defaults to sector 63 as the start sector for the first partition and 512B as the block size for the sectors.

Newer operating systems and file systems have moved away from this original partitioning schema in order to avoid I/O misalignment. Oracle's ZFS, for instance, uses 256 as the start sector when a disk is directly put under ZFS control.

In the following illustration, the disk/LUN uses an NBS of 4KB.

The default block size (cluster size) in an NTFS-based file system varies between 4KB and 64KB, depending on the size of the partition the file system is created in. Because the start of the partition is not aligned with the start of a block on the disk/LUN, each time an NTFS cluster block is read, two blocks on the disk/LUN need to be read. When a block on the file system is written, two blocks on the disk/LUN need to be read and written back, resulting in four physical I/Os for one logical I/O!

Ensuring that the first sector of a partition as specified in fdisk is located at the start of a disk/LUN block prevents extra I/Os being issued to the disk/LUN when blocks are updated in the file system layer as shown in the following illustration. When a block in the file system is read or written, only one block on the underlying disk/LUN needs to be accessed, as seen in Figure 3.

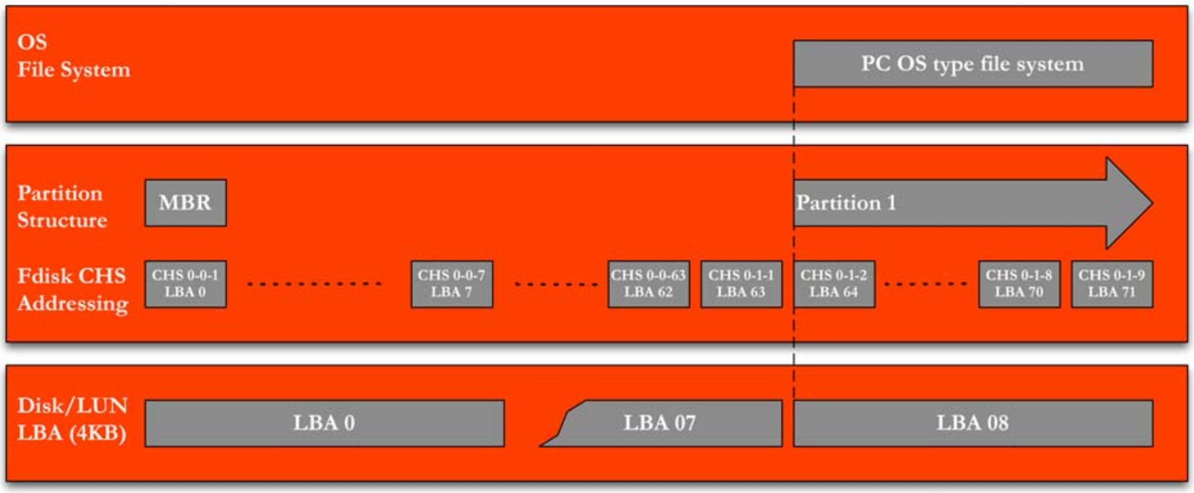


Figure 3. Aligned fdisk partition example for Microsoft Windows-based operating system

Three main reasons that may lead to I/O misalignment issues:

- The tools available to create and manage partitions still use the CHS abstraction model in which a disk/LUN is carved up in sectors and cylinders, with a sector size of 512B. These tools are not aware of disks/LUNs using the LBA abstraction model.
- Most OS platforms use some space at the beginning of a disk/LUN to store boot information and disk label information. This information must not be overwritten by any overlap of the first partition.
- Although not recommended, most OS platforms permit the use of multiple partitions on a single disk/LUN. Each partition must be properly aligned.

Using Multiple Partitions on a Single Disk or LUN

Using multiple partitions per disk/LUN provides no particular benefits for partition alignment. If possible, letting the application or file system use the whole disk simplifies partition alignment. For instance, in the ZFS file system, you can select a whole disk to add to a ZFS pool and the ZFS file system will handle the NBS alignment within the ZFS layer.

There are, however, a number of reasons to use multiple partitions per disk/LUN:

- In most cases, the computer hardware's boot process dictates the use of a partition for access to the boot information of the OS to be started.
- The OS requires the use of more than one partition during installation.
- Different OS boot images reside on the same disk/LUN.

- The OS boot file system may need to be separate from the application file system. One reason for this might be backup-restore design requirements. The OS partition is less likely to change than the application partition, so having separate partitions allows for different backup schedules for each of them.

  Multiple partitions also help to reduce the restore time for getting the machine up. Once the machine is up, the rest of the backups can be restored according to the priority of the Recovery Time Objective (RTO) requirement of each application environment.

Considerations for Preparing and Partitioning the Disk

Understand your options. Can you choose your disk partition/label tools, and do you have the option to use them during the installation process? If you do not, determine whether you can use a disk that already contains a partition layout. If you do not have these options, you are out of luck in creating a boot/OS environment that uses aligned partitions unless the OS properly does so by default, as with Windows 2008, for example.

Also, determine if it is important to have an aligned root file system. If you can keep the root and application data on separate partitions, in most cases it is enough to only align the partitions that contain the application data.

If you do have a choice of using a disk label/partition tool during installation – for example, the **parted** tool during Oracle Solaris live image installation – use a tool that allows alignment by sector. This means you can specify a partition start and end using a sector number.

If possible, use a single partition. This will keep the calculations for the start of the partition simple, and you only have to do the calculation once. A single partition reduces unused (wasted) disk space between partitions.

If you have the option to use prepartitioned drives in the installation process, use a generic boot image of your favorite OS with the tools you are familiar with, partition the drive(s) according to your requirements, and then start the OS installation process. A good swiss army knife-type tool kit for x86 platforms can be found at http://www.ultimatebootcd.com/ or http://www.sysresccd.org/Main_Page .

Be aware that for a boot disk, the first range of sectors is reserved for label and boot information. It is generally safe to use sector 64 for the first partition, but check your product information.

Once an OS environment is running and new disks/LUNs need to be added, use a partition tool that can operate in sector mode, like the GNU tool **parted**.

To make sure that file system blocks are aligned with the underlying disk/LUN block structure, use a file system block size that is equal to or a power of 2 of the size of the disk/LUN’s NBS. In some cases, like LUNs presented by a Sun ZFS Storage Appliance, you can define the disk/LUN block size when you create the LUN. For OLTP-type workloads, keep the size of a block on a LUN equal to the size of the blocks used by the file system/application using the disk/LUN.

Aligning Storage Subsystem LUNs Compared to Direct Attached Disks

As previously mentioned, an operating system’s assigned disk space can either be provided by storage subsystems or by using directly attached disks. When dealing with direct-attached disks, the OS directly interfaces with the disk firmware and its presentation of the disk's geometry characteristics.

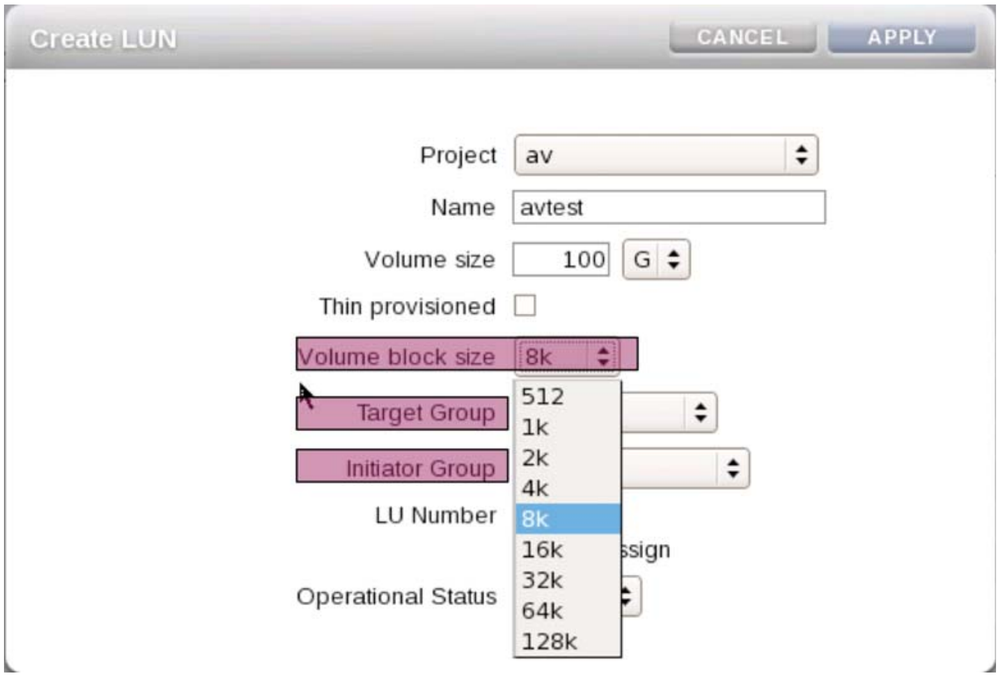
However, when using LUNs from a storage subsystem, there is an extra abstraction layer used by the storage subsystem software that employs RAID-type data protection features to provide protection against disk failures. The capacity of LUNs served out of an array will be spread over several physical disks. The array software takes care of the proper alignment between the LUN logical blocks and the underlying physical disk blocks. In the case of the Sun ZFS Storage Appliance, Oracle Solaris and ZFS direct the internal disk storage and capacity management, ensuring a proper alignment of the blocks between each layer in the storage software stack.

