SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – I – VIRTUALIZATION TECHNIQUES SCSA7022

UNIT 1 OVERVIEW OF VIRTUALIZATION

Basics of Virtualization - Virtualization Types - Desktop Virtualization - Network Virtualization - Server and Machine Virtualization - Storage Virtualization - System-level or Operating Virtualization - Application Virtualization - Virtualization Advantages - Virtual Machine Basics - Taxonomy of Virtual machines - Process Virtual Machines - System Virtual Machines - Hypervisor - Key Concepts
Hardware Virtualization - Virtual Hardware Overview - Sever Virtualization - Physical and Logical Partitioning - Types of Server Virtualization - Business cases for Sever Virtualization - Uses of Virtual server Consolidation - Planning for Development - Selecting server Virtualization Platform

Basics of Virtualization

Virtualization is a technology that helps us to install different Operating Systems on hardware. They are completely separated and independent from each other. Virtualization hides the physical characteristics of computing resources from their users, their applications or end users. This includes making a single physical resource (such as a server, an operating system, an application or a storage device) appear to function as multiple virtual resources. It can also include making multiple physical resources (such as storage devices or servers) appear as a single virtual resource..."

Virtualization Types

The term virtualization is widely applied to a number of concepts, some of which are described below:

- Server virtualization
- Client & desktop virtualization
- Services and application virtualization
- Network virtualization
- Storage virtualization

Desktop virtualization

This is also called as Client virtualization; this time is on the user's site where you virtualize their desktops. We change their desktops with thin clients and by utilizing the datacenter resources.

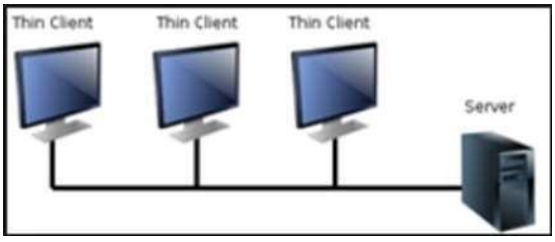


Fig No.1: Desktop Virtualization

Network virtualization

It is a part of virtualization infrastructure, which is used especially if you are going to visualize your servers. It helps you in creating multiple switching, Vlans, NAT-ing, etc.

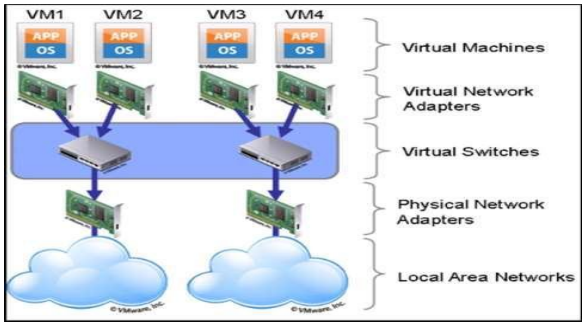


Fig No.2: Network Virtualization

Server and Machine virtualization

It is virtualizing your server infrastructure where you do not have to use any more physical servers for different purposes

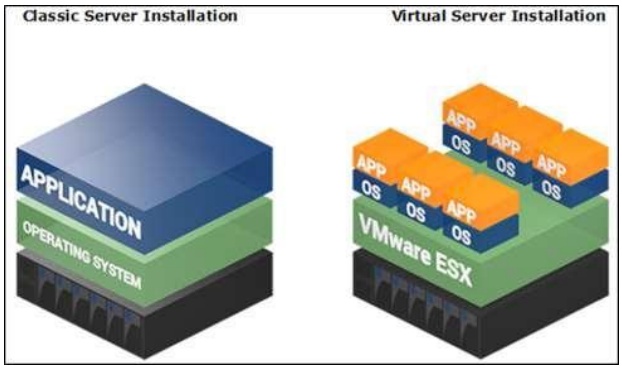


Fig No.3. Server and Machine virtualization

Storage virtualization

This is widely used in datacenters where you have a big storage and it helps you to create, delete, allocated storage to different hardware. This allocation is done through network connection. The leader on storage is SAN. A schematic illustration is given below:

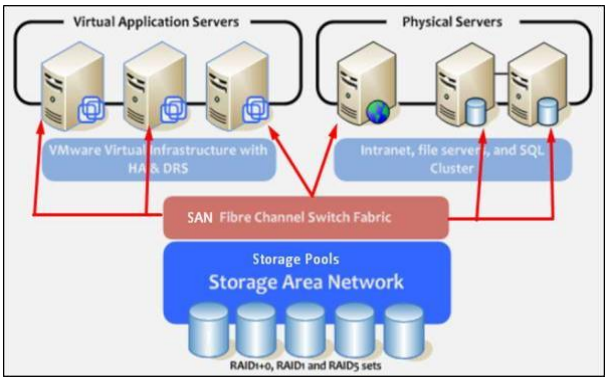


Fig No.4. Storage virtualization

Operating System Level virtualization

This refers to an abstraction layer between traditional OS and user applications. OS-level virtualization creates isolated **containers** on a single physical server and the OS instance to utilize the hardware and software in data centers. The containers behave like real servers. OS- level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users. It is also used, to a lesser extent, in consolidating server hardware by moving services on separate hosts into containers or VMs on one server.

Application virtualization

The virtualization technology isolates applications from the underlying operating system and from other applications, in order to increase compatibility and manageability. For example – Docker can be used for that purpose.

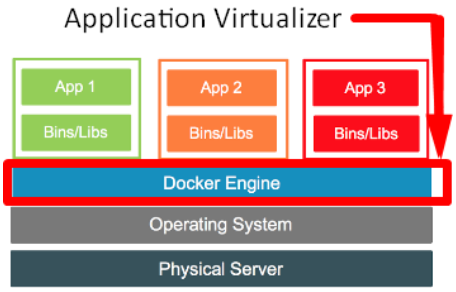


Fig No.5.Application virtualization

Advantages of virtualization
Using Virtualization for Efficient Hardware Utilization

Virtualization decreases costs by reducing the need for physical hardware systems. Virtual machines use efficient hardware, which lowers the quantities of hardware, associated maintenance costs and reduces the power along with cooling the demand. You can allocate memory, space and

CPU in just a second, making you more self-independent from hardware vendors.

Using Virtualization to Increase Availability

Virtualization platforms offer a number of advanced features that are not found on physical Servers, which increase uptime and availability. Although the vendor feature names may be different, they usually offer capabilities such as live migration, storage migration, fault tolerance, high availability and distributed resource scheduling. These technologies keep virtual machines chugging along or give them the ability to recover from unplanned outages. The ability to move a virtual machine from one server to another is perhaps one of the greatest single benefits of virtualization with far reaching uses. As the technology continues to mature to the point where it can do long-distance migrations, such as being able to move a virtual machine from one data center to another no matter the network latency involved.

Disaster Recovery

Disaster recovery is very easy when your servers are virtualized. With up-to-date snapshots of your virtual machines, you can quickly get back up and running. An organization can more easily create an affordable replication site. If a disaster strikes in the data center or server room itself, you can always move those virtual machines elsewhere into a cloud provider. Having that level of flexibility means your disaster recovery plan will be easier to enact and will have a 99% success rate.

Save Energy

Moving physical servers to virtual machines and consolidating them onto far fewer physical servers' means lowering monthly power and cooling costs in the data center. It reduces carbon footprint and helps to clean up the air we breathe. Consumers want to see companies reducing their output of pollution and taking responsibility.

Deploying Servers too fast

You can quickly clone an image, master template or existing virtual machine to get a server up and running within minutes. You do not have to fill out purchase orders, wait for shipping and receiving and then rack, stack, and cable a physical machine only to spend additional hours waiting for the operating system and applications to complete their installations. With virtual backup tools like Veeam, redeploying images will be so fast that your end users will hardly notice there was an issue.

Save Space in your Server Room or Datacenter

Imagine a simple example: you have two racks with 30 physical servers and 4 switches. By virtualizing your servers, it will help you to reduce half the space used by the physical servers. The result can be two physical servers in a rack with one switch, where each physical server holds 15 virtualized servers.

Testing and setting up Lab Environment

While you are testing or installing something on your servers and it crashes, do not panic, as there is no data loss. Just revert to a previous snapshot and you can move forward as if the mistake did not even happen. You can also isolate these testing environments from end users while still keeping them online. When you have completely done your work, deploy it in live.

Shifting all your Local Infrastructure to Cloud in a day

If you decide to shift your entire virtualized infrastructure into a cloud provider, you can do it in a day. All the hypervisors offer you tools to export your virtual servers.

Possibility to Divide Services

If you have a single server, holding different applications this can increase the possibility of the services to crash with each other and increasing the fail rate of the server. If you virtualize this server, you can put applications in separated environments from each other as we have discussed previously.

Disadvantages of Virtualization

Although you cannot find many disadvantages for virtualization, we will discuss a few prominent ones as follows:

Extra Costs

Maybe you have to invest in the virtualization software and possibly additional hardware might be required to make the virtualization possible. This depends on your existing network. Many businesses have sufficient capacity to accommodate the virtualization without requiring much cash. If you have an infrastructure that is more than five years old, you have to consider an initial renewal budget.

Software Licensing

This is becoming less of a problem as more software vendors adapt to the increased adoption of virtualization. However, it is important to check with your vendors to understand how they view software use in a virtualized environment.

Learn the new Infrastructure

Implementing and managing a virtualized environment will require IT staff with expertise in virtualization. On the user side, a typical virtual environment will operate similarly to the non-virtual environment. There are some applications that do not adapt well to the virtualized environment.

Virtual Machine Basics

Virtualization Basics. A **virtual machine** is a software **computer** that, like a physical computer, runs an operating system and applications. The hypervisor serves as a platform for running **virtual machines** and allows for the consolidation of computing resources.

Types of Virtual Machine

- System Virtual machines
- Process Virtual machines

System Virtual Machine

Hardware Virtual machine provides a complete system platform environment which supports the execution of a complete operating system (OS) VMWare, Xen, Virtual BOX.

Process Virtual Machine

Application Virtual machine provides a platform-independent programming environment that abstracts away details of the underlying hardware or operating system from software or application runtime.

Eg: Java Virtual Machine, .NET Framework

Hypervisor

A hypervisor is a thin software layer that intercepts operating system calls to the hardware. It is also called as the Virtual Machine Monitor (VMM).It creates a virtual platform on the host computer, on top of which multiple guest operating systems are executed and monitored.

Hypervisors are two types:

- Native of Bare Metal Hypervisor
- Hosted Hypervisor

Native of Bare Metal Hypervisor

Hypervisors run directly on the system hardware – A "C" embedded **hypervisor.**

Native hypervisors are software systems that run directly on the host's hardware to control the hardware and to monitor the Guest Operating Systems. The guest operating system runs on a separate level above the hypervisor. All of them have a Virtual Machine Manager.

Examples of this virtual machine architecture are Oracle VM, Microsoft Hyper-V, VMWare ESX and Xen.

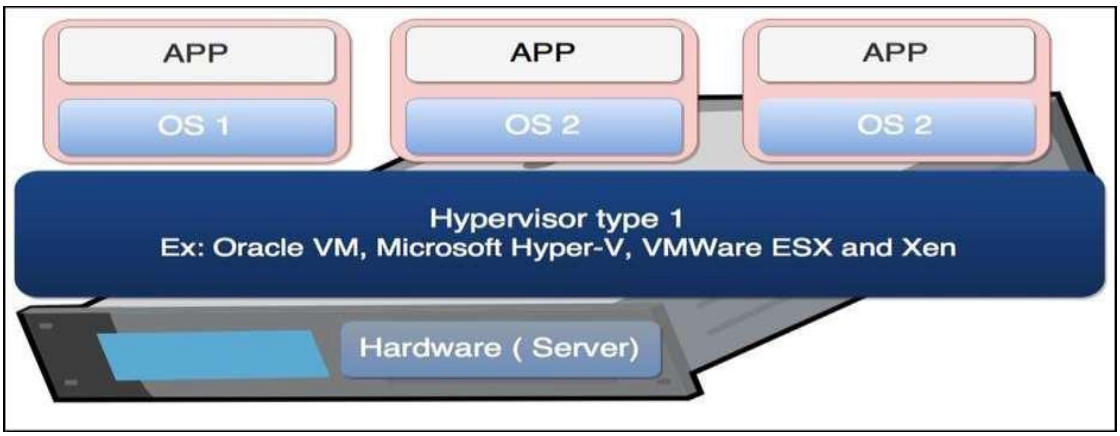


Fig No.6. Bare Metal Hypervisor

Hosted Hypervisor

Hypervisors run on a host operating system that provides virtualization services, such as I/O device support and memory management.

Example of a hosted hypervisor is Oracle VM Virtual Box. Others include VMWare Server and Workstation, Microsoft Virtual PC, KVM, QEMU and Parallels.

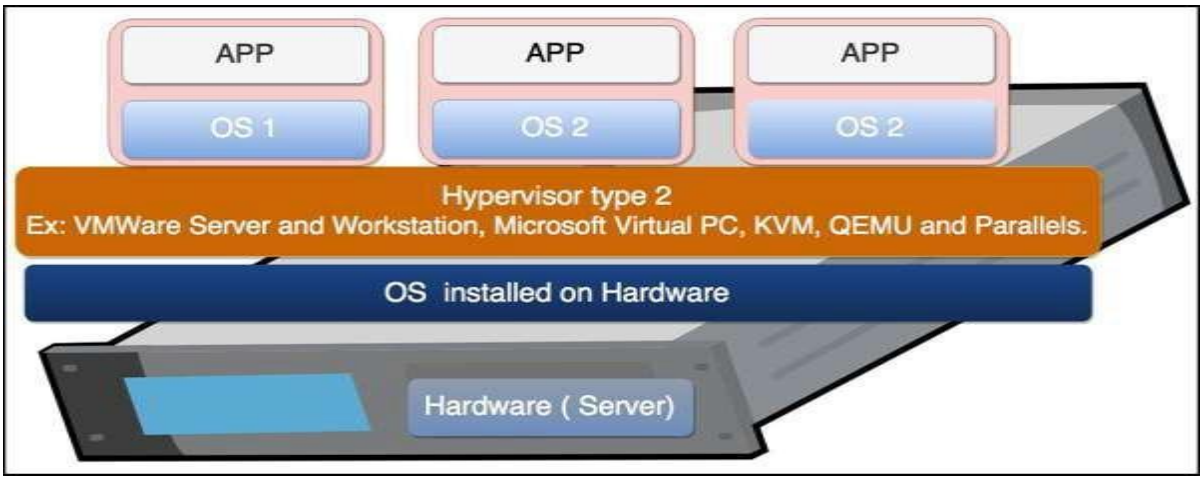


Fig No.7. Hosted Hypervisor

Hardware virtualization

Hardware virtualization is accomplished by abstracting the physical **hardware** layer by use of a hypervisor or VMM (Virtual Machine Monitor). When the virtual machine software or virtual machine manager (VMM) or hypervisor software is directly installed on the **hardware** system is known as **hardware virtualization**.

Virtual Hardware Overview

A virtual machine is a software computer that, like a physical computer, runs an operating system and applications.

The virtual machine consists of a set of specification and configuration files and is backed by the physical resources of a host. Every virtual machine has virtual devices that provide the same functionality as physical hardware, while being more portable, more secure, and easier to manage.

Virtual machines have a guest operating system on which you can install and run any software supported by that operating system. A guest operating system is an operating system that runs inside a virtual machine. You can install a guest operating system in a virtual machine and control guest operating system customization for virtual machines created from templates.

Virtualization of CPU

A VM is a duplicate of an existing computer system in which a majority of the VM instructions are executed on the host processor in native mode. Thus, unprivileged instructions of VMs run directly on the host machine for higher efficiency. The critical instructions are divided into three

 Control

 categories. sensitive instructions Behavior sensitive instructions

Privileged instructions execute in a privileged mode and will be trapped if executes outside this mode. Control sensitive instructions attempt to change the configuration of resources used. Behavior sensitive instructions have different behaviors depending on the configuration of resources, including the load and store operations over the virtual memory.

CPU's user mode while the VMM run in supervisor mode. When the privileged instructions including control and behavior sensitive instructions of a VM are executed they are trapped in the VMM. RISC CPU architectures can be naturally virtualized because all control and behavior sensitive instructions are privileged instruction.

Hardware Assisted CPU virtualization

 There are two modes to run under virtualization: root operation and non-root operation. Usually only the virtualization controlling software, called Virtual Machine Monitor (VMM), runs under root operation, while operating systems running on top of the virtual machines run under non-root operation. Software running on top of virtual machines is also called 'guest software,.

 To enter virtualization mode, the software should execute the VMXON instruction and then call the VMM software. Then VMM software can enter each virtual machine using the VMLAUNCH instruction, and exit it by using the VMRESUME. If VMM wants to shut down and exit virtualization mode, it executes the VMXOFF instruction.

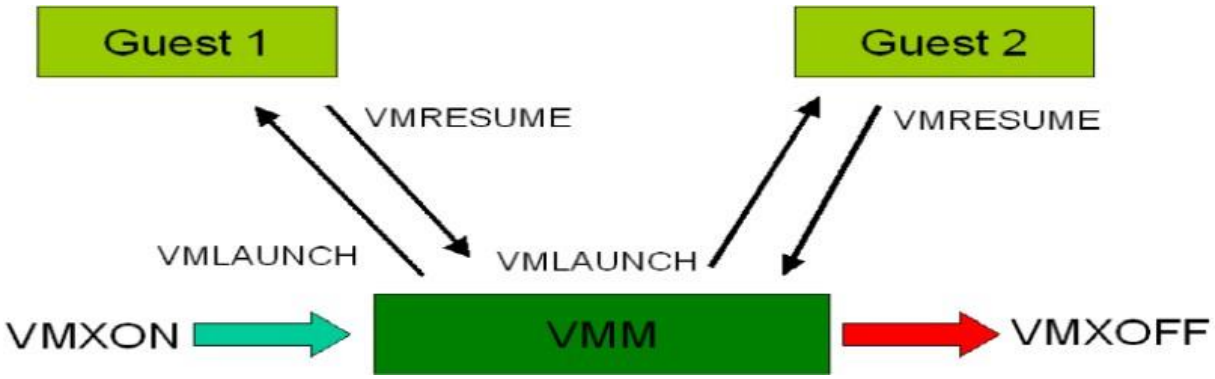


Fig No.8. Hardware Assisted CPU virtualization

Memory Virtualization

Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems. In a traditional execution environment the OS maintains mappings of virtual memory to machine memory using page tables, which is one stage mapping from virtual memory to machine memory. All modern x86 CPUs include a Memory management Unit and a translation Look-aside Buffer to optimize virtual memory performance. In virtual execution environment virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the physical memory of the VMs.

Guest OS sees flat ‚physical' address space.
Page tables within guest OS:
• Translate from virtual to physical addresses.
Second-level mapping:
• Physical addresses to machine addresses.
VMM can swap a VM's pages to disk.

Traditional way is to have the VMM maintain a shadow of the VM's page table. The shadow page table controls which pages of machine memory are assigned to a given VM. When OS updates its page table, VMM updates the shadow

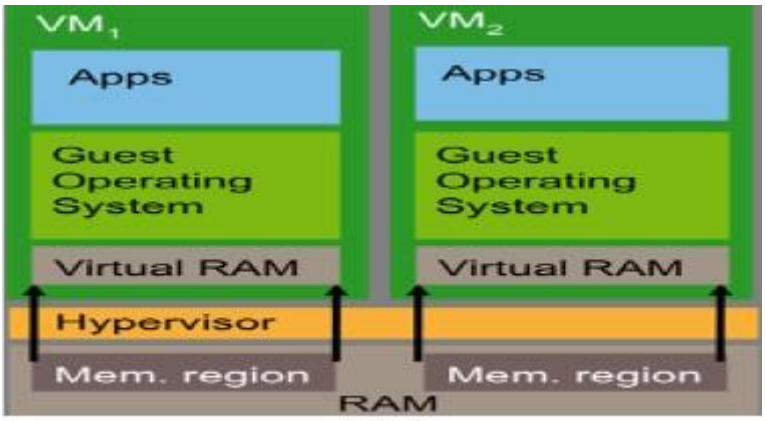


Fig No.9. Memory Virtualization

I/O Virtualization

Input/output (I/O) virtualization is a methodology to simplify management, lower costs and improve performance of servers in enterprise environments. I/O virtualization environments are created by abstracting the upper layer protocols from the physical connections.

The technology enables one physical adapter card to appear as multiple virtual network interface cards (vNICs) and virtual host bus adapters (vHBAs). Virtual NICs and HBAs function as conventional NICs and HBAs, and are designed to be compatible with existing operating systems, hypervisors, and applications. To networking resources (LANs and SANs), they appear as normal cards.

In the physical view, virtual I/O replaces a server's multiple I/O cables with a single cable that provides a shared transport for all network and storage connections. That cable (or commonly two cables for redundancy) connects to an external device, which then provides connections to the data center networks.

Server I/O is a critical component to successful and effective server deployments, particularly with virtualized servers. To accommodate multiple applications, virtualized servers demand more network bandwidth and connections to more networks and storage. According to a survey, 75% of virtualized servers require 7 or more I/O connections per device, and are likely to require more frequent I/O reconfigurations.

In virtualized data centers, I/O performance problems are caused by running numerous virtual machines (VMs) on one server. In early server virtualization implementations, the number of virtual machines per server was typically limited to six or less. But it was found that it could safely run seven or more applications per server, often using 80 percentage of total server capacity, an improvement over the average 5 to 15 percentage utilized with non-virtualized servers.

Sever virtualization

Server virtualization is a virtualization technique that involves partitioning a physical server into a number of small, virtual servers with the help of virtualization software. In server virtualization, each virtual server runs multiple operating system instances at the same time.

Physical and Logical Partitioning

Physical Partitioning

Physical partitioning refers to the separation of execution environments by literally using physically separate hardware devices or by using physical hardware-based partitioning. Physical hardware separation (see Figure 1) is the easiest and most prolific form of partitioning. It is the practice of using multiple physical servers (or computers), each having a single instance of an operating system, to serve different needs or purposes. A common example of this practice is an organization that has a separate server for each of the following server roles or applications: file sharing, print spooling, domain authentication and authorization, database server, email server, web server, FTP server, and so on.

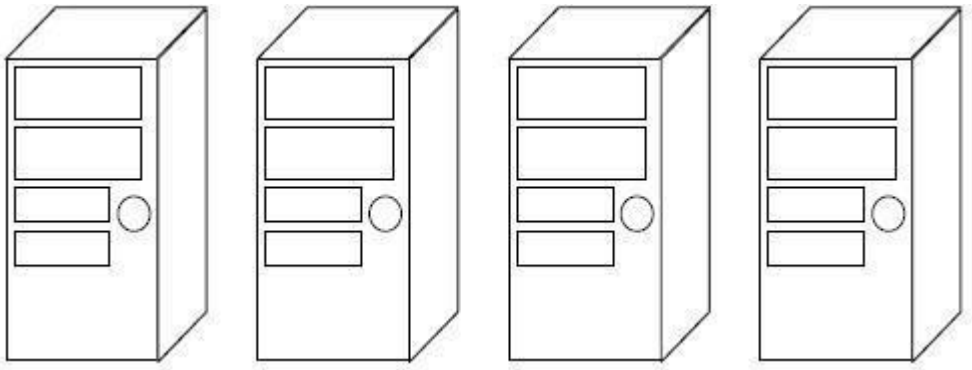


Fig No.10. Physical Partitioning

Physical Hardware Separation

Physical hardware separation is commonly driven by applications that have mutually exclusive requirements of the hardware devices or operating system, applications with high resource utilization, or server stability. Some applications cannot share the same environment as other

applications because they were designed to have control of the entire operating system and the server's resources. Other applications have high resource utilization such as processor, memory, intensive disk I/O, storage size limitations, or network adapter bandwidth that often requires a dedicated server. Consider installing Microsoft SQL Server and Microsoft Exchange Server together on a single server. Although it is technically possible, it is likely that each server application will perform poorly as they continuously compete for control of the same system resources.

Applications have also been separated onto dedicated physical servers because of the idea that it is generally more stable to have fewer applications running within the same instance of an operating system, usually because of poor resource management by the operating system (whether true or perceived) or because of poor resource handling or wasteful resource utilization by an application. Another reason that applications are installed on separate hardware is because they were designed and written for different operating systems or different hardware architectures. For example, Microsoft BizTalk Server must be installed onto a Windows operating system-based server with an Intel processor whereas applications written for IBM's OS/400 operating system must be installed on IBM AS/400 server hardware, while applications written for the Microsoft Windows operating system must be installed on IA-32 compatible computer hardware.

Hardware partitioning, shown in Figure 2 is a highly-specialized hardware technology that allows the computing resources of a single, physical computer to be divided into multiple partitions, often called hard partitions, each of which can host its own, isolated instance of an operating system. Hardware partitioning has existed for quite some time, originating in high-end mainframe systems from IBM.

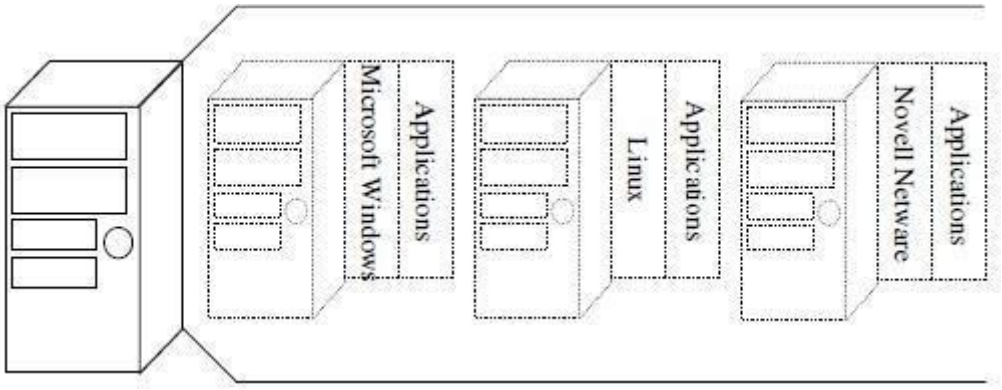


Fig No.11. Physical Hardware Separation

Hardware Partitioning

Today, there are several hardware partitioning technologies available, although each implementation is proprietary and requires very specific server hardware and software to be used. In some implementations, only one or two very specific operating systems are supported. In general, all of the required components of a system featuring hardware partitioning are only

available from a single vendor, due to their proprietary nature.

One of the key advantages of hardware partitioning is its very efficient resource sharing and management capabilities. These systems are much more efficient than equivalent software partitioning systems because the resource management between hard partitions is handled using separate hardware components (chips, circuits, memory, storage, etc.). The specialized software (sometimes referred to as microcode) that performs the actual resource management resides in the specialized resource management hardware components as well. As a result, the available performance in each hard partition is maximized and remains unaffected by the resource management system's overhead.

This is very different from software partitioning technologies where the partitioning occurs in software that is executed using the same hardware that is being managed and shared. Another advantage, available in some implementations of hardware partitioning, is electrical isolation of each hard partition. Electrical isolation in hardware partitioning systems allows a hardware fault to occur in one hard partition while not affecting any other hard partition in the same system. Systems offering hardware partitioning technologies are usually mid-range to high-end computing systems that are generally very scalable (usually scaling up) and robust.

Hardware partitioning systems have several disadvantages: expensive, proprietary hardware and software, additional costs incurred by the support and maintenance of the proprietary hardware, limited support for various operating systems, limited hardware portability for an existing installed base of hard partitions, and vendor lock-in. Proprietary hardware and software systems almost always have additional costs for installation, training, support, and maintenance due to the lack of expertise of most IT organizations with these types of systems.

Often vendors will only allow their services organization to perform the installation and support of these systems. Hardware partitioning systems generally only allow one type of operating system to be installed; of course, each hard partition supports a separate instance of that operating system. There are some systems that are more flexible and support more than one operating system, but it is almost always limited to operating systems provided by the vendor. Aside from limited operating system support, hardware partitioning systems have very limited portability of existing partitions. Generally, these partitions may only be moved to systems comprised of the same vendor's hardware because of the lack of complete hardware abstraction.

Investment in proprietary hardware and software systems almost always leads an organization into what is known as vendor lock-in. Vendor lock-in occurs when an organization has made an investment in a single vendor's proprietary technologies and it thus is cost prohibitive for the organization to move to a different technology or vendor. Vendor lock-in affects organizations for long periods of time, usually five or more years at a time. Of course, the vendor reaps the benefit of vendor lock-in because of the expense and difficulty an organization faces when attempting to switch to another vendor. The organization suffers due to cost and inflexibility in changing hardware and software, which makes it difficult to quickly move on to new opportunities.

Logical Partitioning

Logical partitioning refers to the separation of execution environments within a computing system using logic implemented through software. There are different ways in which the resources of a computer system may be managed and shared. Logical partitioning includes software partitioning, resource partitioning, and service partitioning technologies.

Software partitioning is a software-based technology that allows the resources of a single, physical computer to be divided into multiple partitions (also called soft partitions or virtual machines), each of which can host its own, isolated instance of an operating system. Software partitioning is generally similar to hardware partitioning in that multiple instances of operating systems may coexist on a single physical server. The major difference between hardware and software partitioning is that in software partitioning, the isolation of each soft partition and the management of the shared resources of the computer are completely handled by a special software layer called a Virtual Machine Monitor (VMM) or Hypervisor. The VMM, as well as each operating system within each soft partition, all consume computing resources from the same set of hardware, thus software partitioning incurs overhead that does not exist in hardware partitioning.

The overhead produced by the VMM varies from each implementation of software partitioning systems, but always has an impact on the performance of each soft partition. Depending on how resources are managed by the VMM, it is conceivable that the computing resources consumed by each soft partition could also impact the VMM's performance as well. Software partitioning implementations exist on mid-range to high-end computing systems as well as commodity server and workstation computers. Server virtualization (and the term virtualization) as described in this book refers directly to server-based software partitioning systems, generically referred to as virtualization platforms.

Software partitioning systems are generally implemented in one of two ways. They are either hosted as an application in an existing operating system installed on a physical computer or they are installed natively on a physical computer without an operating system. When software partitioning systems are hosted in an existing operating system, they gain the advantages of leveraging that operating system's resource management capabilities, its hardware compatibility, and application programming interfaces (APIs). This allows the software partitioning system to be smaller and potentially easier to write and support. This configuration also imposes the deficiencies and inefficiencies of the host operating system upon the software partitioning system as well as the additional resource consumption of the host operating system.

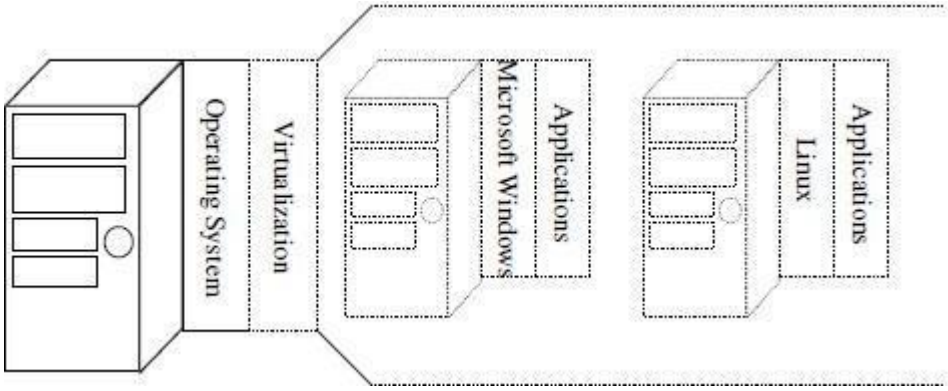


Fig No.12 Hardware Partitioning

Software Partitioning Hosted in an Existing Operating System

Hosted software partitioning systems generally have more overhead and less performance than their native counterparts. Current server-based implementations of software partitioning systems include Microsoft Virtual Server and VMware GSX Server for, Windows both of which run in a Windows Server operating system and VMware GSX Server for Linux, which runs in a Linux operating system.

Software partitioning systems installed natively onto "bare metal" (the physical computer hardware) without an operating system are generally more efficient in their management of the computer's resources. This is because the software partitioning system has full control over those resources. Although this type of implementation is more difficult to write and support, it is not burdened by theo verhead of another operating system, generally allowing more performance for each soft partition. VMware ESX Server is currently the most mature implementation of a natively installed software partitioning system for x86-based server architectures.