Note: Particular references to ZFS and Oracle Solaris in the following sections refer to clients running the Oracle Solaris OS with a ZFS file system and using the LUNs from the Sun ZFS Storage Appliance. These references do not apply to the Sun ZFS Storage Appliance itself.

Preparing to Use a LUN in the Sun ZFS Storage Appliance

With the Sun ZFS Storage Appliance, you can make block storage volumes available to clients by using either the Fibre Channel or iSCSI protocol.

In both cases a volume is created from a storage pool using the following dialog window in the Sun ZFS Storage Appliance browser user interface (BUI).



Project: av
Name: avtest
Volume size: 100 G
Thin provisioned: ☐
Volume block size: 8k
Target Group
Initiator Group
LU Number
Operational Status

Figure 4. Creating a LUN in the Sun ZFS Storage Appliance

Carefully select the block size of the LUN because, once the LUN is created, the block size cannot be changed. The block size should match the performance requirements of the application(s) that will use the LUN.

When opting for the use of disk partitions, consider the now familiar LUN partition alignment precautions.

As part of defining access restrictions, a LUN is allocated to an FC or iSCSI target group and an initiator group. Only hosts (HBAs) that are members of the initiator group are allowed access to this LUN.

Sun ZFS Storage Appliance in an FC Environment

When using LUNs from a Sun ZFS Storage Appliance, use more than one physical path to a LUN. In case of FC connections, recommended practice is to use a two independent SAN fabrics topology and a Sun ZFS Storage Appliance clustered configuration, as seen in the following figure.

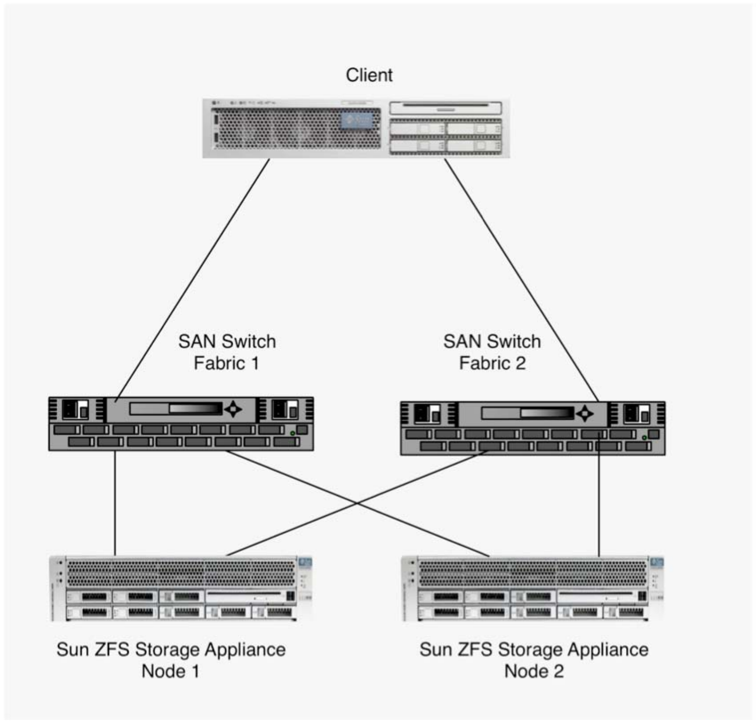

Client
SAN Switch Fabric 1
SAN Switch Fabric 2
Sun ZFS Storage Appliance Node 1
Sun ZFS Storage Appliance Node 2

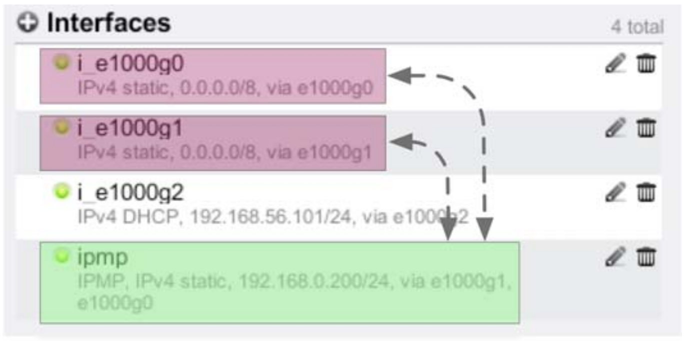
Figure 5. Redundant SAN configuration for Sun ZFS Storage Appliance

This optimal redundant SAN configuration provides protection against any component failure in the Fibre Channel path. Such a failure will trigger a path failover of a LUN within a Sun ZFS Storage Appliance node.

A Sun ZFS Storage Appliance node failure will cause the remaining Sun ZFS Storage Appliance node to regain control over the storage from the failed node and serve the LUN from its surviving node. With this configuration, correct zoning in the fabrics and proper target and initiator groups in the Sun ZFS Storage Appliance nodes must be set up to avoid the client FC HBAs from seeing each other. More detailed info about the use of Fibre Channel on the Sun ZFS Storage Appliance can be found in the document “Understanding the Use of Fibre Channel in the Sun ZFS Storage Appliance.” (See the Appendix for location.)

Sun ZFS Storage Appliance in an iSCSI Environment

When using the iSCSI protocol to provide LUNs to a host environment, you can use the Sun ZFS Storage Appliance IP MultiPathing (IPMP) functionality to create path redundancy in the networking infrastructure.



Interfaces 4 total

i_e1000g0
IPv4 static, 0.0.0.0/8, via e1000g0

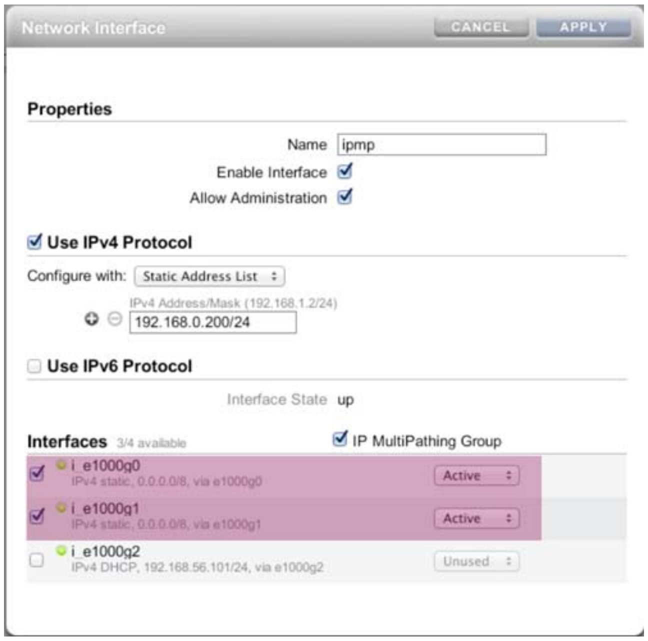
i_e1000g1
IPv4 static, 0.0.0.0/8, via e1000g1

i_e1000g2
IPv4 DHCP, 192.168.56.101/24, via e1000g2

ipmp
IPMP, IPv4 static, 192.168.0.200/24, via e1000g1, e1000g0

Figure 6. LAN IPMP group in the Sun ZFS Storage Appliance BUI

An IPMP group contains a logical IP address that is allocated to an active physical port in the group. If the connection to the network through that port is lost, the IP traffic and its IP address are automatically routed to another port in the group. Any number of IP ports can be allocated to an IPMP group as long as they are on the same subnet (LAN, InfiniBand [IB] partition, or VLAN).



Properties

Name: ipmp
Enable Interface: ✓
Allow Administration: ✓

✓ Use IPv4 Protocol

Configure with: Static Address List
IPv4 Address/Mask (192.168.1.2/24): 192.168.0.200/24

☐ Use IPv6 Protocol

Interface State: up

Interfaces 3/4 available — ✓ IP MultiPathing Group

✓	i_e1000g0 IPv4 static, 0.0.0.0/8, via e1000g0	Active
✓	i_e1000g1 IPv4 static, 0.0.0.0/8, via e1000g1	Active
☐	i_e1000g2 IPv4 DHCP, 192.168.56.101/24, via e1000g2	Unused

Figure 7. LAN IPMP group setup in the Sun ZFS Storage Appliance BUI

See the Sun ZFS Storage Appliance documentation (referenced in the Appendix) for more information on the use of IPMP. Before creating LUNs, make sure the iSCSI initiator and target groups are properly configured.

Managing Disks in Oracle Solaris

The Oracle Solaris operating system is supported on both SPARC- and x86 processor-based systems. It uses the format utility to label disks and create slices on the disks/LUNs. As previously mentioned, Oracle Solaris supports two types of disk labels:

- SMI – The traditional Oracle Solaris Volume Table of Contents (VTOC) label. Note that on x86 platforms the SMI disk label is created within an MBR fdisk partition. For Oracle Solaris x86-based systems to be able to boot Oracle Solaris from a disk, the disk must first be partitioned using an fdisk-type utility. During the Oracle Solaris installation process, an fdisk-type MBR partition is created before an Oracle Solaris root file system is installed on an Oracle Solaris SMI slice within the MBR-type partition.
- EFI – Newer disk labeling to overcome the 2TB size limit of the fdisk/SMI label structure. For disks over 2TB, a 64-bit Oracle Solaris kernel is needed. EFI may also be referred to as a GPT partition.

To determine which disk label schema is best suited for your situation, consider the following characteristics as well as restrictions for EFI:

- EFI disk label schema provides support for disks greater than 2TB in size.
- EFI disk label schema provides usable slices 0-6, where slice 2 is just another slice.
- Partitions (or slices) cannot overlap with the primary or backup label, or with any other slices. The size of the EFI label is usually 34 sectors, so slices usually start at sector 34. This feature means that no partition can start at sector zero (0).
- No cylinder, head, or sector information is stored in the EFI label. Sizes are reported in blocks.
- Some information that was stored in the alternate cylinders area is now stored in the last two cylinders of the disk or Oracle Solaris partition.
- If you use the format utility to change partition sizes, the unassigned partition tag is assigned to partitions with sizes equal to zero. By default, the format utility assigns the `usr` partition tag to any partition with a size greater than zero. You can use the partition change menu to reassign partition tags after the partitions are changed. However, you cannot change a partition with a non-zero size to the unassigned partition tag.
- Layered software products intended for systems with VTOC-labeled disks might be incapable of accessing a disk with an EFI disk label.
- Booting from a disk with an EFI disk label is supported from Oracle Solaris 11.1 onwards. It requires boot firmware on SPARC systems supporting EFI labeled disks and the Grand Unified Bootloader (GRUB) 2.1 environment on x86 platforms.
- The EFI disk label is not supported on IDE disks.

- A 64-bit Oracle Solaris kernel is needed to access disks over 2TB in size.
- On x86-based systems, you can use the Oracle Solaris `fdisk` command on a disk with an EFI label that is greater than 2TB in size.
- The EFI specification prohibits overlapping slices. The entire disk is represented by cxtydz. Slices are represented by cxtydzs*n*, in which *n* is the slice number.
- The EFI disk label provides information about disk or partition sizes in sectors and blocks, but not in cylinders and heads.
- The following format utility options are either not supported or are not applicable on disks with EFI labels:
  - The `save` option is not supported because disks with EFI labels do not need an entry in the `format.dat` file.
  - The backup option is not applicable.

In Oracle Solaris release 11.1 and later, EFI is the default disk label during the installation process on platforms that support booting from EFI-labeled disks. On x86-based systems, the boot management utility GRUB has been adapted to support booting Oracle Solaris from EFI-labeled disk partitions. Also new is that throughout the I/O stack and the Oracle Solaris disk utilities, great effort is made to detect the disk/LUN NBS and create partitions properly aligned to the disks/LUNs' NBS size. Not all storage vendors report the NBS correctly. In those cases, the utilities fdisk and format will use the standard defaults as described in the following examples.

What does this all mean for partition alignment when using Oracle Solaris? This paper focuses on LUNs presented to Oracle Solaris from the Sun ZFS Storage Appliance using either iSCSI or Fibre Channel. In most cases, the Oracle Solaris system partitions are located on local disks. Those disks, in most cases, still use a sector size of 512B, for which the alignment issue is not applicable. For situations in which partition alignment becomes an issue, such as when using SSDs as boot devices, the following described practices for achieving partition/slice alignment are equally applicable to the disk used for the Oracle Solaris root file system. You need to be familiar with the Oracle Solaris installation process and where to intervene with the standard installation procedure to customize the partition and/or Oracle Solaris slices' layout. This topic, however, is outside the scope of this paper.

If you have the option of using separate disks for application and application dataset storage, you should opt for that. The following sections describe the partition and/or Oracle Solaris slice issues for x86 and SPARC platforms for both SMI and EFI disk label schemas.

Oracle Solaris on x86-Based Systems

The disk partitioning structure on a PC-type architecture has a dependency on the BIOS firmware that is used on that architecture. During the power-up phase, the BIOS controls the disk I/O, hence the BIOS must recognize the partitioning structure to be able to load the Master Boot

Record (MBR) and, in turn, the MBR to load the OS boot start software. For this reason, the MBR DOS-type partition structure must be present for bootable disks.

Oracle Solaris uses its own disk label structure. From Oracle Solaris 11.1 release onwards, the GNU GRUB version 2.1 supports booting from EFI (GPT) labeled disks. The Oracle Solaris format utility creates a slice structure within an MBR-type partition.

Using SMI Disk Labels

When using SMI disk labels, consider two levels of alignment: the start of the FDISK/MBR partition and the start of the SMI slice. Both the Oracle Solaris fdisk and format utilities operate in CHS addressing mode. Each of the tools uses cylinder 1 as the start cylinder, reserving cylinder 0 for boot and OS use. The format utility starts numbering the cylinders from the start of the FDISK partition. Note that if more than one FDISK partition is used, each SMI VTOC starts with cylinder 0 within its FDISK partition.

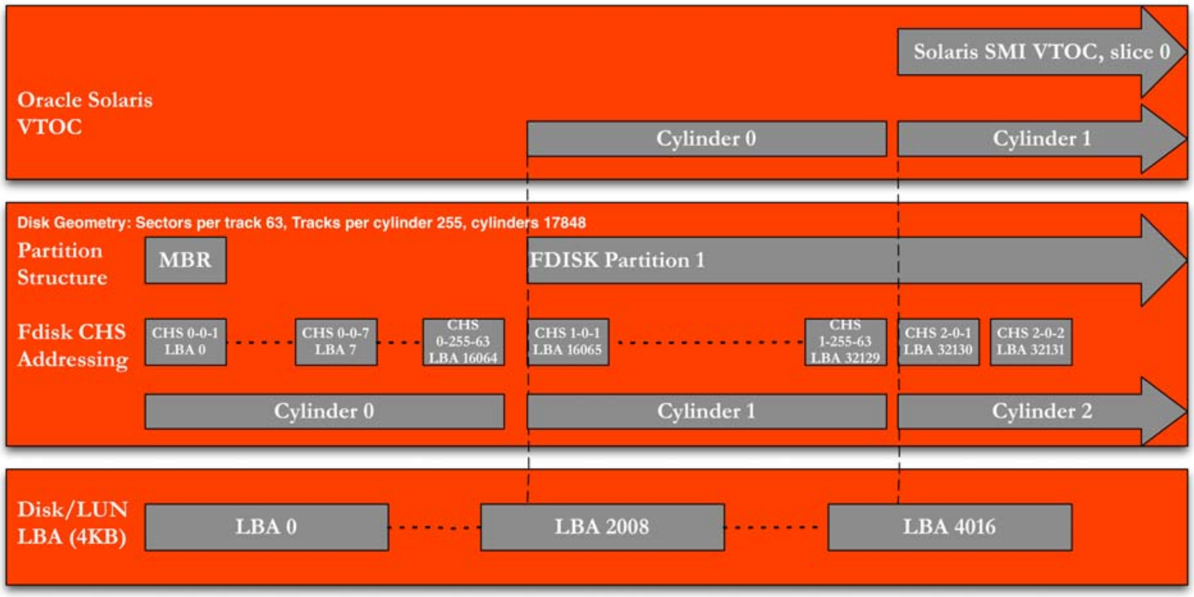


Figure 8. Default fdisk and SMI VTOC layout for Oracle Solaris on an x86 system

Check the layout shown in the previous illustration using the Oracle Solaris `fdisk` and `format` commands. Use the Oracle Solaris command `fdisk -W - <rawdisk>` to check the disk partition layout.

```
root@aie-linlithgow:~#fdisk -W - /dev/rdsk/c6t0d0p0
* /dev/rdsk/c6t0d0p0 default fdisk table
* Dimensions:
*    512 bytes/sector
*     63 sectors/track *    255 tracks/cylinder
*   17848 cylinders
*
* systid:
```

```
*    1: DOSOS12
*    2: PCIXOS
*    4: DOSOS16
…
*  191: SUNIXOS2
*  238: EFI_PMBR
*  239: EFI_FS
*
*Id    Act  Bhead  Bsect  Bcyl    Ehead  Esect  Ecyl    Rsect      Numsect
 191   128  0      1      1       254    63     1023    16065      286712055
 0     0    0      0      0       0      0      0       0          0
 0     0    0      0      0       0      0      0       0          0
 0     0    0      0      0       0      0      0       0          0
```

From the previous table, note that there is one partition created on disk c6t0d0 starting on CHS 1-0-1, Bcyl = 1, Bhead=0, Bsect=1.