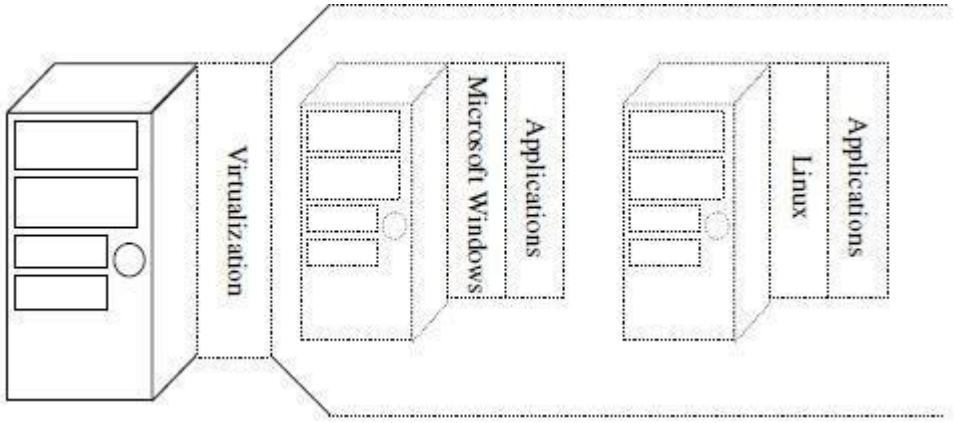


Fig No.13 Software Partitioning Hosted in an Existing Operating System

Software Partitioning Installed Natively on Hardware

A big advantage of software partitioning systems available for x86-based computers over hardware partitioning is cost. These systems run on standardized, commodity server hardware, which is much less expensive than the proprietary hardware partitioning systems. Because the hardware is also standardized across the industry, most IT organizations have the necessary skills to properly scope, deploy, configure, and administer the hardware today. This also lessens the implementation and support costs of software partitioning versus hardware partitioning. Software partitioning systems also offer the advantage of hardware portability. Each soft partition (or virtual machine) is fully abstracted from the underlying hardware that allows the partition to be moved to any physical computer that has the software partitioning system installed. An example being two computers having dramatically different hardware, such as a dual-processor, rack-mounted server and a laptop computer each running Microsoft Virtual Server. This is a powerful concept that makes server virtualization platforms very powerful in their capabilities. The benefit is also realized in terms of hardware upgrades because as long as the software partitioning system is supported on newer hardware, each soft partition will run as expected and in most cases it is trivial to move the soft partitions to another physical computer. The issue of vendor lock-in is also avoided in regard to the physical hardware since software partitioning systems for x86-based computer architectures can be installed on many different manufacturer's hardware (again due to industry standardization).

Software partitioning systems use a combination of emulation, simulation, and pass-through in their hardware abstraction methods. Each soft partition "sees" its own set of hardware resources that it may consume. This leads to an interesting question: can a software- partitioning virtualization platform be installed and used within a soft partition? Although this is theoretically possible, it is highly impractical and unusable as a solution. In some cases, depending on the specific virtualization platforms used, the virtualization platform may not even complete the installation process. In other cases, operating systems installed in a soft partition of another soft partition execute too slowly to be used effectively. This is most likely due to the multiplication of overhead within the system. For instance, when using preemptive multi-tasking operating systems for the host platform installed directly on the hardware and for the operating systems installed in each soft partition, the physical effect of time-slicing the physical CPU between all of the processes is multiplied for those processes executing within the first and second-level soft partitions. If more than one soft partition is created in the virtualization platform installed in a soft partition, the effect is worsened because the multiplier increases. It is generally a bad idea to embed entire software partitioning systems within existing soft partitions.

Application partitioning is a software-based technology that allows the operating system resources on which an application depends to be placed in an alternate container within the operating system without the application's knowledge. These resources are said to be virtualized by the application partitioning system. The isolated application can then be executed in multiple instances simultaneously in the same operating system, by one or more users, without the application

instances interfering with one another. Each instance of the application has no knowledge that the other instances exist and the application does not require any modifications to be hosted by the application partitioning system.

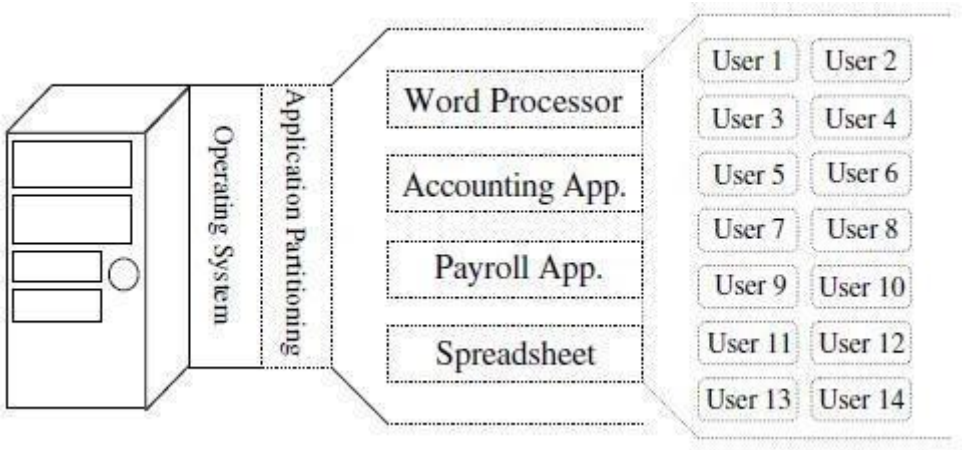


Fig No.14 Software Partitioning Installed Natively on Hardware

Application Partitioning

The primary advantage of an application partitioning system is that any application, regardless if it was designed to be used by a single user or multiple users, can be centrally managed and made available in a distributed fashion. A single server can execute many instances of the application and each application instance state is written into a separate container. Each container is automatically handled by the application partitioning system. Application partitioning can consolidate a single application from multiple desktop computers and servers onto a single server and the application can be managed much like a single instance of the application. The operating system itself is not completely abstracted from the application, only certain subcomponents such as data storage facilities (file systems), therefore only applications normally run on the operating system being used are allowed to be hosted under the application partitioning system.

Resource partitioning is a software-based technology that abstracts how certain operating system resources are allocated to application instances or individual processes executing within the operating system. This technology is used to control resource consumption of applications and processes, allowing more granular control than what is provided by the operating system. Resource partitioning systems also allow the resource consumption to be controlled not only at the application or process level, but also by the combination of application or process and user account. Resource partitioning systems enable the operating system to become Quality-of-Service enabled.

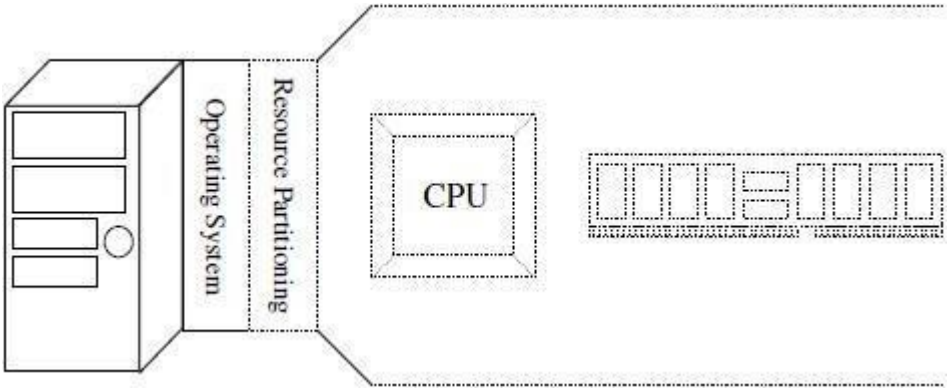


Fig No.15 Application Partitioning

Resource Partitioning

Application instances or processes can be given parameters, which allow certain minimum and maximum levels of resource utilization such as CPU, memory, and disk I/O to be effectively controlled and managed. Just as in application partitioning, resource partitioning does not abstract the entire operating system from an application, therefore only applications that would normally run on the operating system being used are allowed to be controlled by the resource partitioning system.

Service partitioning is a software-based technology in which a single application instance provides multiple, isolated instances of a service. Each instance of the service usually appears to consumers of that service to be a dedicated application instance (and often a dedicated server instance). Abstraction between the service application and the operating system is not required. The abstraction occurs on top of the application instance, which allows multiple instances of the application's service to coexist. The level of isolation between service instances can vary greatly between implementations of service partitioning, and in some systems can even be controlled, providing complete isolation at one extreme and no isolation at the other extreme of the isolation configuration settings.

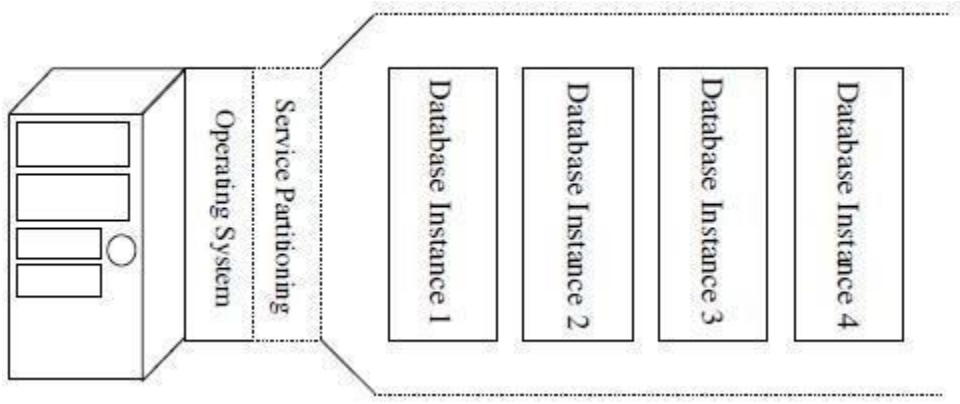


Fig No.16 Resource Partitioning

Service Partitioning

Common examples of service partitioning systems include database and web server applications. In a database server application, a single instance of the database server executes within the operating system. The primary service of the database server is to provide database access. Database servers can typically provide multiple databases per server instance. Each database can be configured to appear to a consumer to be the only database on the server, when in reality there may be 20 databases being concurrently accessed by 500 users. Most modern Web server applications allow multiple "virtual" Web sites to be created and hosted simultaneously from a single instance of the application. Each Web site is isolated from the other and, from a Web surfer's point-of-view, each Web site appears as if it is hosted on its own dedicated server, when in reality, there may be 100 Web sites running concurrently from the single Web server application instance.

Types of Sever virtualization

- Hypervisor
- Para Virtualization
- Full virtualization
- Hardware Assisted Virtualization
- Kernel level Virtualization
- System Level or OS virtualization

Business cases for Sever virtualization

Windows Server virtualization, the deployment of a virtual version of a Windows-Server operating environment, is used to reduce hardware costs, gain efficiencies, and improve the availability of computing resources. It refers to installing a virtual environment onto one or more “virtualized” hardware servers (termed Physical Hosts) and deploying multiple virtual Windows Server operating systems (termed Virtual Guests) onto this virtual environment.

In small to medium-sized businesses, we typically see three levels of Windows Server virtualization with these increasing benefits:

- Single Physical Host – Cost savings (energy and hardware) with some flexibility
- Multiple hosts with Storage Area Network (SAN) – Highly available environment with minimal downtime
- Multiple hosts with Site-to-Site Failover – Disaster recovery to separate location We review each of these levels below.

Single Physical Host

This virtualization level has these components:

- Single hardware server with onboard storage – This hardware server is the platform for the Physical Host; it could be a HP ML350/ML370 tower server or equivalent with multiple disk drives.

- Virtualizing software – The operating environment for virtualization; typically the free versions of either VMware's VSphere or Microsoft's Hyper-V. (These products are available as free downloads from the manufacturer.) Installing the virtualizing software onto the hardware server creates the Physical Host.

- Multiple Virtual Guests – The virtual operating systems installed onto the Physical Host; usually one or more instances of Microsoft's Windows Server. (These instances must each be licensed copies of Windows Server and any associated, server-based applications.)

This environment consolidates several Windows Server instances onto a single hardware server with sufficient processing capability, Random Access Memory (RAM), and on-board disk storage. It introduces cost savings in hardware, energy, and support and provides some flexibility in the transfer of a virtualized instance to a new hardware platform (although this transfer is manual and requires a second hardware server).

Primary business benefits:

- Less up-front acquisition cost (capital expenditure or CapEx) since a single hardware server can be used rather than two or more hardware servers. Plus, the virtualizing software at this level is basically free.

- Less energy required to power a single hardware server than multiple hardware servers; leads to reduced operating expenses (OpEx).

- Fewer components to support; could lead to lower support costs.

- Increased flexibility and scalability when migrating to a new hardware server.

This virtualizing environment works well in a business with a couple of Windows Servers that is looking to capital and operating reduce costs.

Multiple Physical Hosts with a Storage Area Network

At this level, we separate the storage (disk-drives) from the Physical Host and move them to a separate Storage Area Network (SAN). We also add sophisticated virtualizing software capable of automatically managing the location of Virtual Guests.

A major benefit of this approach is termed: "High availability".

High availability refers to “A system design approach and associated service implementation that ensures a prearranged level of operational performance will be met…” (from Wikipedia under “High availability”). Basically, if designed properly, this level provides complete redundancy of all critical components within the equipment stack such that any single component can fail without compromising system reliability.

Improved performance is also likely since the virtualizing software can automatically balance available resources against Virtual Guest needs.

This virtualization level has these primary hardware components:

- Storage Area Network (SAN), preferably with redundant disk chassis and network switching2
- Two or more Physical Hosts, preferably with N+1 redundancy3
- Two or more VLAN-capable Ethernet switches4 Each item is a critical of the overall design:
- All data and Virtual Guests reside on the SAN
- Virtual Guests are balanced among the Physical Hosts
- Ethernet switches route all the traffic between the SAN and the Physical Hosts

If any item fails, the system fails. So, each item must be redundant (to increase reliability) and must be properly maintained. Multiple Hosts with Site-to-Site Failover Our highest level of Windows Server virtualization, Multiple Hosts with Site-to-Site Failover, addresses the issue of a single-site failure; how long does it take to recover to a new location if your primary site fails (as in a building catastrophe such as long-term power outage, flooding, fire, theft, etc.).

Like most data-center-uptime strategies, redundancy is the core concept; in this case, a second site is equipped with comparable equipment and the data is synchronized between the primary and secondary site. Done properly, the secondary site can be brought up either automatically or, when budget is a constraint, within a short interval of an hour or less.

Configuring for automatic failover can be considerably more expensive than allowing a short interval of an hour or less to recover since you essentially need to duplicate the primary site at the remote location, have sufficient bandwidth between the locations to permit real-time replication, and deploy some additional equipment and software to manage the automatic failover.

While automatic failover is feasible, we structure the failover interval (automatic or short) to meet the client’s requirements and budget.

When configuring for automatic failover, several items must be adjusted:

- P4500 SANs must be deployed at the primary and remote site(s) and must be configured in a multi-site cluster
- VMware vSphere Enterprise or better is required and must be licensed for both the primary and

remote (recovery) site(s)

- Windows Server licensing at the primary site must be duplicated for the recovery site(s)
- Sufficient bandwidth must exist for real-time disk-writes since this configuration cannot fall behind and catch-up during slack periods
- Additional VMware utilities and enhanced licensing for applications may be required to enable true automatic failover

Uses of virtual server Consolidation

Server consolidation works on the principles of server virtualization, where one or more virtual servers reside on a physical server. Server consolidation uses a multi-tenant architecture where all the installed and hosted virtual servers share a processor, storage, memory and other I/O and network processes.

The primary objective behind server consolidation is to consume all of a server's available resources and reduce the capital and operational expenses associated with multiple servers. Traditionally, only 15-30 percent of a physical server's overall capacity is used. With server consolidation, the utilization rate can be increased to well over 80 percent. Server consolidation works on the principles of server virtualization, where one or more virtual servers reside on a physical server.

Server consolidation uses a multi-tenant architecture where all the installed and hosted virtual servers share a processor, storage, memory and other I/O and network processes. However, each virtual server has a separate operating system, applications and internal services.

Planning for Development

One of the best existing models for quality management is called PDCA or also the Deming cycle Act. Thorough Completion of all 4 Stages ensures quality of the final product.

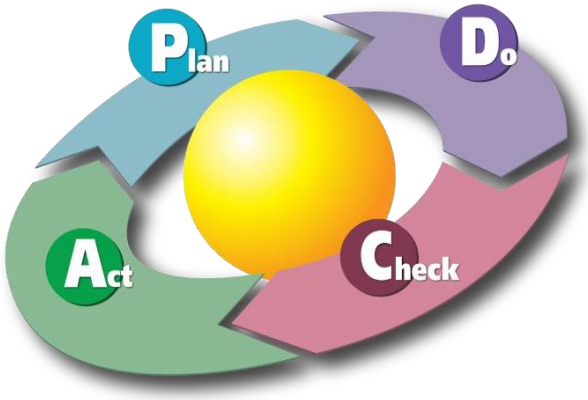


Fig No.17 Planning for Development

Plan

Establish objectives and processes required to deliver the desired results.

Do

The do phase allows the plan from the previous step to be done. Small changes are usually tested, and data is gathered to see how effective the change is.

Check

During the check phase, the data and results gathered from the do phase are evaluated. Data are compared to the expected outcomes to see any similarities and differences. The testing process is also evaluated to see if there were any changes from the original test created during the planning phase. If the data is placed in a chart it can make it easier to see any trends if the PDCA cycle is conducted multiple times. This helps to see what changes work better than others, and if said changes can be improved as well.

Example: Gap analysis, or Appraisals.