Using the Oracle Solaris `format` command, you can check the layout of the Oracle Solaris SMI slice structure within the FDISK partition:

```
root@aie-linlithgow:~# format /dev/rdsk/c6t0d0s2
selecting /dev/rdsk/c6t0d0s2
[disk formatted]
/dev/dsk/c6t0d0s0 is part of active ZFS pool rpool. Please see zpool(1M).
/dev/dsk/c6t0d0s3 is part of active ZFS pool homes. Please see zpool(1M).

FORMAT MENU:
        disk       - select a disk
        …
        quit
format> partition
PARTITION MENU:
        0      - change `0' partition
        …
        print  - display the current table
        …
partition> print
Current partition table (original):
Total disk cylinders available: 17845 + 2 (reserved cylinders)

Part      Tag    Flag     Cylinders         Size            Blocks
  0       root    wm       1 -  5312       40.69GB    (5312/0/0)   85337280
  1 unassigned    wm       0                0         (0/0/0)             0
  2     backup    wu       0 - 17844      136.70GB    (17845/0/0) 286679925
  3        usr    wm    5313 - 17844       96.00GB    (12532/0/0) 201326580
  4 unassigned    wm       0                0         (0/0/0)             0
  5 unassigned    wm       0                0         (0/0/0)             0
  6 unassigned    wm       0                0         (0/0/0)             0
  7 unassigned    wm       0                0         (0/0/0)             0
  8       boot    wu       0 -     0        7.84MB    (1/0/0)         16065
  9 unassigned    wm       0                0         (0/0/0)             0

partition> quit
```

There is no option to change the number of sectors per track and the number of tracks per cylinder. This makes alignment of the start of an SMI slice with a LUN block difficult. For a slice to be aligned with the LUN disk block structure, the following must be true:

*(partition start cylinder + slice start cylinder) * sectors per cylinder * sector size) % LUN blocksize == 0*

In the preceding example, where the default Oracle Solaris partition and slice layout are used, the following slice start cylinders calculated for a number of LUN LBA block sizes would result:

TABLE 1. ALIGNED SMI PARTITIONS FOR VARIOUS NBS VALUES

SUN ZFS STORAGE APPLIANCE LUN BLOCK SIZE	PARTITION START CYLINDER	SLICE START CYLINDER	ACTUAL START CYLINDER
4 KB	1	7	8
8 KB	1	15	16
16 KB	1	31	32
32 KB	1	63	64
64 KB	1	127	128
128 KB	1	255	256

Using EFI Disk Labels

Oracle Solaris EFI type disk labels are created with the `format -e` command. This creates a slice structure that uses the whole disk. Slices' starts and ends are expressed in 512B-sized sectors, starting with sector 0. The default start of the first slice is on sector 34. The following illustration shows that slice 0 and consequently the file system on top of it will not be aligned with the start of a block of the underlying LUN volume.

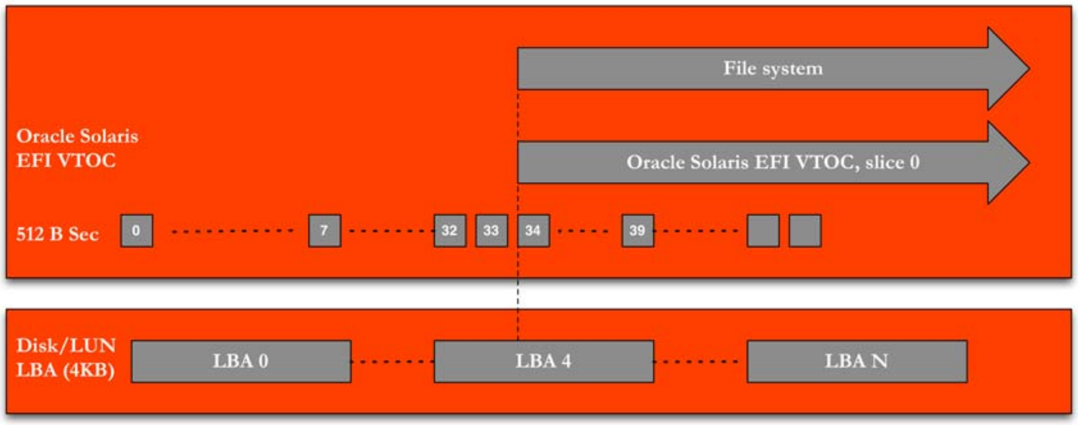


Figure 9. Default EFI partition layout in Oracle Solaris

By using a block size of 4K for the ZFS file system, each I/O to the LUN will be spread over two blocks of the LUN volume. The `partition` option in the `format -e` command offsets the option to change the start sector of a slice. By using an offset value of 256, the slice will be aligned for any power of 2 block size of the LUN up to 128K.

Warning: This is a **destructive operation**. Any data stored on a partition whose layout will be changed will be **LOST**.

Note that if a disk is added directly to a ZFS pool using the zpool utility, the utility will create a partition starting at sector 256. This will automatically create a properly aligned partition for the ZFS file systems on top of the created partition.

```
ymachine: ~# format -e
format>label
[0] SMI Label
[1] EFI Label
Specify Label type[1]: 1
Ready to label disk, continue? Y

format> partition
       ~~~~~~~~~~~~~~
partition> print
Current partition table (original):
Total disk sectors available: 209698782 + 16384 (reserved sectors)

Part      Tag    Flag    First Sector         Size           Last Sector
  0        usr    wm           34           99.00GB         207618081
  1 unassigned    wm            0              0                    0
  2 unassigned    wm            0              0                    0
  3 unassigned    wm            0              0                    0
  4 unassigned    wm            0              0                    0
  5 unassigned    wm            0              0                    0
  6 unassigned    wm            0              0                    0
  7 unassigned    wm            0              0                    0
  8   reserved    wm    209698783            8.00MB         209715166

partition> 0
Part      Tag    Flag     First Sector         Size         Last Sector
  0        usr    wm                34       99.00GB          207618081
Enter partition id tag[usr]:
Enter partition permission flags[wm]:
Enter new starting Sector[34]: 256
Enter partition size[207618048b, 207618303e, 101376mb, 99gb, 0tb]:209698782e
partition> print
Current partition table (unnamed):
Total disk sectors available: 209698782 + 16384 (reserved sectors)

Part      Tag    Flag     First Sector      Size           Last Sector
  0        usr    wm         256          99.00GB         207618303
  1 unassigned    wm           0             0                    0
  2 unassigned    wm           0             0                    0
  3 unassigned    wm           0             0                    0
  4 unassigned    wm           0             0                    0
  5 unassigned    wm           0             0                    0
  6 unassigned    wm           0             0                    0
  7 unassigned    wm           0             0                    0
  8   reserved    wm   209698783           8.00MB         209715166
partition>label
0] SMI Label
[1] EFI Label
Specify Label type[1]: 1
Ready to label disk, continue? Y
partition> quit
label>
Mymachine: ~#
```

When using the `zpool create/add` command, the full LUN will be used with a default start sector of 256. So EFI slices created using `zpool create/add` will always be properly aligned for any power of 2 block sizes of the LUN up to 128KB.

LUNs labeled this way on x86 platforms will replace any MBR-type labels on the LUN if they were present, and will subsequently lose all MBR-type partitions and data.

Oracle Solaris on SPARC

Disks labeled for Oracle Solaris on SPARC will have a slice structure directly written to the disk, so there are no MBR partition alignment issues as there are with the x86-based platforms. Only the SMI disk label schema is currently supported for boot disks.

Using SMI Disk Labels

As previously noted, the SMI labeling schema uses the CHS addressing method. This means that the slices will always start at a cylinder boundary. A disk/LUN labeled with an SMI label and the default slice layout will start on cylinder 0. This means that slice 0 will always be correctly aligned. The default slice layout creates little space in the root slice and also assigns space to slices 1 (`swap`) and 6 (`usr`). The extra slices are not aligned to the underlying LUN block structure.

Using `format -> verify` on a 10GB LUN from a Sun ZFS Storage Appliance provides the following information:

```
format> verify
Primary label contents:

Volume name = <        >
ascii name  = <SUN-Sun Storage 7210-1.0 cyl 323 alt 2 hd 254 sec 254>
pcyl        =  325
ncyl        =  323
acyl        =    2
nhead       =  254
nsect       =  254
Part      Tag    Flag     Cylinders       Size            Blocks
 0       root    wm       0 -   4      157.51MB    (5/0/0)     322580
 1       swap    wu       5 -   9      157.51MB    (5/0/0)     322580
 2     backup    wu       0 - 322        9.94GB    (323/0/0) 20838668
 3 unassigned    wm       0              0         (0/0/0)          0
 4 unassigned    wm       0              0         (0/0/0)          0
 5 unassigned    wm       0              0         (0/0/0)          0
 6        usr    wm      10 - 322        9.63GB    (313/0/0) 20193508
 7 unassigned    wm       0              0         (0/0/0)          0

format>
```

From this information you can use the nsect (number of sectors per cylinder) to check if the start of a slice is aligned with the underlying LUN NBS, applying the following formula:

*(cylinder number * sectors/cylinder * sector size) % LUN blocksize == 0*

To avoid any misalignment issues, good practice is to create one large slice containing all the disk space. Use slice 0 as the free hog partition so that all unallocated disk space is collected there.

```
format> partition

PARTITION MENU:


        0      - change `0' partition
~~~~~~~~~
 !<cmd> - execute <cmd>, then return
 quit
partition> modify
Select partitioning base:
 0. Current partition table (original)
 1. All Free Hog
Choose base (enter number) [0]? 1

Part      Tag    Flag     Cylinders       Size            Blocks
 0       root    wm       0              0         (0/0/0)          0
 1       swap    wu       0              0         (0/0/0)          0
 2     backup    wu       0 - 322        9.94GB    (323/0/0) 20838668
 3 unassigned    wm       0              0         (0/0/0)          0
 4 unassigned    wm       0              0         (0/0/0)          0
 5 unassigned    wm       0              0         (0/0/0)          0
 6        usr    wm       0              0         (0/0/0)          0
 7 unassigned    wm       0              0         (0/0/0)          0

Do you wish to continue creating a new partition
table based on above table[yes]?
Free Hog partition[6]? 0
Enter size of partition '1' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '3' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '4' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '5' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '6' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '7' [0b, 0c, 0.00mb, 0.00gb]:

Part      Tag    Flag     Cylinders       Size            Blocks
 0       root    wm       0 - 322        9.94GB    (323/0/0) 20838668
 1       swap    wu       0              0         (0/0/0)          0
 2     backup    wu       0 - 322        9.94GB    (323/0/0) 20838668
 3 unassigned    wm       0              0         (0/0/0)          0
 4 unassigned    wm       0              0         (0/0/0)          0
 5 unassigned    wm       0              0         (0/0/0)          0
 6        usr    wm       0              0         (0/0/0)          0
 7 unassigned    wm       0              0         (0/0/0)          0

Okay to make this the current partition table[yes]?
Enter table name (remember quotes): "c0t600144F0F05E906C00004F200EA50001d0"

Ready to label disk, continue? Yes

partition>
```

Using EFI Disk Labels

As the EFI disk labeling schema used for Oracle Solaris SPARC is the same as for Oracle Solaris running on an x86 system, using the `format -e` command to label a disk/LUN (as described in the "Oracle Solaris on x86-Based Systems" section) produces the same default start sector of 34. Use the `format -e` command to change the start sector of slice 0. The first 34 sectors contain EFI disk label information, which is why the start sector must be sector 34 or greater.

When using the command `zpool` to add disks to a pool, ZFS will use a start sector of 256 for slice 0 and will use the whole disk.

Summary for Oracle Solaris Partition Alignment

The following table summarizes the different defaults when labeling disks on an Oracle Solaris client.

TABLE 2. SUMMARY OF DEFAULTS FOR ORACLE SOLARIS PARTITIONS

CLIENT OS	**SMI** **USE FOR BOOT DISKS/LUNS ON ORACLE SOLARIS RELEASES BEFORE VERSION 11.1** **CHS ADDRESSING**	**EFI** **USE BOOT DISKS/LUNS ON ORACLE SOLARIS 11.1 AND FOR DATA DISKS/LUNS** **LBA (SECTOR) ADDRESSING, STARTING AT SECTOR 0**
Oracle Solaris on x86	On top of PC FDISK partition. FDISK partition starts at cylinder 1. Slice 0 starts at cylinder 1 relative to the start of the FDISK partition it is created in.	The command `format -e` uses sector 34 as the default start for slice 0. ZFS `zpool` uses sector 256 as start.
Oracle Solaris on SPARC	Directly on disk, starting at cylinder 0	

Recommendations for Oracle Solaris Clients

If possible, use EFI-type labels for data disks. When manually labeling the disks/LUNs with the `format -e` command, correct the start sector of at least 256. This guarantees alignment with the underlying block structure of a Sun ZFS Storage Appliance LUN for block sizes up to 128KB. Use a start sector value of 2048 (1MB), the current accepted industry standard for space offset used when partitioning a disk/LUN.

When a SMI-type disk label is required on an Oracle Solaris x86 client, create one slice that holds the whole Oracle Solaris root file system. For that slice start cylinder, use the following formula to find the correct start cylinder value:

*(cylinder number * sectors/cylinder * sector size) % LUN blocksize == 0*

Sun ZFS Storage Appliance LUN Configuration Tips for Oracle Solaris Host OS

When FC LUNs are used in a multipathing environment, be sure to enable this multipathing in the Oracle Solaris kernel configuration using the command `stmsboot -e`.

Use the following command to list host HBAs and detected LUNs:

```
root@aie-perth:~# prtconf -vp | fgrep -i wwn
 node-wwn:  2000001b.3284b467
 port-wwn:  2100001b.3284b467
 node-wwn:  2001001b.32a4b467
 port-wwn:  2101001b.32a4b467
 node-wwn:  2000001b.32859020
 port-wwn:  2100001b.32859020
 node-wwn:  2001001b.32a59020
 port-wwn:  2101001b.32a59020
root@aie-perth:~#
```

Use the `luxadm probe` command to identify LUNs.

```
root@aie-perth:~# luxadm probe
No Network Array enclosures found in /dev/es

Found Fibre Channel device(s):
 Node WWN:200100e08bb2a1cf  Device Type:Disk device
 Logical Path:/dev/rdsk/c0t600144F0F05E906C00004F200EA50001d0s2
root@aie-perth:~#
```

For each listed device path of the detected LUN, you can check the LUN's details, including the total physical number of paths used for the LUN and the system HBA ports through which the connections to the SAN are made.

```
root@aie-perth:~# luxadm display /dev/rdsk/c0t600144F0F05E906C00004F200EA50001d0s2
DEVICE PROPERTIES for disk: /dev/rdsk/c0t600144F0F05E906C00004F200EA50001d0s2
  Vendor:               SUN
  Product ID:           Sun Storage 7210
  Revision:             1.0
  Serial Num:
  Unformatted capacity: 10240.000 Mbytes
  Read Cache:           Enabled
    Minimum prefetch:   0x0
    Maximum prefetch:   0x0
  Device Type:          Disk device
  Path(s):

  /dev/rdsk/c0t600144F0F05E906C00004F200EA50001d0s2
  /devices/scsi_vhci/ssd@g600144f0f05e906c00004f200ea50001:c,raw
   Controller           /devices/pci@0/pci@0/pci@8/pci@0/pci@a/SUNW,qlc@0/fp@0,0
    Device Address              210100e08bb2a1cf,18
    Host controller port WWN    2100001b32859020
    Class                       primary
    State                       ONLINE
   Controller           /devices/pci@0/pci@0/pci@8/pci@0/pci@a/SUNW,qlc@0,1/fp@0,0
    Device Address              210000e08b92a1cf,18
    Host controller port WWN    2101001b32a59020
    Class                       primary
    State                       ONLINE

root@aie-perth:~#
```

Managing Disks in Linux

The majority of the Linux distributions run on an x86-based platform, so they all use the 'standard' PC-type disk partition mechanism. Disk partition tools for other hardware platforms are not covered here. Linux distributions always contain an fdisk-type partition tool to partition disks. Some distributions might also carry the GNU parted tool. Linux distributions do not contain any tools to analyze disk I/O for misalignment of file systems with the underlying LUN block structure.

The Linux fdisk partitioning tool also uses cylinders to define the start and end of a partition. The cylinder start position should align with the blocks of the underlying ZFS LUN file system. The Linux `fdisk` command has the `-S` option to specify the number of sectors per track. By using a value that is a multiple of the ZFS block size, you can ensure alignment.

There is, however, a practical problem. The `fdisk -S` option does not accept values beyond 63. In most cases, 63 is the default value used. This will always cause misalignment for ZFS block sizes greater than 1KB.

A practical way forward is to specify 32 and use the `FDISK -u` option to specify the start value of a partition expressed in sectors. To cover all ZFS block sizes, specify 256 for the first partition and a multiple of 256 for all of the following partitions. Use a multiple of 256 to specify the number of sectors in the partition, with the +*n* option, where *n* is the number of sectors in the partition.