Act

Also called "Adjust", this act phase is where a process is improved. Records from the "do" and "check" phases help identify issues with the process. These issues may include problems, non-conformities, and opportunities for improvement, inefficiencies and other issues that result in outcomes that are evidently less-than-optimal. Root causes of such issues are investigated, found and eliminated by modifying the process. Risk is re-evaluated. At the end of the actions in this phase, the process has better instructions, standards or goals. Planning for the next cycle can proceed with a better base-line. Work in the next do phase should not create recurrence of the identified issues; if it does, then the action was not effective.

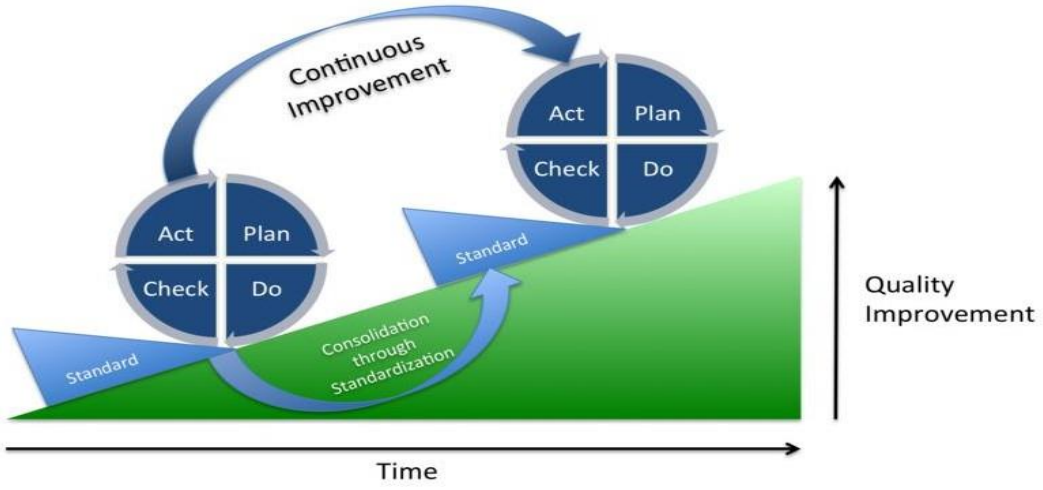


Fig No.18 Plan Execution

Selecting Sever virtualization Platform

Hardware Environment

These are the servers, storage and networking components of the data center being virtualized. It is possible to reuse the

hardware already in place. In fact, virtualization supports this by providing a consistent interface to the application to be deployed even if the hardware differs. The hardware environment chosen plays a crucial role in determining the software platform to be used.

Software Platform

The software layer abstracts the hardware environment to provide the hosted environments with an idealized environment. Distinct virtualization software has unique hardware requirements so when existing hardware is to be used, software choices are limited by compatibility with the hardware. Even when compatible hardware is used, the specifics influence performance. If exceptionally high performance is critical, hardware should be chosen very carefully.

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – II – VIRTUALIZATION TECHNIQUES SCSA7022

UNIT II BINARY TRANSLATION AND OPTIMIZATION

Virtual Machine basics – Interpretation – Interpreting Complex Instruction Set – Binary Translation – Dynamic Translation – Instruction Set issues – case Study Dynamic Binary Optimization: Program behaviour – profiling –optimizing translation blocks – framework – code reordering – optimization – ISA optimization system – VM Architecture: Object oriented high level language virtual machines – JVM architecture –Microsoft Common Language Infrastructure.

BINARY TRANSLATION AND OPTIMIZATION

Virtual Machines

Virtual machines (VMs) provide an intermediate stage for the compilation of programming languages

- VMs are machines because they permit a step-by-step execution of programs
- VMs are virtual (abstract) because typically they are not implemented in hardware omit many details of real (hardware) machines
- VMs are tailored to the particular operations required to implement a particular (class of) source language(s)

Pros

Bridge the gap between the high level of a programming language and the low level of a real machine.

- Require less implementation effort
- Easier to experiment and modify (crucial for new PLs)
- Portability is enhanced

– VM interpreters are typically implemented in C
– VM code can be transferred over the net and run in most machines
– VM code is (often significantly) smaller than object code

- Easier to be formally proven correct
- Various safety features of VM code can be verified
- Profiling and debugging are easier to implement

Cons

Inferior performance of VM interpreters compared with a native code compiler for the same language

- Overhead of interpretation
- Significantly more difficult to take advantage of modern hardware features (e.g. hardware-based branch prediction)

Interpretation

– simple and easy to implement, portable
– low performance
– threaded interpretation

- Binary translation

– complex implementation
– high initial translation cost, small execution cost

– selective compilation

ISOLATION AND App DOMAINS

- In Multiprogramming environment, programs must be isolated from each other for security reasons
- Conventional solution for this isolation is providing individual VM for each process
- JVM user loader namespaces to achieve isolation among programs
- CLI supports App Domains (Processes can share the same VM with isolation)

• An interpreter needs to maintain the complete architected state of the machine implementing the source ISA

Forms of Interpreters

Programming language implementations often use two distinct kinds of interpreters:

– Command-line interpreter
• Reads and parses language constructs in source form
• Used in interactive systems
– Virtual machine instruction interpreter
• Reads and executes instructions in some intermediate form such as VM bytecode Implementation of Interpreters

There are various ways to implement interpreters:

1.Direct string interpretation
2.Compilation into a (typically abstract syntax) tree and interpretation of that tree
3.Compilation into a virtual machine and interpretation of the VM code Basic Structure of a Bytecode Interpreter

```
byte *pc = &byte_program[0]; while(TRUE) {
opcode = pc[0]; switch (opcode) {
…
case GET_CONST2: source_reg_num = pc[1];
const_num_to_match = get_2_bytes(&pc[2]);
… // get_const2 code pc += 4;
break;
…
case JUMP:
jump_addr = get_4_bytes(&pc[1]); pc = &byte_program[jump_addr]; break;
}
```

Bytecode Interpreter with Aligned Instructions

```
byte *pc = &byte_program[0];
while(TRUE) { opcode = pc[0]; switch (opcode) {
…
case GET_CONST2: source_reg_num = pc[1];
const_num_to_match = get_2_bytes(&pc[2]);
… // get_const2 code pc += 4;
```

```
break;
…
case JUMP: // aligned version jump_addr = get_4_bytes(&pc[4]); pc = &byte_program[jump_addr]; break;
…
}
}
```

Indirectly Threaded Interpreters

In an indirectly threaded interpreter we do not switch on the opcode encoding. Instead we use the bytecodes as indices into a table containing the addresses of the VM instruction implementations

•The term threaded code refers to a code representation where every instruction is implicitly a function call to the next instruction

•A threaded interpreter can be very efficiently implemented in assembly

•In GNU CC, we can use the labels as values C language extension and take the address of a label with && label name

•We can actually write the interpreter in such a way that it uses indirectly threaded code if compiled with GNU CC and a switch for compatibility

Structure of Indirectly Threaded Interpreter

```
byte *pc = &byte_program[0];
while(TRUE) { next_instruction:
opcode = pc[0]; switch (opcode) {
…
case GET_CONST2:
get_const2_label:
source_reg_num = pc[GET_CONST2_ARG1];
 const_num_to_match = get_2_bytes(&pc[GET_CONST2_ARG2]);
… // get_const2 code
pc += GET_CONST2_SIZEOF;
NEXT_INSTRUCTION;
…
case JUMP: // aligned version jump_label:
jump_addr = get_4_bytes(&pc[JUMP_ARG1]);
 pc = &byte_program[jump_addr]; NEXT_INSTRUCTION;
…
}
}
```

Directly Threaded Interpreter

In a directly threaded interpreter, we do not use the bytecode instruction encoding at all during runtime

•Instead, the loader replaces each bytecode instruction encoding (opcode) with the address of the implementation of the instruction

•This means that we need one word for the opcode,

which slightly increases the VM code size Structure of Directly Threaded Interpreter
byte *pc = &byte_program[0]; while(TRUE) { next_instruction:
opcode = pc[0]; switch (opcode) {
…
case GET_CONST2:
get_const2_label:
source_reg_num = pc[GET_CONST2_ARG1];
const_num_to_match = get_2_bytes(&pc[GET_CONST2_ARG2]);
… // get_const2 code
pc += GET_CONST2_SIZEOF;
NEXT_INSTRUCTION;
…
case JUMP: // aligned version jump_label:
pc = get_4_bytes(&pc[JUMP_ARG1]); NEXT_INSTRUCTION;
…
}

}

Binary Translation

Translate source binary program to target binary before execution

- is the logical conclusion of predecoding
- get rid of parsing and jumps altogether
- allows optimizations on the native code
- achieves higher performance than interpretation
- needs mapping of source state onto the host state (state mapping)
- Each guest ISA instruction translates into some set of host (or *native*) ISA instructions
- Instead of dynamically fetching and decoding instructions at run-time, translate entire binary program and save result as new native ISA executable
- Removes interpretive fetch-decode overhead
- Can optimize translated code to improve performance
  - register allocation for values flowing between guest ISA instructions
  - native instruction scheduling to improve performance
  - remove unreachable code
  - inline assembly procedures
  - remove dead code e.g., unneeded ISA side effects

<u>x86 Source Binary</u>

addl %edx,4(%eax) movl 4(%eax),%edx add %eax,4
<u>Translate to PowerPC Target</u>
r1 points to x86 register context block r2 points to x86 memory image r3 contains x86 ISA PC value

```
lwz     r4,0(r1)        ;load %eax from register block
addi    r5,r4,4         ;add 4 to %eax
lwzx    r5,r2,r5        ;load operand from memory
lwz     r4,12(r1)       ;load %edx from register block
add     r5,r4,r5        ;perform add
stw     r5,12(r1)       ;put result into %edx
addi    r3,r3,3         ;update PC (3 bytes)
lwz     r4,0(r1)        ;load %eax from register block
addi    r5,r4,4         ;add 4 to %eax
lwz     r4,12(r1)       ;load %edx from register block
stwx    r4,r2,r5        ;store %edx value into memory
addi    r3,r3,3         ;update PC (3 bytes)
lwz     r4,0(r1)        ;load %eax from register block
addi    r4,r4,4         ;add immediate
stw     r4,0(r1)        ;place result back into %eax
addi    r3,r3,3         ;update PC (3 bytes)
```

Code Discovery Problem

- May be difficult to *statically* translate or predecode the entire source program
- Consider x86 code

```
mov   %ch,0 ??
31 c0 8b b5 00 00 03 08 8b bd 00 00 03 00
movl  %esi, 0x08030000(%ebp)     ??
```

- Contributors to code discovery problem
  - variable-length (*CISC)* instructions
  - indirect jumps
  - data interspersed with code
  - padding instructions to align branch targets

Dynamic Translation

- Translate code sequences as needed at run time, but cache results
- Can optimize code sequences based on dynamic information (e.g., branch targets encountered)

- Tradeoff between optimizer run-time and time saved by optimizations in translated code
- Technique used in Java JIT (Just-In-Time) compilers

Instruction Set Issues
• Register architectures
– register mappings, reservation of special registers
• Condition codes
– lazy evaluation as needed
• Data formats and arithmetic
– floating point
– decimal
– MMX
• Address resolution
– byte vs word addressing
• Data Alignment
– natural vs arbitrary
• Byte order
– big/little endian

Dynamic Optimization
–binary is improved for higher performance may be done as part of emulation
–may optimize same ISA (no emulation needed)

Why do Dynamic Binary Translation and Optimization? Improve Performance:
–Optimization and Profile-Directed Feedback hidden from user. Make Architecture a Layer of

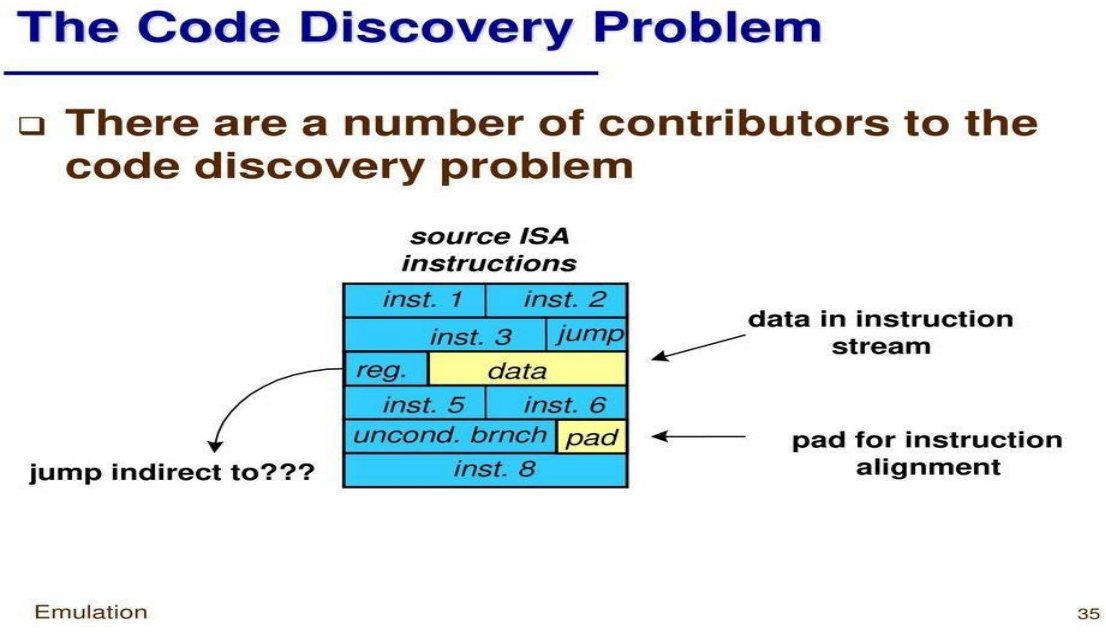


Fig.No.19 Code Discovery Problem

Software:
–Compatibility: with Legacy Architectures.

–Novelty: use great new architecture ideas.
–Multiplicity: of legacy architectures on one machine. Reduce Hardware Complexity.
–Save Power: Memory less power than logic) Software translator translates once and saves in memory. – No need to optimize with logic transistors each time code is executed.

Benefits of Dynamic Optimization
Legacy code where source is unavailable can be optimized.
Many commercial programs are never compiled with -O due to time-constraints on shipping product. – Development done with -g, not -O.

– No time for additional testing with -O compilation. Dynamic Optimization still allows high code quality.

Translated basic blocks can be layed out contiguously in order they are naturally visited. This helps ICache Performance.
Not limited in optimization scope. Can cross boundaries:
– Indirect Calls
– Function Returns
– Shared Libraries
– System Calls

Benefits of Dynamic Translation
Compatible: No changes are needed in existing code. Compatible: between VLIW's of different sizes, generations.
– Transmeta already uses this advantage.
Upgradable: If better compiler algorithms are found, only a software patch is needed to install them. Upgradable: Future architectural improvements are transparent to the user.
Reliable: If a bug is found in dynamic translator a software patch is sufficient to fix it. Reliable: Some hardware bugs can be worked around by translator.
High Chip Yield: Smarts in software not hardware) Smaller chip with higher yield.
Simple Fast Hardware: Intelligence is in software, allowing simple in-order implementations. Wide scope for ILP: Can look at arbitrarily long fragments of code. Problems with Dynamic Translation Takes away cycles from program.
Takes memory and resources from emulated machine. Especially slow at start.
Potential realtime difficulties.
Debugging can be difficult: – Target machine code several times removed from source code.
– Behavior can be non-deterministic in real system.

ISA Implementations Partly in Software
Often good idea to implement part of ISA in software:
- Expensive but rarely used instructions can cause trap to OS emulation routine:
  - e.g., decimal arithmetic in µVax implementation of VAX ISA
  - Infrequent but difficult operand values can cause trap
  - e.g., IEEE floating-point denormals cause traps in almost all floating-point unit implementations
  - Old machine can trap unused opcodes, allows binaries for *new*

ISA to run on *old* hardware
  - e.g., Sun SPARC v8 added integer multiply instructions, older v7 CPUs trap and emulate

Supporting Non-Native ISAs
Run programs for one ISA on hardware with different ISA

- Emulation *(OS software interprets instructions at run-time)*
  - E.g., OS for PowerPC Macs had emulator for 68000 code
  - Binary Translation *(convert at install and/or load time)*
  - IBM AS/400 to modified PowerPC cores
  - DEC tools for VAX->Alpha and MIPS->Alpha
  - Dynamic Translation *(non-native ISA to native ISA at run time)*
  - Sun's HotSpot Java JIT (just-in-time) compiler
  - Transmeta Crusoe, x86->VLIW code morphing
  - Run-time Hardware Emulation
  - IBM 360 had IBM 1401 emulator in microcode
  - Intel Itanium converts x86 to native VLIW (two software-visible ISAs)
  - ARM cores support 32-bit ARM, 16-bit Thumb, and JVM (three software-visible ISAs!)

VM Architecture

Architecture of Virtual Machines

- VM can support individual processes or a complete system
- Virtualization can be from OS to programming languages to processor architecture.

- VMs enhance

  - Software interoperability (to work together)
  - System impregnability (having strength)
  - Platform versatility

Abstraction and Virtualization

- Computer system is complex, and yet it continue to evolve.
- Computer is designed as hierarchies of well- defined interfaces that separate level of abstraction
- Simplifying abstractions hide lower-level implementation details

Abstraction

- Ex. Disk storage
- Hides hard-disk addressing details
  (sectors and tracks)

It appears to application software as a variable sized files.

- User can create,write and read files without knowing the underneath details.

- Virtualization of system or components like – processor,memory or an at a given abstraction level.

- It transforms a entire system or components of the system

- Ex. disk storage

Virtual Machine

- Virtualization can be applied to entire machine.

- VM can be implemented by adding a software layer to a real machine to support desired architecture.

- VM implementation lie at architected interfaces

Architected Interfaces

\Architecture, as applied to computer systems, refer to a formal specification to an interface in the system, including the logical behavior of the resources managed via the interface.

- Implementation describes the actual embodiment of an architecture.
- Abstraction levels correspond to implementation layers, having its own interface or architecture.
- Computer System Architecture
- Interfaces at or near the H/w S/w boundary :-
- ISA – Instruction Set Architecture.
- API – Application Program Interface
- ABI – Application Binary Interface

Advantages of Virtualization

Managed Execution types :-

- Sharing
  - Creating separate computing environment within the same host.
  - Underline host is fully utilized.
- Aggregation
  - A group of separate hosts can be tied togetherand represented as single virtual host.

- Emulation
  - Controlling & Tuning the environment exposed to guest.
- Isolation
  - Complete separate environment for guests.
- HLL PVM similar to a *conventional* PVM
- V-ISA not designed for a real hardware processor

Virtualizing Conventional ISA Vs. High-Level-Language VM ISA Drawbacks of virtualizing a conventional ISA

- not developed for being virtualized!
- operating system dependencies
- issues with fixed-size address space, page-size
- memory address formation
- maintaining precise exceptions
- instruction set features
- instruction discovery during indirect jumps
- self-modifying and self-referencing code

C-ISA Not for Being Virtualized Conventional ISA

- after the fact solution for portability
- no built-in ISA support for virtualization High-level language V-ISA

- VM based portability is a primary design goal
- generous use of metadata
- metadata allows better type-safe code verification, interoperability, and performance

Operating System Dependencies Conventional ISA

- most difficult to emulate
- exact emulation may be impossible (different OS) High-level language V-ISA
- find a least common denominator set of functions
- programs interact with the library API
- library interface is higher level than conventional OS interface

Memory Architecture Conventional ISA

- fixed-size address spaces
- specific addresses visible to user programs High-level language V-ISA
- abstract memory model of indefinite size
- memory regions allocated based on need
- actual memory addresses are never visible
- out-of-memory error reported if process requests more that is available of platform

Memory Address Formation Conventional ISA

- unrestricted address computation
- difficult to protect runtime from un- authorized guest program accesses High-level-language V-ISA
- pointer arithmetic not permitted
- memory access only through explicit memory pointers
- static/dynamic type checking employed

Precise Exceptions Conventional ISA

- many instructions trap, precise state needed
- global flags enable/disable exceptions High-level language V-ISA
- few instructions trap
- test for exception encoded in the program

- requirements for precise exceptions are relaxed

Instruction Set Features Conventional ISA

- guest ISA registers > host registers is a problem
- ISAs with condition codes are difficult to emulate High-level language V-ISA
- stack-oriented
- condition codes are avoided

Instruction Discovery Conventional ISA

- indirect jumps to potentially arbitrary locations
- variable-length instruction, embedded data, padding High-level-language V-ISA
- restricted indirect jumps
- no mixing of code and data
- variable-length instructions permitted

- Used in a networked computing environment
- Important features of HLL VMs
  - security and protection
    - protect remote resources, local files, VM runtime
  - robustness
    - OOP model provides component-based programming, strong type- checking, and garbage collection
  - networking
    - incremental loading, and small code-size
  - performance
    - easy code discovery allows entire method compilation
- Java Virtual Machine Architecture ó CLI
  - analogous to an ISA
- Java Virtual Machine Implementation óCLR
  - analogous to a computer implementation
- Java bytecodes ó Microsoft Intermediate - Language (MSIL), CIL, IL
  - the instruction part of the ISA
- Java Platform - .NET framework
  - ISA + Libraries; a higher level ABI

Modern HLL VM

- Compiler frontend produces binary files

- standard format common to all architectures

- Binary files contain both code and metadata

JVM Architecture - Java Virtual Machine

- Abstract entity that gives meaning to class files
- Has many concrete implementations
  - hardware
  - interpreter
  - JIT compiler
- Persistence
  - an instance is created when an application starts
  - terminates when the application finishes JVM Implementation

- A typical JVM implementation consists of
  - class loader subsystem , memory subsystem, emulation/execution engine, garbage collector

Class Loader

- Functions
  - find the binary class
  - convert class data into implementation- dependent memory image
  - verify correctness and consistency of the loaded classes
- Security checks
  - checks class magic number
  - component sizes are as indicated in class file
  - checks number/types of arguments verify integrity of the bytecode program

Java Native Interface(JNI)

- Allows java code and native code to interoperate
  - access legacy code, system calls from Java
  - access Java API from native functions
  - each side compiles to its own binary format
  - different java and native stacks maintained
  - arguments can be passed; values/exceptions returned Microsoft Common Language Infrastructure

- The Microsoft CLI

- Attributes
- Microsoft Intermediate Language
- Isolation and App Domains

COMMON LANGUAGE INTERFACE

- CLI is designed to support programs multiple, interoperating High level languages
- CLI supports programs that do not have to be verified by a class loader
- CLI provides HLL independence as well as platform independence
- CLI is implemented as CLR, a part of the overall .NET framework
- Code and Metadata of various languages are represented as modules in Microsoft Intermediate Language(MSIL)

There are three categories of programs executed in CLI namely,

- Verifiable and valid
- Unverifiable and valid
- Invalid

JVM:

- JNI is needed
- Interoperability is at method level

CLI:

- No such interfaces are needed

Interoperability is at method and data level Q

ATTRIBUTES

- In CLI environment it is possible to assign values for attributes during runtime
- GetCustomAttribute method serves this purpose Example: conversion inches to centimetres in the metadata

MICROSOFT INTERMEDIATE LANGUAGE

- The MSIL is Stack oriented
- Memory architecture contains,
  - Local data area

- Argument area
- Metadata Streams (as Constant Pool in JVM)

➢ As JVM Stack Tracking facility is available in CLI

ISOLATION AND App DOMAINS

- ➢ In Multiprogramming environment, programs must be isolated from each other for security reasons
- ➢ Conventional solution for this isolation is providing individual VM for each process
- ➢ JVM user loader namespaces to achieve isolation among programs
- ➢ CLI supports App Domains (Processes can share the same VM with isolation)

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – III – VIRTUALIZATION TECHNIQUES SCSA7022

UNIT III NETWORK VIRTUALIZATION

Design of Scalable Enterprise Networks - Virtualizing the Campus WAN Design - WAN Architecture - WAN Virtualization - Virtual Enterprise Transport Virtualization - VLANs and Scalability - Theory Network Device Virtualization Layer 2 - VLANs Layer 3 VRFInstances Layer 2 - VFIs Virtual Firewall Contexts Network Device Virtualization - Data - Path Virtualization Layer 2: 802.1q - Trunking Generic Routing Encapsulation - IPsec L2TPv3 Label Switched Paths - Control - Plane Virtualization - Routing Protocols - VRF - Aware Routing Multi-Topology Routing.

Design of Scalable Enterprise Networks

Introduction

As a business grows, so does its networking requirements. Businesses rely on the network infrastructure to provide mission-critical services. Network outages can result in lost revenue and lost customers. Network designers must design and build an enterprise network that is scalable and highly available.

This chapter introduces strategies that can be used to systematically design a highly functional network, such as the hierarchical network design model, the Cisco Enterprise Architecture, and appropriate device selections. The goals of network design are to limit the number of devices impacted by the failure of a single network device, provide a plan and path for growth, and create a reliable network. Implementing a Network Design Hierarchical Network Design The Need to Scale the Network

Businesses increasingly rely on their network infrastructure to provide mission-critical services. As businesses grow and evolve, they hire more employees, open branch offices, and expand into global markets. These changes directly affect the requirements of a network. A large business environment with many users, locations, and systems is referred to as an enterprise. The network that is used to support the business enterprise is called an enterprise network.

An enterprise network must support the exchange of various types of network traffic, including data files, email, IP telephony, and video applications for multiple business units. All enterprise networks must:

- Support critical applications
- Support converged network traffic
- Support diverse business needs
    - Provide centralized administrative control Expanding the Network

Design for Scalability

To support an enterprise network, the network designer must develop a strategy to enable the network to be available and to scale effectively and easily. Included in a basic network design

strategy are the following recommendations:

- Use expandable, modular equipment or clustered devices that can be easily upgraded to increase capabilities. Device modules can be added to the existing equipment to support new features and devices without requiring major equipment upgrades. Some devices can be integrated in a cluster to act as one device to simplify management and configuration.
- Design a hierarchical network to include modules that can be added, upgraded, and modified, as necessary, without affecting the design of the other functional areas of the network. For example, creating a separate access layer that can be expanded without affecting the distribution and core layers of the campus network.
- Create an IPv4 or IPv6 address strategy that is hierarchical. Careful IPv4 address planning eliminates the need to re-address the network to support additional users and services.
- Choose routers or multilayer switches to limit broadcasts and filter other undesirable traffic from the network. Use Layer 3 devices to filter and reduce traffic to the network core.

More advanced network design requirements include:

- Implementing redundant links in the network between critical devices and between access layer and core layer devices.
- Implementing multiple links between equipment, with either link aggregation (Ether Channel) or equal cost load balancing, to increase bandwidth. Combining multiple Ethernet links into a single, load-balanced Ether Channel configuration increases available bandwidth. Ether Channel implementations can be used when budget restrictions prohibit purchasing high-speed interfaces and fiber runs.
- Implementing wireless connectivity to allow for mobility and expansion.
- Using a scalable routing protocol and implementing features within that routing protocol to isolate routing updates and minimize the size of the routing table.

Virtualizing the Campus

To preserve the hierarchy and path symmetry provided by a multilayer routed core, this routed core must be kept in place. It is tempting for the network architect to associate the concept of virtual networks (VNs) with that of VLANs. Although a VLAN is a form of a VN, it is not by itself a hierarchical or modular VN and therefore does not preserve all the characteristics of resiliency and scalability desired in the enterprise. VLANs are an important component of a VN because they allow the virtualization of the Layer 2 portion of the network. However, to preserve the desired scalability and resiliency, a routed core is necessary. Therefore, the VLANs in the Layer 2 access must be combined with Layer 3 VPNs in the routed core. By combining VLANs with Layer 3 VPNs, you can create an end-to-end hierarchical VN in the campus. Therefore, an overlay of VPNs must be added to the routed core to virtualize it. The choice of VPN technology to use depends significantly on how well the VPN technology accommodates a hierarchical routing structure, the efficient use of redundant paths for load balancing, and failover and support for the different types of traffic present in the enterprise.

WAN Design

As the network is spread over long distances, reliable and fast transmission media with high bandwidth is required, thus fiber optic cable is mostly used for WAN connectivity. The switching technology used in WAN includes both circuit and packet switching depending upon the network architecture.

The WAN networks are designed in such a way in which the enterprise's head office will be connected with the branch offices and centralized data center with internet connectivity to all the end users if they have relevance.

Design Concerns

- The network should be designed in such a way in which the overall architecture designed should be cost-effective and within the budget.
- The links used for connectivity should be reliable and in protection. By provisioning protection, if one link fails the network will still be alive by using the protection link.
- The overall network throughput should come out best and packet delay should be as minimal as possible.
- The network should be designed in such a way in which there should be minimal interference, jitter, and packet loss.
- The basic goal of a well-designed network is to deliver data to the destination host from the source host by using the shortest path.
- The components equipped in the network should be well utilized and managed properly.
- A strong firewall system should be used to provide reliable and secure transmission.
- The network topology, transmission modes, routing policy and the other network parameters should be chosen depending upon the type and need of the system to be implemented.

WAN Networking Technologies

There are two technologies used in the WAN network designing. Below are the classifications:

Circuit Switching: The example of circuit switching includes DWDM, SDH, or TDM.

Packet switching: The type of switching includes ATM, frame relay, multi-protocol label switching (MPLS) and IPV4 or IPV6.

Circuit Switching

It is the method of employing a communication networking system in which a dedicated communication channel is established between the two communicating nodes throughout the communication process. The channel or circuit has been provided with a dedicated bandwidth throughout the communication process.

SDH and DWDM technologies use circuit switching for communication.

Packet Switching

Packet switching is a kind of switching process in which data is sent in a network in the form of packets. The big chunk of data is firstly broken into small variable length data called the packets. Then these are sent over the transmission media. At the destination end, these are reassembled and delivered to the destined host. No pre-setup of the link is required in this method. The data transmission is fast and transmission latency is minimal. Packet switching deploys the store and forwards the procedure for routing the packets. Each of the packets has both a source and destination address through which it can reach the destination by following various paths.

If there is congestion at any hop level, then the packet will follow a different path to reach the destination. If the receiver discards the data packets, then it can be re-transmitted again.

Packet switching is of two types i.e. Connection-oriented and Connectionless switching.

(i) **Connectionless Switching:** In video streaming, online gaming, online TV, Internet etc., the connectionless packet switching is used as if some of the packets are lost during transmission, it doesn't impact the overall data much.

(ii) **Connection-oriented Switching:** In Invoice and data transmission, connection-oriented packet switching is used.

IPV4 and IPV6 are few common types of packet switching methods.

Software-Defined Wide Area Network (SD-WAN)

A software-defined wide-area network (SD-WAN), is a network that is abstracted from its hardware, creating a virtualized network overlay. Operators can remotely manage and quickly scale this overlay, which can span over large geographical distances. It is an application of software-defined networking (SDN).

An SD-WAN can connect several branch locations to a central hub office or cover multiple locations in a large campus such as a university campus. Because it is abstracted from hardware, it is more flexible and available than a standard WAN. It relies on four central components:

- Edge connectivity abstraction
- WAN virtualization
- Centralized management
- Elastic traffic management
- SD-WAN Architecture

SD-WAN use an abstracted architecture for its network. In an abstracted architecture, the network is divided into two parts: the control plane and the forwarding plane. This architecture moves the control plane to a centralized location like an organization's headquarters. That way, the network can be managed remotely without the need for an on-premises IT crew.

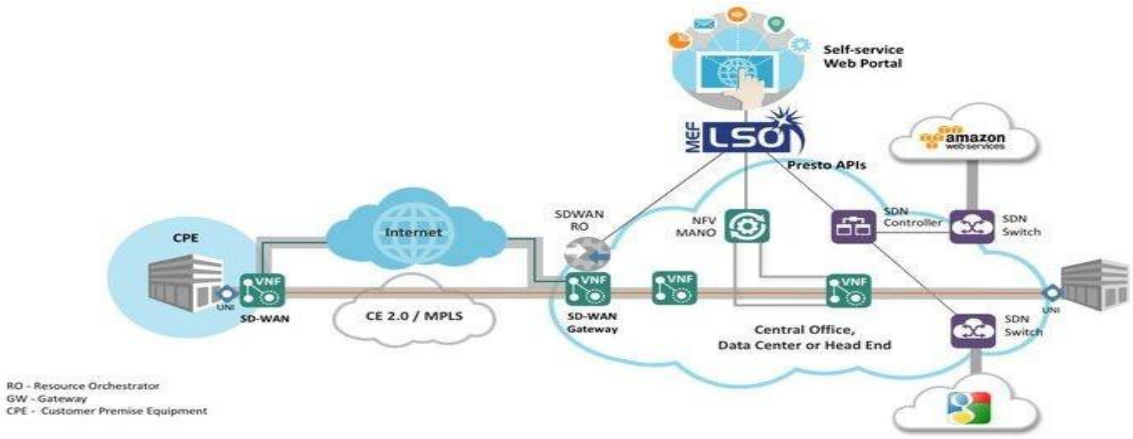


Fig No.20: SD-WAN architecture with MPLS integration. Source: MEF

There are three main components to an SD-WAN: The SD-WAN edge, the controller, and the orchestrator.

- The SD-WAN edge is where the network endpoints reside. This can be a branch office, a remote data center, or cloud platform.
- An SD-WAN Orchestrator is the virtualized manager for network, overseeing traffic and applying policy and protocol set by operators.
- The SD-WAN Controller centralizes management, and enables operators to see the network through a single pane of glass, and set policy for the orchestrator to execute.