```
root@petertest /root % fdisk -u -S 32 /dev/sda
 The number of cylinders for this disk is set to 5140

Command (m for help): n
Command action
   e   extended
 p   primary partition (1-4)p
Partition number (1-4): 1
First sector (32-41942399, default 32):256
Using default value 256
Last sector, +sectors or +size{K,M,G} (256-41942399, default 41942399):
Using default value   419424399
 Command (m for help): p
 Disk /dev/sda: 21.4 GB, 21474836480 bytes
255 heads, 32 sectors/track, 5140 cylinders, total  41942400 sectors
Units = sectors of1* 512 bytes
Disk identifier: 0x000da626

 Device   Boot      Start       End      Blocks   Id  System
/dev/sda1   *         256      41942399    106080   83  Linux
 Command (m for help): w
 The partition table has been altered! Calling ioctl() to re-read partition table.
 Syncing disks.
root@petertest /root % fdisk -l -u /dev/sda
 Disk /dev/sda: 21.4 GB, 21474836480 bytes
255 heads, 32 sectors/track, 5140 cylinders, total  41942400 sectors
Units = sectors of 1 * 512 = 512 bytes
Disk identifier: 0x000da626

 Device    Boot      Start       End      Blocks   Id  System
 /dev/sda1   *         256     41942399    2097120   83  Linux root@petertest /root %
```

Note: The fdisk utility might display some warnings if the partition does not end (or start) on a cylinder boundary. Because the concept of cylinders is artificial, it is safe to ignore those warnings.

Recommendations

When adding disks to a system and a partition layout is needed, manually create a partition layout before using any other administrative procedures to add a file system or incorporate it in a Logical Volume Manager (LVM) environment. This prevents any admin tools from creating partitions with default (misaligned) partitions and denying you the option of entering the optimal partition start and end values expressed in number of sectors.

When installing a new Linux environment, try to have a system disk partitioned the way you would like it to be, either by temporarily adding the system disk in another system or by using a boot CD that contains the utilities needed to create the required partition layout.

When opting for the use of disk partitions, the now familiar LUN partition alignment precautions have to be taken into consideration.

When using LUNs from a Sun ZFS Storage Appliance, use more than one physical path to a LUN. In the case of FC connections, using a topology of two independent SAN fabrics is recommended.

Tips for Configuring a Sun ZFS Storage Appliance LUN on Linux

There is no uniform way of finding FC HBA port WWNs on Linux operating systems.

One option is to look in the `/proc` structure in `/proc/scsi/`*`adapter_type/n`*, where *`adapter_type`* is the host adapter type and *`n`* the host adapter number.

A second option is to look in the `/sys` directory structure for a directory `fc_host`.

Example:

```
[root@x4450-1 fc_host]# pwd
/sys/class/fc_host
[root@x4450-1 fc_host]# ls
host10  host9
[root@x4450-1 fc_host]# cat host9/port_name
0x2100001b320b5eb6
[root@x4450-1 fc_host]#
```

When multiple path connections are used, use the Linux `multipath` utility to display the detected paths to a LUN, as shown in the following example.

```
[root@x4450-1 fc_host]# multipath -ll
mpath1 (3600144f0f05e906c00004fe617540002) dm-6 SUN,Sun Storage 7210
size=128G features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=1 status=enabled
```

```
| `- 9:0:0:1   sdf 8:80  active ready running
`-+- policy='round-robin 0' prio=1 status=enabled
  `- 10:0:0:1  sdi 8:128 active ready running
mpath0 (3600144f0f05e906c00004fe6173d0001) dm-5 SUN,Sun Storage 7210
size=64G features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=1 status=enabled
| `- 9:0:0:0   sde 8:64  active ready running
`-+- policy='round-robin 0' prio=1 status=enabled
  `- 10:0:0:0  sdh 8:112 active ready running
[root@x4450-1 fc_host]#
```

The LUNs listed above are available as `/dev/dm-5` and `/dev/dm-6`.

Managing Disks with Oracle VM

Oracle VM offers an integrated virtualization solution portfolio that can virtualize and manage the full hardware and software stack for both Oracle and non-Oracle workloads. Oracle VM includes Oracle VM Server for x86 and Oracle VM Manager.

Oracle VM Manager provides a central place to manage Oracle VM servers and virtual machines. It runs on 64-bit Oracle Linux 5.5 or later. Typical installations consist of a single disk containing the 5.5 Linux kernel and Oracle software to manage and monitor the Oracle VM servers. The installation process for Oracle Linux follows the typical Linux installation procedure. Installation disks can be selected and partitioned during the installation process, thus providing disks with partitions that are not aligned with disks that use disk blocks larger than 512B, such as SSDs.

However, this may not be a typical installation environment. If partition alignment is important in your case, use a disk that already has the partitions correctly laid out and select the partitions to be used for the kernel installation during the Linux installation process.

The Oracle software is installed under the `/u01` directory structure. You could opt for locating it in a separate disk partition.

An Oracle VM server will use a Linux-type disk structure to store the hypervisor and kernel environment. Storage repositories can be used from a local disk or using NFS or iSCSI and FC for block devices. The use of local disks for Oracle VM storage repositories is not recommended, as this restricts the use of virtual machines on that specific server hardware.

When installing a client OS in the Oracle VM environment, follow the recommendations for the specific OS as described in the previous sections.

Managing Disks Using Oracle Automatic Storage Management

Oracle Automatic Storage Management adds an abstraction layer on top of physical disks and file systems to simplify the administration of those components. Oracle Automatic Storage Management lets administrators manage disk groups. Disk groups are used to provide protection against failure of physical disks and provide I/O load balancing without the need for any manual tuning.

Disk can be used as a whole by Oracle Automatic Storage Management or in part by using the OS disk label/partitioning mechanism.

The Oracle Automatic Storage Management configuration information can be obtained using the Oracle Enterprise Manager or using the command utility `asmcmd`. To set the correct environment variables to access this information, execute the `oraenv` command, as seen in the following example.

```
 [oracle@x4450-1 ~]$ . oraenv
ORACLE_SID = [oracle] ? +ASM
The Oracle base for ORACLE_HOME=/u01/app/oracle/product/11.2.0/grid is /u01/app/oracle
[oracle@x4450-1 ~]$ asmcmd
ASMCMD> lsdsk -k
Total_MB  Free_MB   OS_MB  Name          Failgroup     Library  Label  UDID  Product  Redund   Path
 65536    65484   65536  UCMDATA_0000  UCMDATA_0000  System                         UNKNOWN  /dev/dm-5
 131072   131019  131072  UCMFRA_0000   UCMFRA_0000   System                         UNKNOWN  /dev/dm-6
ASMCMD> quit
[oracle@x4450-1 ~]$
```

The `asmcmd` subcommand `lsdsk` lists the disk groups and the physical disks comprising them.

From this list, you know which disks or partitions are used by Oracle Automatic Storage Management. When partitions are used, use the `fdisk` command of the host OS to check if the start of the partitions are correctly aligned with the block structure of the underlying LUN.

For Oracle Solaris, use: `fdisk -W - <`*`rawdisk device name`*`>`

For Linux, use: `fdisk -lu <`*`rawdisk device name`*`>`

When an ASM disk group contains partitions on disks that are not properly aligned, you can go through a process of removing those partitions from the disk group one by one, correct the partition structure, and add the partition back in the disk group. You might need to add a temporary extra disk or disk partition to the disk group in order to remove a partition for correction.

Considerations for Alignment in Oracle Automatic Storage Management

Using Oracle Automatic Storage Management presents just one decision that affects partition alignment: whether to permit the software to use the whole disk, or for you to take over management of the disk partitions.

The advantage to the default option of permitting Oracle Automatic Storage Management to use the whole disk is that you do not have to worry about partition alignment. Oracle Automatic Storage Management will use the disk/LUN from the first block so all I/O from Oracle Automatic Storage Management will be aligned with the disk/LUN block structure.

The disadvantage is that fdisk does not recognize a disk/LUN as being in use by Oracle Automatic Storage Management. So the risk exists that an administrator can assume the disk is not in use and can write a partition structure on it, resulting in the corruption of the data volume that was created by Oracle Automatic Storage Management.

By choosing the option to designate Oracle Automatic Storage Management's use of partitions on a disk, it becomes clear when a disk is in use. The disadvantage is that you will need to ensure that the start of each partition is aligned with the block structure of the underlying disk/LUN.

Which option is best really depends on the configuration management practices in your organization and the number of system administrators in the IT department. Well-documented configuration information and administration process policies, as well as disciplined IT staff, will help to avoid actions being executed that would cause data corruption.

Managing Disks with Microsoft Windows Operating Systems

Consider the following information in managing disks/LUNs in the various Microsoft Windows operations systems.

Microsoft Windows Server 2008 and Windows Vista

In Microsoft Windows Server 2008 as well as Windows Vista operating systems, partitioning alignment is by default set to 1MB (for disks larger than 4GB). You can check the value that is used in the registry under the following reference:

HKLM\SYSTEM\CurrentControlSet\Services\VDS\Alignment

Microsoft Windows Server 2003 and Earlier

Partitions created in a Windows environment up to and including Microsoft Windows 2003 Server are by definition not aligned. The default start of the partition table is 32256 bytes. For Windows dynamic disks, use the command `dmdiag -v` to check the partition offset. For basic disks, use the `wmic` command.

Use the diskpart tool to create partitions on the created FC LUN. The following template can be used to set the partition offset, assign a drive letter, and specify a File Allocation Unit Size (`unit=<`*`XXK`*`>`).