These components make up the basic structure of an SD-WAN. In addition, there are three main types of SD-WAN architecture: on-premises, cloud-enabled, and cloud-enabled with a backbone.An On-premises SD-WAN is where the SD-WAN hardware resides on-site. Network operators can directly access and manage the network and the hardware it resides on, and it does not use the cloud for its connections. This makes making it ideal for sensitive information that cannot be sent over the internet. Cloud-enabled SD-WANs connect to a virtual cloud gateway over the internet, which makes the network more accessible, and provides better integration and performance with cloud-native applications. Cloud-Enabled with Backbone SD- WANs give organizations an extra backup by connecting the network with a nearby point of presence (PoP), such as within a data center. It allows SD-WAN to switch from public internet to a private connection, which adds an extra layer of security and consistency in case the connection is overwhelmed or fails.

SD-WAN Infrastructure

Because of its virtualized architecture, SD-WAN doesn't require specific hardware for specialized network functions. Instead, the infrastructure is made of commercial off-the-shelf (COTS)

equipment, also known as white boxes. Certain types of COTS hardware, such as universal customer premises equipment (uCPE) can host a variety of network functions. This simplifies network management at a network edge or organization's headquarters. Enterprises can deploy SD-WAN in a DIY manner, where the business owns the network and equipment and takes full responsibility for the network operation and upkeep. In turn, enterprises can use a managed service provider, who owns all network equipment and maintains some control over the network, and takes the brunt of the network management responsibility.

The Benefits of SD-WAN

The global SD-WAN market is predicted to swell to $5.25 Billion in 2023, according to an IDC forecast in July 2019, as more businesses embrace the benefits of a virtualized network.

The key benefits include:

- Increased bandwidth at a lower cost since the network traffic can be provisioned for optimal speeds and throttle low-priority applications.
- Centralized management across branch networks through a simple management console, which reduces the need for manual configuration and on-site IT staff
- Full visibility into the network, as the controller gives operators a holistic view of the network.
- More options for connection type and vendor selection, since the network can reside on COTS hardware and use both private and public connections to route its traffic.

WAN Virtualization

The term "virtualization" has been historically used for describing technologies where a group of physical assets are intelligently managed by a software layer for a simpler, more efficient and better performing presentation of the resources to the higher layers. As an example server virtualization is the technology that describes the overlay software layer on top of many servers (or alternatively creating multiple virtual instances from a single server) that presents a simple interface to the applications that utilize it.

A similar concept has emerged in the WAN (Wide Area Networks) space, where an intelligence/management WAN virtualization layer (Broadband Bonding appliance) on top of the real WAN resources (DSL, Cable, T1, MPLS etc.) will provide a simple, higher performance IP pipe to the applications that are using the Internet connectivity. With Wan virtualization (Broadband Bonding) various number of Internet lines can be bonded into a single connection. This provides faster connectivity (the sum of all the line speeds) as well as an intelligent management of the latency within the tunnel. WAN Virtualization / Broadband Bonding is quickly becoming a must have technology tool for any IT manager or business owner in today's cloud based (public and private) information technology business world we live in.

Transport Virtualization

CE devices connect to customer Ethernet clouds and face a provider packet-switching network (or enterprise network core) using a Layer 3 interface. Every CE device creates an overlay tunnel interface that is very much similar to MDT Tunnel interface specified in Draft Rosen. Only IP transport is required in the core and thus the technology does not depend on MPLS transport.

Just like with Rosen's mVPNs all CE's need to join a P (provider) multicast group to discover other CE's over the overlay tunnel. The CE's then establish a full mesh of IS-IS adjacencies with other CE's on the overlay tunnel. This could be thought as an equivalent for full-mesh of control-plane link in VPLS. The provider multicast group is normally an ASM or Bidir group.

IS-IS nodes (CEs) flood LSP information including the MAC addresses known via attached physical Ethernet ports to all other CE's. This is possible due to flexible TLV structure found in ISIS LSPs. The same LSP flooding could be used to remove or unlearn a MAC address if the local CE finds it unavailable.

Ethernet forwarding follows the normal rules, but GRE encapsulation (or any other IP tunneling) is used when sending Ethernet frames over the provided IP cloud to a remote CE. Notice that GRE packets received on overlay tunnel interfaces do NOT result in MAC-address learning for encapsulated frames. Furthermore, unknown unicast frames are NOT flooded out of overlay tunnel interface – it is assumed that all remote MAC addresses are signaled via control plane and should be known.

Multicast Ethernet frames are encapsulated using multicast IP packets and forwarded using provider multicast services. Furthermore, it is possible to specify a set of "selective" or "data" multicast groups that are used for flooding specific multicast flows, just like in mVPNs. Upon a reception of an IGMP join, a CE will snoop on it (similar to the classic IGMP snooping) and translate into core multicast group PIM join. All CE's receiving IGMP reports for the same group will join the same core multicast tree and form an optimal multicast distribution structure in the core. The actual multicast flow frames will them get flooded down the selective multicast tree to the participating nodes only. Notice one important difference from mVPNs – the CE devices are not multicast routers, they are effectively virtual switches performing IGMP snooping and extended signaling in provider core.

OTV handles multi-homed scenarios properly, without running STP on top the overlay tunnel. If two CE's share the same logical backdoor link (i.e. they hear each other ISIS hello packets over the link) one of the devices is elected as appointed (aka authoritative) forwarder for the given link (e.g VLAN). Only this device actually floods and forwards the frames on the given segment, thus eliminating Layer 2 forwarding loop. This concept is very similar o electing a PIM Assert winner on a shared link. Notice that this approach is simialar to VPLS draft proposal for multihoming, but uses IGP signaling instead of BGP.

OTV supports ARP optimization in order to reduce the amount of broadcast traffic flooded across the overlay tunnel. Every CE may snoop on local ARP replies and use the ISIS extensions to signal IP to MAC bindings to remote nodes. Every CE will then attempt to respond to an ARP request

using its local cache, instead of forwarding the ARP packet over the core network. This does not eliminate ARP, just reduces the amount of broadcast flooding

Now for some conclusions. OTV claims to be better than VPLS, but this could be argued. To begin with, VPLS is positioned as provider edge technology and OTV is customer-edge technology. Next, the following list captures similarities and differences between the two technologies:

The same logical full-mesh of signaling is used in the core. IS-IS it outlined in the patent document, but any other protocol could be obviously used here, e.g. LDP or BGP. Even the patent document mentions that. What was the reason to re-inventing the wheel? The answer could be "TRILL" as we see in the following section. But so far switching to new signaling makes little sense in terms of benefits.

OTV runs over native IP, and does not require underlying MPLS. Like we said before, it was possible to simple change VPLS transport to any IP tunneling technique instead of coming with a new technology. By missing MPLS, OTV loses the important ability to signal optimal path selection in provider networks at the PE edge.

Control Plane MAC-address learning is said to reduce broadcast in the core network. This is indeed accomplished but at a significant price. Here is the problem: If a topology change in one site is to be propagated to other sites, control plane must signal the removal of locally learned MAC addresses to the remote sites. Effectively, this will translate data-plane "black-holing" until the MAC addresses are not re-learned and signaled again, as OTV does not flood over the IP core.The things are even worse in control plane. A topology change will flush all MAC addresses known to a CE an result in LSP flooding to all ajdacent nodes. The amount of LSP
replication could be optimized using IS-IS mesh-groups, but at least N copies of LSP should be sent, where N is the number of adjacencies. As soon as new MACs are learned, additional LSP will be flooded out to all neighbors! Properly controlling LSP generation, i.e. delaying LSP sending may help reduce flooding but again will result in convergence issues in data plane.

To summarize, the price paid for flooding reduction is slower convergence in presence of topology changes and control-plane scalability challenges. The main problem – topology unawareness that leads to the need of re-learning MAC address is not addressed in OVT (yet). However, if you think that data-plane flooding in data-centers could be very intensive, the amount of control plane flooding introduced could become acceptable.

Optimized multicast support seems to be the only big benefit of OTV that does not result in significant trade-offs. Itroducing native multicast it's probably due to the fact that VPLS multicasting is still not standardized, while datacenters need it now. The multicast solution is a copy of mVPNs model and not something new and exciting, like M-LSPs are. Like we said before, the same idea could be deployed in VPLS scenarios by means of co-located mVPN. Also, when deployed over SP networks, this feature requires SP multicast support for auto-discovery and optimized forwarding. This is not a major problem though, and OTV has support for unicast static discovery.

To summarize, it looks like OTV is an attempt to fast-track a slightly better "customer" VPLS in datacenters, while IETF folks struggle for actual VPLS standardization. The technology is "CE-centric", in essence that it does not require any provider intervention with exception to providing multicast and L3 services. It is most likely that OTV and VPLS projects are being carried by different teams that are being time-pressed and thus don't have resources to coordinate their efforts and come with a unified solution. There are no huge improvements so far in terms of Ethernet optimization, with except to reduced flooding in network core, traded for control-plane complexity. At its current form, OTV might look a bit disappointing, unless is a first step to implementing TRILL (Transport Interconnection for Lots of Links) – new IETF standard for routing bridges.

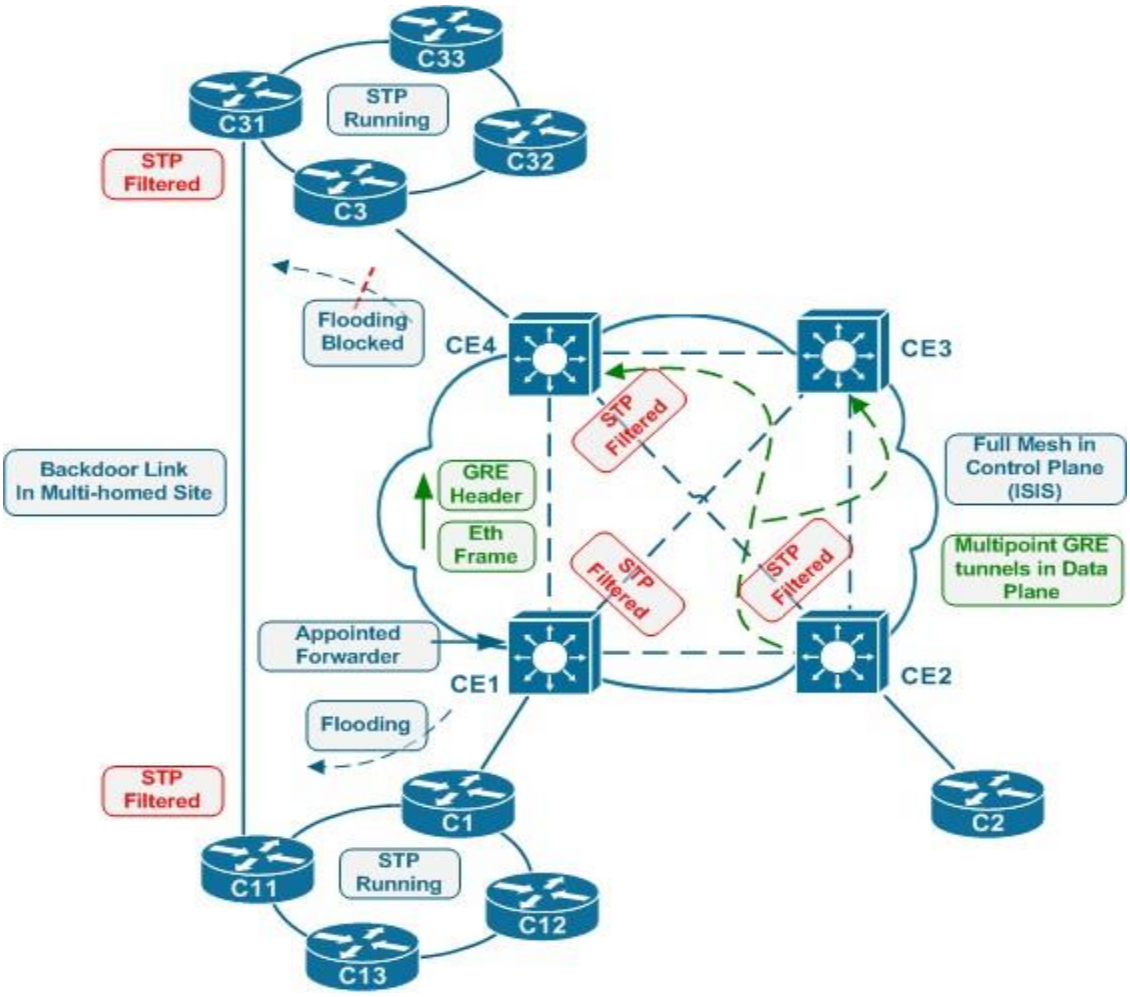


Fig No.21 Transport Virtualization Architecture

Network Virtualization advantages

- Infrastructure utilization
- Infrastructure is shared between many different users or purposes

• Reduces infrastructure & energy cost

• Scalability

• Easy to extend resources in need

• Administrator can dynamically create or delete virtual network resources

• Agility

• Enables automation of network services establishment

• Network services can be orchestrated together with other IT infrastructure

• Resilience

• Virtual network will automatically redirect packets by redundant links

• In case of disaster, the virtual network can be easily recreated on new physical infrastructure

• Security

• Increased data traffic isolation and user segmentation

• Virtual network should work with firewall software

Objects of network virtualization

• Device virtualization

Virtualize physical devices (nodes) in the network

- Data Plane virtualization
- Control Plane virtualization
- Management Plane Virtualization

• Data path virtualization

• Virtualize communication path between network access points

• Links virtualization

Device Virtualization:

Virtualization is the act of creating a virtual (rather than physical) version of a computer application, storage device, or a computer network resource. Virtualization makes the logical server, storage, and network independent of the deployed physical infrastructure resources.

Oracle Internet of Things Cloud Service uses virtualization to make it easier to integrate external

devices and data with Oracle Internet of Things Cloud Service. Oracle Internet of Things Cloud Service exposes every connected device as a set of resources called a device model. The use of device virtualization abstracts any complexity associated with device connectivity and standardizes device integration with the enterprise. With it, enterprise applications can directly address any device from Oracle Internet of Things Cloud Service regardless of network protocol and firewall restrictions.

A device model is a single entity that represents the interface through which Oracle Internet of Things Cloud Service can interact with a specific device type, regardless of the vendor, underlying standard, or specification that defined that device model. It can represent any object on the device side that can send messages or respond to REST requests. This object type includes devices, gateways, device adapters, and device applications.

VLAN

In the past, the most common approach to VLAN assignment would be to manually configure a port to be a member of a specific VLAN and potentially define a voice VLAN for that port as well. Another method which is becoming much more common today is through the enhanced security capabilities of Flexible Authentication Sequencing using 802.1X, MAC Authentication and Bypass (MAB), or Webauth as alternate means to first authenticate a user against a Radius Server or a Policy Enforcement Server, such as the Cisco Identity Services Engine (ISE), for network access. Once authenticated, by using Radius attributes communicated between the Radius Server and access switch the switchport is dynamically changed to the appropriate VLAN and, optionally, an ACL can be pushed down to the switch enforcing specific access to the network. It is beyond the scope of this document to detail these technologies; refer to the TrustSec Phased Deployment Configuration.

Virtual Routing and Forwarding

We discussed how VLANs are the most basic path isolation technique for Layer 2. However as the goal of every solid network design is to minimize the extent of the broadcast domain and exposure to Spanning Tree loops, a method to translate the Layer 2 VLAN to a Layer 3 virtual network or VPN is required. This Layer 3 VN must be capable of supporting its own unique control plane complete with its own addressing structure and routing tables for data forwarding completely isolated from any other Layer 3 VPN on that device and in the network. The technology enabling this type of functionality is known as Virtual Routing and Forwarding (VRF) instance. Figure 3 draws the comparison between Layer 2 VLANs and Layer 3 VRFs.

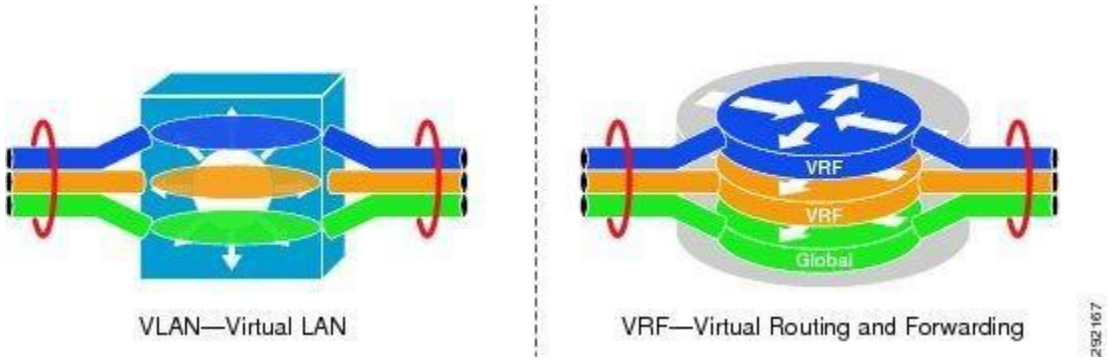


Fig No.22 Virtual Routing and Forwarding

Comparison of Layer 2 VLAN and Layer 3 VRF

The VRF is defined on a networking device that serves as the boundary between the Layer 2, client-side VLANs, and the Layer 3 network. Each VRF instance consists of an IP routing table, a forwarding table, and interface(s) assigned to it. Common routing protocols such as OSPF, EIGRP, BGP, and RIPv2 can be used to advertise and learn routes to populate the routing table unique to each virtual network. This routing information is then used to populate the CEF table using those interfaces, either logical (SVI) or interfaces and sub-interfaces that have been specifically allocated to that VRF through device configuration. VRFs exist on top of a global routing table consisting of IPv4 prefixes and interfaces that have not been assigned to a VRF.