```
Diskpart list disk select disk <DiskNumber> create partition primary align=<Offset_in_KB>
assign letter=<DriveLetter> format fs=ntfs unit=64K label="<label>" nowait
```

Use an offset value of at least 128K or multiples of that. Note that the choice of the File Allocation Unit Size depends on the application recommendation for how to use the Windows file system. It is recommended that you use a ZFS block size of the same value.

Microsoft has an updated version of the disk partition tool available for Windows 2003 (see Microsoft Support article no. 923076). With the new version of diskpart you can specify the block size to align to and the size of the partition required. See also Microsoft Support article no. 929491, '*Disk performance may be slower than expected when you use multiple disks in Windows Server 2003, in Windows XP, and in Windows 2000*'.

Managing Disks with VMware

In a WMware environment, the Virtual Machines use the Virtual Machine File System (VMFS) and their storage blocks are created within a PC fdisk partition. It is important that the created partitions are aligned with the underlying Sun ZFS Storage Appliance's ZFS structure.

Guest OS data file systems are stored in Virtual Machine DisK format (VMDK) on top of a VMFS file system, with which they need to be aligned.

In addition, guest operating systems can use Raw Device Mappings (RDM). Partition alignment with the RDM option is the same as for an OS directly using a LUN as previously decribed.

When using VMware ESX 3.0, you must use the fdisk tool to manage the partitions for ESX. From VMware ESX4.0 onwards, the VMware vSphere client is used to manage VMFS partitions. The defaults when creating VMFS partitions are different for the VMware ESX 4.0 and ESX 5.0 versions. There are two VMFS formats:

- VMFS3 **–** Still uses MBR-type partition format. In VMware ESX 4.0, the default size of the MBR is 64K. In VMware ESX 5.0, the MBR space is 1MB in size.
- VMFS5 **–** Uses GPT-type partition format and has a GPT block at the start of 1MB in size.

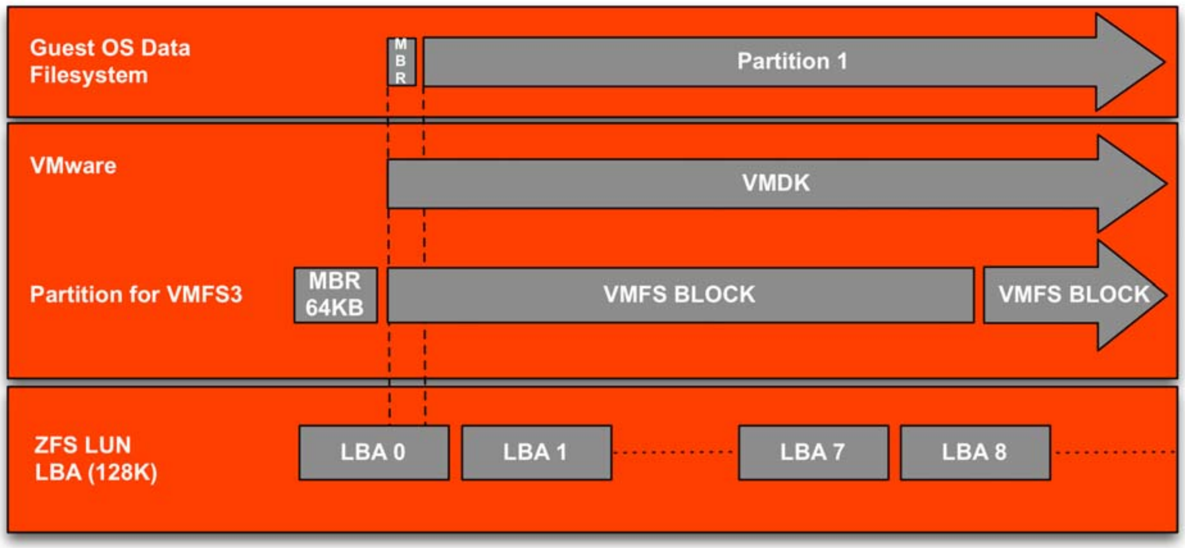


Figure 10. VMware default VMFS3 partition layout

The recommendation is to always use an MBR area size of 1MB. This will bring your system in line with the current industry standard of a 1MB reserved area and will always keep the VMFS aligned with the block structure of the Sun ZFS Storage Appliance LUNs.

As for alignment of the guest OS disks, follow the recommendations for each of the operating systems as described in the earlier sections.

Worth mentioning is that VMware provides a utility to convert physical OS instances to virtual instances, or move virtual-to-virtual instances. This tool has the option to correct the guest OS partition layout to an optimal partition layout.

VMware VMFS Block Sizes

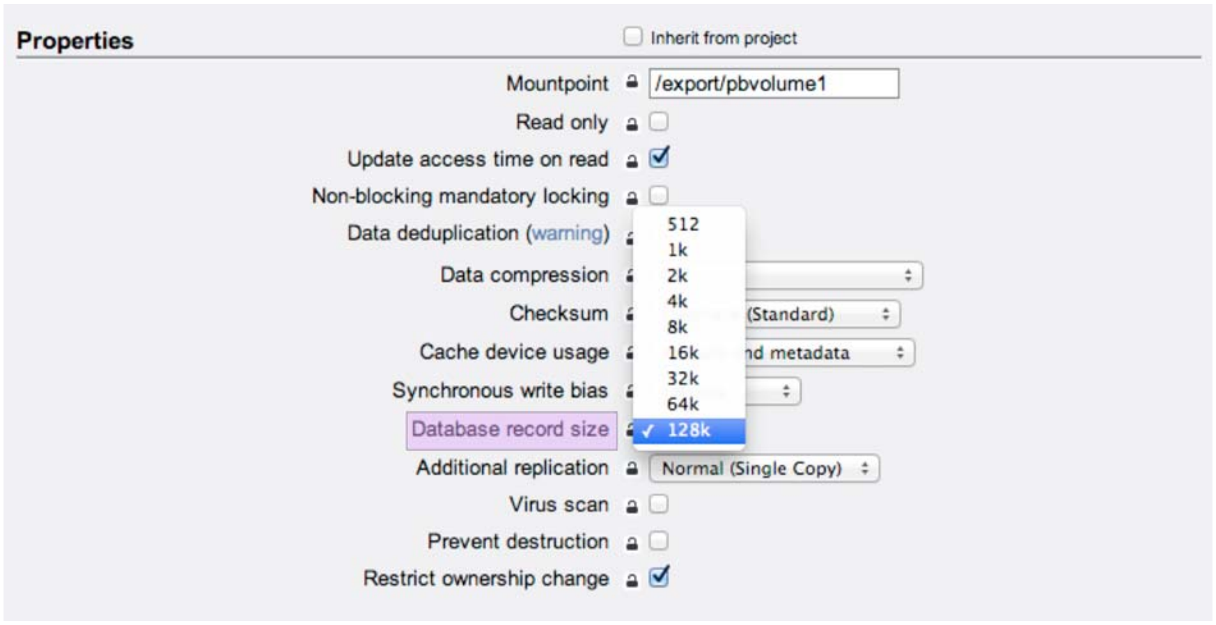
The size of the VMFS block size determines the maximum size of a VMFS instance. There is no relationship between the I/O sizes used by the guest OS (file system) and the VMFS block size. Guest OS read and write operations are transparently passed through to the external storage subsystem by the vmkernel. So the value of the used ZFS block size should be determined by the I/O size used by the guest OS file system/application.

VMware and NFS

Instead of using LUNs as storage, VMware can mount NFS file systems to be stored as VMDK files from a VM. This bypasses the VMFS layer. VMware will mount the configured NFS volume(s) from the Sun ZFS Storage Appliance. You can view VMware's mount options from the VMware vSphere client by selecting the VMware ESX host and looking under the Configurations tab ->Advanced Settings -> NFS.

The start of a VMDK file will always be aligned with a block of the volume from the Sun ZFS Storage Appliance from which the NFS file system is served. It is important to align the start of the partition(s) of the guest OS with the block structure of the Sun ZFS Storage Appliance. By using a partiton offset of 1MB for the first partition and using a start sector of any other partition that aligns with the start of the 1MB block, you are guaranteed to have aligned I/O from the guest OS to the Sun ZFS Storage Appliance.

When setting up a share on the Sun ZFS Storage Appliance, make sure you set the ZFS volume block size to a value that is optimal for your (virtual) application environment. In the Sun ZFS Storage Appliance BUI, use the property option 'Database record size’ in the share Properties window.



Properties

[ ] Inherit from project

Mountpoint	/export/pbvolume1
Read only	[ ]
Update access time on read	[✓]
Non-blocking mandatory locking	[ ]
Data deduplication (warning)	
Data compression	
Checksum	(Standard)
Cache device usage	nd metadata
Synchronous write bias	
Database record size	✓ 128k
Additional replication	Normal (Single Copy)
Virus scan	[ ]
Prevent destruction	[ ]
Restrict ownership change	[✓]

Figure 11. Setting a share record size in the Sun ZFS Storage Appliance BUI

Diagnosing Misalignment Using Sun ZFS Storage Appliance Analytics

The DTrace Analytics option in the Sun ZFS Storage Appliance is a powerful tool to analyze I/O workloads generated by various hosts connected to the Sun ZFS Storage Appliance. You can analyze workloads by network, protocols, cache, CPU and disk subsystem, with options to drill down in views per host, type of operations, and so on.