VRF instances can be compared to virtual routers co-resident on a single Layer 3 switch or router. However, the comparison stops inasmuch as the VRF does not carve out any dedicated

compute or memory from the physical device. Figure 4 depicts three VRFs residing on a single physical device.

As we discuss characteristics of routing in a virtualized environment, it is important to understand that Cisco routers support a number of routing protocols and individual processes per router. Some routing protocols such as BGP only support a single instance of that process and hence the concept of routing contexts was developed. These contexts were designed to support isolated copies of the routing protocol running per VRF.

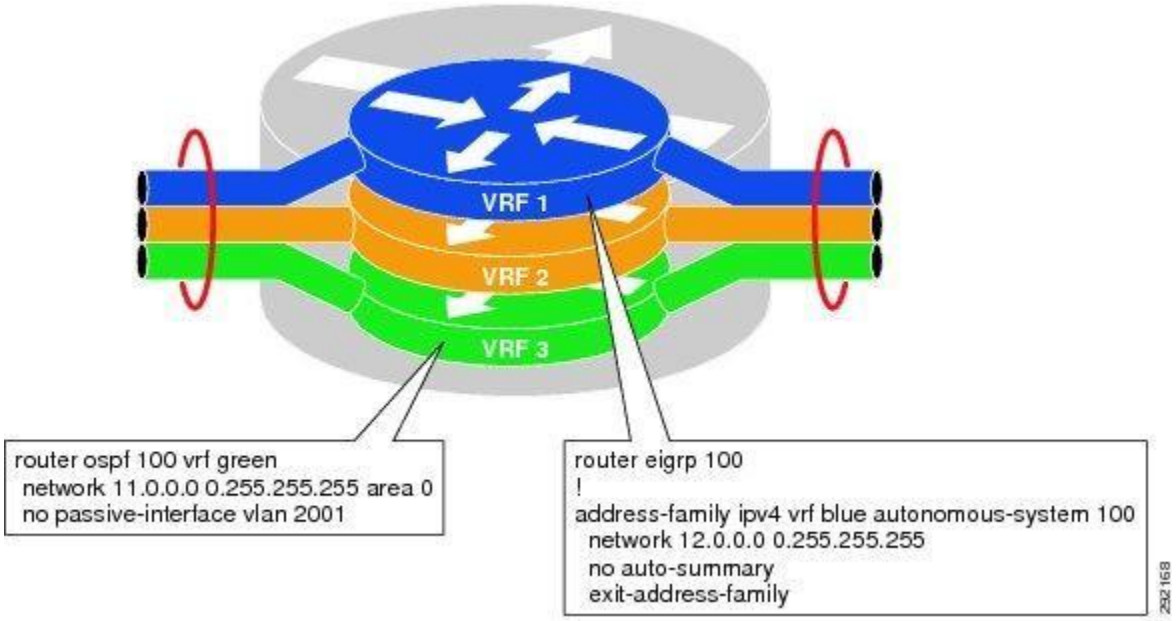


Fig No.23 . Routing Contexts

Routing Contexts through Processes and Address Families

An important concept to understand is that VRFs and the virtual networks they represent, complete with their own routing tables and forwarding paths, are an overlay on top of a base or non-virtualized IP infrastructure. This non-virtualized infrastructure has its own "Global" routing table and data forwarding plane. Typically, as an organization approaches network virtualization, all user/device traffic flows across the non-virtualized infrastructure using the global routing table. In most cases, a virtual network is only defined to accommodate the special requirements of a specific type of traffic or policy surrounding a group of users such as "Guest" or "Contractor" and what those users can access. As such the vast majority of an organization's user and device traffic may remain within the global table. Once a specific type of traffic has been migrated to a virtual network, it is possible to "leak" routes between the virtual networks or the global domain to grant specific access to resources within other VNs or the global table. Obviously, it is possible to remove all user/device traffic from the global domain and place it into a virtual network, but a great deal of consideration must be given to providing sufficient access between virtual networks so as to not unintentionally block access to other resources

Network Device Virtualization

One of the characteristics of a VN is that it provides what are essentially private communication paths between members of a group over a shared infrastructure. This creates two requirements for the network infrastructure:

Traffic from one group is never mixed with another—For sending and receiving traffic over shared links, tunnels (many borrowed from existing *virtual private network* [VPN] solutions) can

guarantee data separation. Network devices need to enforce group separation in their internal memory (for example, during routing table lookups, access lists processing, or NetFlow statistics gathering).

Each VN has a separate address space—This requirement is derived from the fact that VNs offer the same characteristics as a physical network. Address space and forwarding within it are two of the most basic aspects of any network.

The first problem to solve is how to virtualize the forwarding plane in a way that meets the requirements for address and traffic flow separation. Depending on the type of device, the virtual separation can go by the following names:

- Virtual LAN (VLAN)
- Virtual routing and forwarding (VRF)
- Virtual forwarding instance (VFI)
- Virtual firewall context

Layer 2: VLANs

VLANs are a good example of a piece of the virtualization puzzle that has been around for quite some time. A VLAN is a logical grouping of ports on a switch that form a single broadcast domain. Ports in a VLAN can communicate only with other ports in the same VLAN. How a given switch does this is implementation dependent, but a common solution is for the switch to tag each frame with a VLAN number as it arrives on a port. When a frame is sent to other ports, the output hardware copies the packet only if it is configured with the VLAN number carried in the frame. On an Ethernet switch, there is typically a single MAC table, which maps ports to MAC addresses. To support VLANs (and simple Layer 2 virtualization), the MAC table has a field for the VLAN

number on which the station was discovered, as demonstrated in Example.

```
Switch# show mac-address-table
...
Non-static Address Table:
Destination Address Address Type VLAN Destination Port
------------------- ------------ ---- --------------------
0010.0de0.e289      Dynamic         1 FastEthernet0/1
0010.7b00.1540      Dynamic         2 FastEthernet0/5
0010.7b00.1545      Dynamic         2 FastEthernet0/5
0060.5cf4.0076      Dynamic         1 FastEthernet0/1
0060.5cf4.0077      Dynamic         1 FastEthernet0/1
0060.5cf4.1315      Dynamic         1 FastEthernet0/1
0060.70cb.f301      Dynamic         1 FastEthernet0/1
00e0.1e42.9978      Dynamic         1 FastEthernet0/1
00e0.1e9f.3900      Dynamic         1 FastEthernet0/1
```

Table No 1. Mac address table

The summary effect of the VLANs is to partition the switch into logical Layer 2 domains. Each domain has its own address space and packets from one domain are kept separate from those of another.

Layer 3: VRF Instances

VRFs are to Layer 3 as VLANs are to Layer 2 and delimit the domain of an IP network within a router. The Cisco website has a more formal definition:

VRF—A VPN Routing/Forwarding instance. A VRF consists of an IP routing table, a derived forwarding table, a set of interfaces that use the forwarding table, and a set of rules and routing protocols that determine what goes into the forwarding table.

Unlike the VLAN scenario, where an extra column in the MAC table is adequate, a VRF partitions a router by creating multiple routing tables and multiple forwarding instances. Dedicated interfaces are bound to each VRF.

Layer 2 VFIs

VFI is a service-specific partition on a switch that associates attachment circuits in the form of VLANs with *virtual switched interfaces* (VSIs). If that did not make much sense, it is useful to have some background on the service itself, namely *Virtual Private LAN Services* (VPLS), to understand VFIs. VPLS is a Layer 2 LAN service offered by *service providers* (SPs) to connect Ethernet devices over a WAN. The customer devices (call them *customer edges* [CEs] for now; we review this in more detail in Chapter 5, "Infrastructure Segmentation Architectures") are all

Ethernet switches. However, the SP uses a Layer 3 network running *Multiprotocol Label Switching* (MPLS) to provide this service. The device on the edge of the SP network is called a *provider edge* (PE). Its role is to map Ethernet traffic from the customer LAN to MPLS tunnels that connect to all the other PEs that are part of the same service instance. The PEs are connected with a full mesh of tunnels and behave as a logical switch, called a VSI. Another way to think about this is to see the VPLS service as a collection of Ethernet ports connected across a WAN. A VSI is a set of ports that forms a single broadcast domain. In many ways, a VSI behaves just as you would expect a regular switch to. When a PE receives an Ethernet frame from a customer device, it first learns the source address, as would any switch, before looking at the destination MAC address and forwarding the frame. If the port mapping for the destination MAC address is unknown, or is a broadcast, the frame is sent to all PEs that are part of the VSI. The PEs use split horizon to avoid creating loops, which in turn means that no spanning tree is needed across the SP network.

Obviously, the previous explanation hides a fair amount of detail, but it should be enough to give a high-level view of what is going on. Once again, there is a need to define and manage groups of isolated ports and tunnels on a switch. The VLAN construct is too limited, and a VRF is strictly a Layer 3 affair, so it is necessary to come up with a new virtual device structure for VPLS, called a VFI.

Virtual Firewall Contexts

Device virtualization is not limited to switches and routers. As a final example, consider a firewall device. For essentially economic reasons, you might want to share a single firewall between multiple different customers or network segments. Each logical firewall needs to have a complete set of policies, dedicated interfaces for incoming and outgoing traffic, and users authorized to manage the firewall.

Many vendors provide this capability today and undoubtedly have their own, well-chosen name for it, but on Cisco firewalls the term context is used to refer to a virtual firewall. Unlike VRFs, VFIs, or VLANs, a context is an emulation of a device (so an example of the VR concept discussed earlier in this chapter).

Firewall contexts are a little unusual in the way they assign a packet to a context. All the partitions we have seen up to now have static assignment of interfaces (you can assign IP packets to a VRF dynamically. We cover that later). A firewall module looks at an incoming packet's destination IP address or Ethernet VLAN tag to decide which context a packet belongs to. All the firewall needs is for one of the two fields to be unique. So, either each context has a unique IP address space on its interfaces or the address space is shared, but each context is in a different VLAN.

Figure shows a simple setup with an Ethernet switch connected to a firewall context using two VLANs. The switch binds the VLANs to VRF BLUE (at the top) and VRF RED. The firewall has two different contexts. The blue one receives all frames on VLAN 101 and the red one gets VLAN 102. In this way, packets from the outside (on the right side of the figure) that belong to VLAN

101 go through a different set of firewall rules than those belong to VLAN 102.

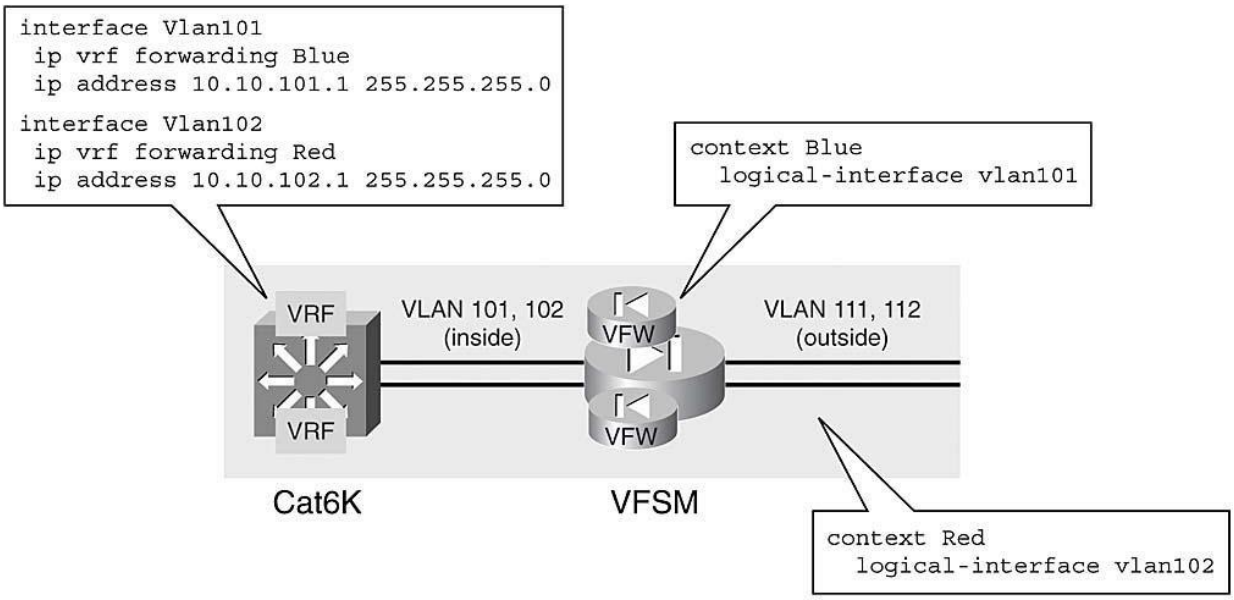


Fig No.24. Virtual firewall context

Data-Path Virtualization

Connecting devices with private paths over a shared infrastructure is a well-known problem. SPs have solved this with different iterations of VPN solutions over the years. Not surprisingly, we can use and adapt many of these same protocols in enterprise networks to create virtualized Layer 2 and Layer 3 connections using a common switched infrastructure. The focus in this section is on the more relevant of the rather overwhelming menu of protocols to build a VPN. Some of this section is a review for many readers, especially the material on 802.1q, *generic routing encapsulation* (GRE), and IPsec, and we do not devote much space to these topics. However, we also include label switching (a.k.a. MPLS) and *Layer 2 Tunnel Protocol Version 3* (L2TPv3), which are probably less familiar and which consequently are covered in more detail.

Layer 2: 802.1q Trunking

You probably do not think of 802.1q as a data-path virtualization protocol. But, the 802.1q protocol, which inserts a VLAN tag on Ethernet links, has the vital attribute of guaranteeing address space separation on network interfaces. Obviously, this is a Layer 2 solution, and each hop must be configured separately to allow 802.1q connectivity across a network. Because a VLAN is synonymous with a broadcast domain, end-to-end VLANs are generally avoided.

VPN Tunneling Protocols

- GRE: Generic Routing Encaptulation (RFC 1701/2)
- L2TP: Layer 2 Tunneling protocol
- IPsec: Secure IP
- MPLS: Multiprotocol Label Switching

GRE

- Generic Routing Encaptulation (RFC 1701/1702)
- Generic Þ X over Y for any X or Y
- Optional Checksum, Loose/strict Source Routing, Key
- Key is used to authenticate the source
- Over IPv4, GRE packets use a protocol type of 47
- Allows router visibility into application-level header
- Restricted to a single provider network Þ end-to-end

GRE provides a method of encapsulating arbitrary packets of one protocol type in packets of another type (the RFC uses the expression X over Y, which is an accurate portrayal of the problem being solved). The data from the top layer is referred to as the payload. The bottom layer is called the delivery protocol. GRE allows private network data to be transported across shared, possibly public infrastructure, usually using point-to-point tunnels.

Although GRE is a generic X over Y solution, it is mostly used to transport IP over IP (a lightly modified version was used in the Microsoft Point-to-Point Tunneling Protocol [PPTP] and, recently, we are seeing GRE used to transport MPLS). GRE is also used to transport legacy protocols, such as Internetwork Packet Exchange (IPX) and AppleTalk, over an IP network and Layer 2 frames.

GRE Header

The second 2 octets of the header contain the payload protocol type, encoded using Internet Assigned Numbers Authority (IANA) Ethernet numbers

The simplest possible expression of a GRE header is a Protocol Type field. All the preceding fields are typically 0, and the subsequent ones can be omitted. You can find freeware implementations that work only with the first 2 octets, but all 4 should be supported.

GRE is purely an encapsulation mechanism. How packets arrive at tunnel endpoints is left entirely up to the user. There is no control protocol, no session state to maintain, no accounting records, and so forth; and this conciseness and simplicity allows GRE to be easily implemented in hardware on high-end systems. The concomitant disadvantage is that GRE endpoints have no knowledge of what is happening at the other end of the tunnel, or even whether it is reachable.