Since partition misalignment results in extra I/O operations issued to the storage subsystem, you can search for misalignment evidence by using DTrace Analytics to compare the number of I/Os issued on a protocol level (FC or iSCSI) with the number of I/Os issued to the disk subsystem. This comparison cannot be done when the host is live with an application that is actively generating a disk I/O load. As with benchmarking, you have to understand and be able to control the workload on the host in order to analyze its behavior on a storage subsystem. Some I/O will be dealt with in the cache subsystem and never reach the disk during the observation time period, while some I/O might be dealt with in the host I/O cache system. You cannot use write I/O because the type of RAID protection used influences how logical writes are broken down to physical disk I/O. Extra I/O operations are needed to write parity or mirror blocks.

The best way to investigate misalignment issues is to use an I/O benchmark tool that can generate a random read load that will not be satisfied from a cache subsystem. You must use a LUN whose size is bigger than the memory in the cache subsystem. Use a random read workload and make sure that blocks that are read have previously been written to. The ZFS file system will directly return a completion on read if there has not been anything written to a block before. If possible, use the disk raw device. Set up DTrace Analytics to monitor reads from the I/O interface (protocol) used and to the disks.

The following screenshot of DTrace Analytics output shows a Fibre Channel LUN on an Oracle Solaris host. The first display shows an EFI-labeled partition starting at sector 128 and the second result highlights a partition starting at sector 34. As the graph shows, in the first measurement period there is a one-to-one relationship between the number of I/Os from the FC LUN and the number of I/Os from the disks: 1850 IOPs.

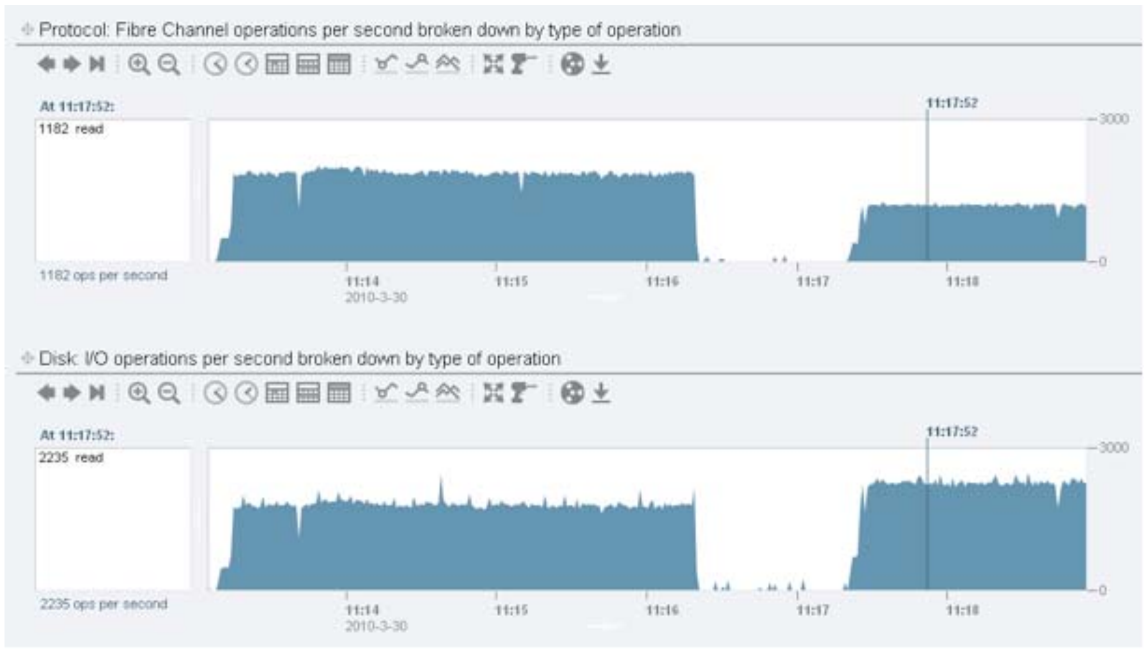


Figure 12. I/O displays for partition of EFI labeled LUN; start sectors 256 and 34 respectively

In the second period, for each read on the FC LUN, two disk read I/Os are generated. As a result, the number of IOPs per second is less than when the FC I/O blocks are aligned with the ZFS block sizes.

The example also illustrates the impact on the read performance of a partition that is misaligned: 1850 IOPs compared to 1182 IOPs result.

Conclusion

Partition alignment to the underlying disk I/O subsystem NBS blocks is critical to optimal utilization of the performance capabilities of the storage subsystem used by the host applications. Often the defaults used by tools to create disk labels and partitions are far from optimal. Understanding the block I/O stack, the placement of partitions, and the alignment relationship between partitions and natural block sizes of the underlying storage device is important for creating an optimal layout of partitions and file systems.

It is equally important to understand the characteristics of the tools your operating system of choice provides for labeling and partitioning storage devices. An example is flexibility in use of a CHS addressing mechanism compared to an LBA addressing mechanism. A tool that only supports CHS addressing makes it more difficult to create properly aligned partitions.

With this understanding and some effort, you can quickly increase the I/O performance of a storage device by 30 to 50 percent. This is a significant amount compared to other efforts, like using more disk devices per volume, or reconfiguring and spreading application volumes over storage devices – each of which involves extra investment in hardware.

Planning is key as, for example, predetermining the right natural block size setting for a LUN before creating it means you avoid making progressive changes in order to create an optimal application volume or OS file system layout – changes that most likely require the destruction of the current volume, file systems, and/or partitions.

Appendix: References

- *Sun ZFS Storage 7000 System Administration Guide*
  Part No. 820-4167-1x
  http://docs.oracle.com/cd/E26765_01/index.html
- The *Sun ZFS Storage Appliance Administration Guide* is also available through the Sun ZFS Storage Appliance help context.
  The Help function in Sun ZFS Storage Appliance can be accessed through the browser user interface.
- Sun ZFS Storage Appliance White Papers and Subject-Specific Resources
  http://www.oracle.com/technetwork/server-storage/sun-unified-storage/documentation/index.html
- Sun ZFS Storage Appliance Product Information
  http://www.oracle.com/us/products/servers-storage/storage/nas/overview/index.html
- *Oracle Solaris System Administration Guide*
  http://download.oracle.com/docs/cd/E19963-01/html
- Unified Extensible File Interface (UEFI) information
  http://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface
- GUID Partition Table information
  http://en.wikipedia.org/wiki/GUID_Partition_Table
- Master Boot Record information
  http://en.wikipedia.org/wiki/Master_boot_record
- Public domain disk utilities example (See section Partition Management),
  http://ubcd.sourceforge.net/index.html
- Microsoft Support Articles
  http://support.microsoft.com

Aligning Partitions to Maximize Storage Performance
November 2012, Version 1.0
Author: Peter Brouwer

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