The time-honored mechanism for detecting tunnel reachability problems is to run a dynamic routing protocol across the tunnel. *Routing Protocol* (RP) keepalives are dropped if the tunnel is down, and the RP itself will declare the neighbor as unreachable and attempt to route around it. You can lose a lot of data waiting for an RP to detect a problem in this way and reconverge. Cisco

added a keepalive option to its GRE implementation. This option sends a packet through the tunnel at a configurable period. After a certain number of missed keepalives (the number is configurable), the router declares the tunnel interface as down. A routing protocol would detect the interface down event and react accordingly.

GRE's lack of control protocol also means that there is essentially no cost to maintaining a quiescent tunnel active. The peers exchange no state information and must simply encapsulate packets as they arrive. Furthermore, like all the data-path virtualization mechanisms we discuss, the core network is oblivious of the number of tunnels traversing it. All the work is done on the edge.
We do not want to suggest that GRE is the VPN equivalent of a universal solvent. There is a cost to processing GRE—encapsulation/decapsulation, route lookup, and so forth—but it's in the data path.

GRE IOS Configuration

On Cisco devices, GRE endpoints are regular interfaces. This seemingly innocuous statement is replete with meaning, because anything in Cisco IOS that needs to see an interface (routing protocols, access lists, and many more) will work automatically on a GRE tunnel.

Example R103 GRE Configuration interface Tunnel0

ip address 40.0.0.1 255.255.255.0 tunnel source Serial1/0

tunnel destination 192.168.2.1

The tunnel source and tunnel destination addresses are part of the transport network address space. They need to match on both endpoints so that a source address on one router is the destination address on the remote device. The router must also have a path in its routing table to the tunnel destination address. The next hop to the tunnel destination must point to a real interface and not the tunnel interface. In this case, the router has a tunnel interface with tunnel destination of 192.168.2.1 on the public network. The 40.0.0.0/24 network used for the tunnel IP's address, however, is part of the private address space used on Sites 1 and 2.

IPsec

IPsec provides a comprehensive suite of security services for IP networks. IPsec was originally conceived to provide secure transport over IP networks. The security services include strong authentication (Authentication Header [AH]) and Encryption (Header [EH]) protocols and ciphers and key-exchange mechanisms. IPsec provides a way for peers to interoperate by negotiating capabilities and keys and security algorithms.

IPsec peers maintain a database of security associations. A *security association* (SA) is a contract between peers, which defines the following:

The specific encryption and authentication algorithms used, such as Triple DES (*Triple Data Encryption Standard*)

The IPsec protocol service (*Encapsulating Security Payload* [ESP] or AH) Key material needed to

communicate with the peer

The SA is negotiated when an IPsec session is initiated. Each IPsec header contains a unique reference to the SA for this packet in a *Security Parameter Index* (SPI) field, which is 32-bit numeric reference to the SA needed to process the packet. Peers maintain a list of SAs for inbound and outbound processing. The value of the SPI is shared between peers. It is one of the things exchanged during IPsec session negotiation.

At the protocol level, there are two IPsec headers:

AH—Offers nonrepudiatable authentication between two parties. The authentication service also provides for message integrity and certain instances of (identity) spoofing.

ESP—Offers encrypted communication between two parties. The encryption service allows message confidentiality, integrity, nonrepudiation, and protection against spoofing and replay attacks.

It is possible to use authentication and encryption services separately or together. If used in combination, the AH header precedes the ESP header.

There are two ways to encapsulate IPsec packets. The first, called tunnel mode, encrypts an entire IP packet, including the header, in the IPsec payload.

IPsec Transport Mode Stack

IPsec requires a lot of negotiation to bring up a session. So much so that there is a separate control channel protocol, called Internet Key Exchange (IKE), used to negotiate the SA between peers and exchange keys material. Note that IKE is not mandatory; you can statically configure the SAs.

IKE is not only used during tunnel setup. During confidential data exchange, the session keys used to protect unidirectional traffic may need to be changed regularly, and IKE is used to negotiate new keys.

IKE traffic itself is encrypted, and, in fact, it has its own SA. Most of the parameters are fixed as follows:

56- bit DES for encryption

Message digest 5 (MD5) algorithm or secure hash algorithm (SHA) hashing Rivest, Shamir,

Adleman (RSA) (public key) signatures or preshared keys IKE runs on UDP/500. IPsec uses IP

protocol values of 50 and 51.

Cisco IOS IPsec Configuration

There is a lot more to IPsec than you will see here, but there are three basic parts to the configuration, which correspond to setting up SAs first for IKE, and then for the session itself, and defining which traffic to encrypt. The steps of the configuration are as follows:

The first basic part is the IKE policy. IKE will negotiate its own SA with the remote peer, so it too needs a policy. The crypto isakamp policy command defines the type of authentication, and the IP address of the remote peer and the shared secret used to protect the IKE exchanges, as indicated in the below Example

Example IKE Policy Settings **crypto isakmp policy 1 authentication pre-share**

crypto isakmp key secret address 10.0.3.11

In the second basic part is a crypto map, the role of the crypto map is to define the remote peer, the encryption and authentication algorithms (called transforms) that this router will accept to set up a SA, and the interesting traffic to be encrypted. As in so much of Cisco IOS, interesting traffic is defined using standard access lists. If a packet matches an access list entry, whatever IPsec policy is defined in the crypto map is applied to the packet. Example 4-9 has a crypto map that configures any traffic to address 10.0.3.11 that matches access list 101 to be encrypted using the IPsec service called ONE.

Example IPsec Crypto Map **crypto map VPN 1 IPsec-isakmp set peer 10.0.3.11**

set security-association lifetime seconds 180 set transform-set ONE

match address 101

The authentication and encryption algorithms for this SA are defined in a transform set (so they can be shared between multiple SA definitions). The transform set is given in Example below It specifies AH and ESP services, with MD5 for authentication and DES for encryption.

Example IPsec Transform Set

crypto ipsec transform-set ONE ah-md5-hmac esp-des

The third step is to apply the crypto map on an outgoing interface, as in Example 4-11. This completes the puzzle. Now when packets enter or leave the Serial0 interface on this router, they are compared against access list 101 and if there is a match, encrypted according to the service defined in above Example.

L2TP

- Layer 2 Tunneling Protocol

- L2F = Layer 2 Forwarding (From CISCO)
- L2TP = L2F + PPTP
- Combines the best features of L2F and PPTP
- Easy upgrade from L2F or PPTP
- Allows PPP frames to be sent over non-IP (Frame relay, ATM) networks also (PPTP works on IP only)
- Allows multiple (different QoS) tunnels between the same end-points. Better header compression. Supports flow control

L2TPV3

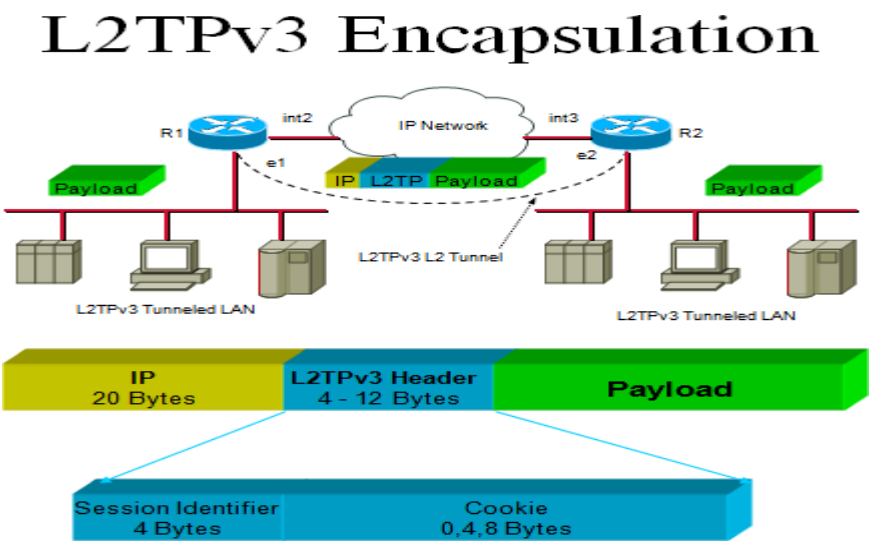


Fig No.25 L2TPv3 Encapsulation

- Allows service providers to offer L2 VPN over IP network.
- L2TPv2 was for tunneling PPP over packet switched data networks (PSDN)
- V3 generalizes it for other protocols over PSDN Þ PPP specific header removed
- Can handle HDLC (High-Level Data Link Control), Ethernet, 802.1Q VLANs, Frame relay, packet over SONET (Synchronous Optical Network)
- Universal Transport Interface (UTI) is a pre-standard effort for transporting L2 frames.
- L2TPv3 extends UTI and includes it as one of many supported encapsulations.
- L2TPv3 has a control plane using reliable control connection for establishment, teardown and maintenance of individual sessions.

The L2TPv3 protocol consists of components to bring up, maintain, and tear down sessions, and the capability to multiplex different Layer 2 streams into a tunnel.

The L2TP protocol has a both a control and data plane. The control channel is reliable. There are 15 different control message types. The major ones are for the setup and teardown of the control channel itself (see Appendix A for more detail). L2TPv3 peers can exchange capability information for the session during the setup phase. The most important of these are the session ID and cookie.

The session ID is analogous to the control channel identifier and it is a "shortcut" value that the receiver associates with the negotiated context for a particular session (for instance, payload type, cookie size, and so forth).

The cookie is an optional, variable-length field of up to 64 bits. The cookie is a cryptographically random number that extends the session identifier space so as to ensure there is little chance that a packet is misdirected because of corrupt session ID. 264 is a large number and, as long as it is random, the cookie makes L2TPv3 impervious to brute-force spoofing attacks, where the attacker tries to inject packets into an active session.

After a session is established through the control session, the L2TP endpoint is ready to send and receive data traffic. Although the data header has a Sequence Number field, the data channel is not reliable. The protocol can detect missing, duplicate, or out-of-order packets, but does not retransmit. That is left to higher-layer protocols.

The RFC allows for the data channel to be set up either using the native control protocol, or statically, or using another control mechanism.

In the design sections after Chapter 5, "Infrastructure Segmentation Architectures: Theory," you will see occasions when, frankly, GRE could solve a problem just as well as L2TPv3. What then are the differences between these two protocols? Following is a list of them:

Ubiquity—GRE can be found just about everywhere. It is an old (in Internet terms anyway), well-established protocol, and implementations should, by now, be robust. L2TPv3, more recent, is less prevalent.

Performance—On high-speed links, especially on enterprise networks, encapsula-tion tax (header length and so forth) is much less of an issue than a couple of decades ago, when trying to wring every last ounce of baud rate from 1200 bps links was an important issue for network administrators the world over. At Gigabit, or 10 Gigabit speeds, the number of bytes used by a well-designed protocol is not really an issue, as long as the implementation runs in hardware. Concerning this last point, it is probably easier to find hardware implementations of GRE than L2TPv3.

Payload protocols—RFC 3931 specifically states that L2TPv3 is designed to carry Layer 2

protocols. GRE is a multipurpose solution that can carry any other protocol. However, the devil is in the details, and GRE "implementations" may be limited to specific protocols (such as just Ethernet or IP). Furthermore, L2TPv3 has been extended to carry IP traffic.

Cookie—This is the most fundamental difference between the two protocols. GRE has no equivalent of the Cookie field. If this is not important to you—and recall that the main advantage is to provide guarantees against spoofing—implementation issues may dictate your choice more than any difference between the protocols themselves.

L2TPv3 IOS Configuration

There are three things to configure for the L2TPv3 IOS configuration: Control channel parameters

Data channel parameters Connection circuit parameters

To configure the first of these parameters, use the **l2tp-class** command for control channel setup. Here, you can change sequence number settings and so on, but the minimum required is the shared password known to both peers. Example demonstrates the use of this command.

Example l2tp-class *Command*

```
l2tp-class L2WAN password 7 00071A150754
```

As in classic L2TP setup, if you do not give a *hostname* parameter, the device name is used.

The second part of the configuration is for the data channel. Cisco IOS uses the **pseudowire** command, which is a generic template also used for Layer 2 over MPLS (called AToM) setup. The **pseudowire-class** specifies the encapsulation and refers to the control channel setup with the **protocol l2tpv3** *name* command (if you omit this, default control channel settings are used). The **pseudowire-class** also contains the name of the interface used as the source address of the L2TPv3 packets.

Example *L2TP* pseudowire-class *Command*

```
pseudowire-class R103R104 encapsulation l2tpv3 protocol l2tpv3 L2WAN

ip local interface Serial1/0
```

L2TPv3 Topology

The final part of the configuration (see Example 14-14) binds the client-facing attachment circuit to the trunk port using the **xconnect** command (already introduced in the discussion on VPLS earlier in this section). The **xconnect** command defines the remote peer IP address and a unique *virtual circuit* (VC) identifier used on each peer to map the L2TPv3 payload to the correct attachment circuit. The L2TPv3 endpoints negotiate unique session and cookie ID values for each

VC ID, as shown in Figure 4-9. You must configure a different VC ID for each VLAN, port, or *data-link connection identifier* (DLCI) transported across an L2TPv3 tunnel (currently, Cisco L2TPv3 supports Ethernet, 802.1q [VLAN], Frame Relay, *High-Level Data Link Control* [HDLC], and PPP).

Example xconnect *Command* **interface Ethernet0/0 description Client Facing Port no ip address**

no cdp enable

xconnect 192.168.2.1 103 encapsulation l2tpv3 pw-class R103R104

It's interesting that although the second and third versions of protocol differ in relatively small ways, the *command-line interface* (CLI) configuration differs significantly from the standard *L2TP access concentrator / L2TP network server* (LAC/LNS) configuration that you might have used for dialup or *digital subscriber line* (DSL) networks. However, there are obvious, and deliberate, similarities with other pseudowire solutions such as *Ethernet over* MPLS (EoMPLS).

Label Switched Paths

Label switched paths (LSPs) are an interesting hybrid of all the preceding data-path solutions: a Layer 2 data path with Layer 3 control plane. Of course, LSPs are found in MPLS networks, which is a topic that has generated entire library shelves of books and other documents. In this chapter, we present a short review of how packets traverse an MPLS network. We do not cover label distribution or any of the major MPLS applications, such as VPN or traffic engineering (MPLS VPNs are discussed in depth in Chapter 5, however).

What we are going to cover may be summarized as follows:

An LSP is a tunnel across an MPLS network made up of individual hop-to-hop segments. MPLS

networks uses the IP control plane.

LSPs are set up for all known IP prefixes in the IP routing table. LSPs are multiplexed across

physical links.

Each node in an MPLS network forwards based on fixed-length labels instead of variable-length prefixes.

Labels are carried in a shim header, between the Layer 2 and Layer 3 headers. Nodes distribute

labels to adjacent nodes using a label distribution protocol.

Basic label switching is easy to configure

Label switching must be configured on all hops.

In a normal routing scenario, when a router needs to forward a packet, it finds the outgoing

interface by looking for a matching IP address prefix in the routing table. The actual interface used for forwarding corresponds to the shortest path to the IP destination, as defined by the routing policy. Other administrative policies, such as QoS and security, may affect the choice of interface. This collection of criteria used for forwarding decisions is more generally referred to as a Forward Equivalency Class (FEC). The classification of a packet to FEC is done on each router along the IP path and happens independently of the other routers in the network.

MPLS decouples packet forwarding from the information in the IP header. An MPLS router forwards packets based on fixed-length labels instead of matching on a variable-length IP address prefix. The label is a sort of shortcut for an FEC classification that has already happened. Where the label comes from is discussed later in this section, but for now, it is enough to say that the labels are calculated based on the topology information in the IP routing table. RFC 3031 puts it like this:

In MPLS, the assignment of a particular packet to a particular FEC is done just once, as the packet enters the network. The FEC to which the packet is assigned is encoded as a short fixed length value known as a "label." When a packet is forwarded to its next hop, the label is sent along with it; that is, the packets are "labeled" before they are forwarded.

In the MPLS forwarding paradigm, once a packet is assigned to a FEC, no further header analysis is done by subsequent routers; all forwarding is driven by the labels.

Before looking at this in more detail, we need to introduce some definitions:

Label switching router (LSR)—A router that switches based on labels. An LSR swaps labels. Unlike a traditional router, an LSR does not have to calculate where to forward a packet based on the IP packet header (which is a simplified way of saying it does not do FEC classification when it receives a packet). An LSR uses the incoming label to find the outgoing interface (and label). LSRs are also called provider (P) routers.

Edge LSR—A router that is on the edge of an MPLS network. The edge LSR adds and removes labels from packets. This process is more formally called imposition and disposition (and also pushing and popping, because labels are said to go on a stack). Edge LSRs are often referred to as provider edge (PE) routers.

Customer edge (CE)—An IP router that connects to the PE device. The CE performs IP forwarding. The PE and CE form routing protocol adjacencies.

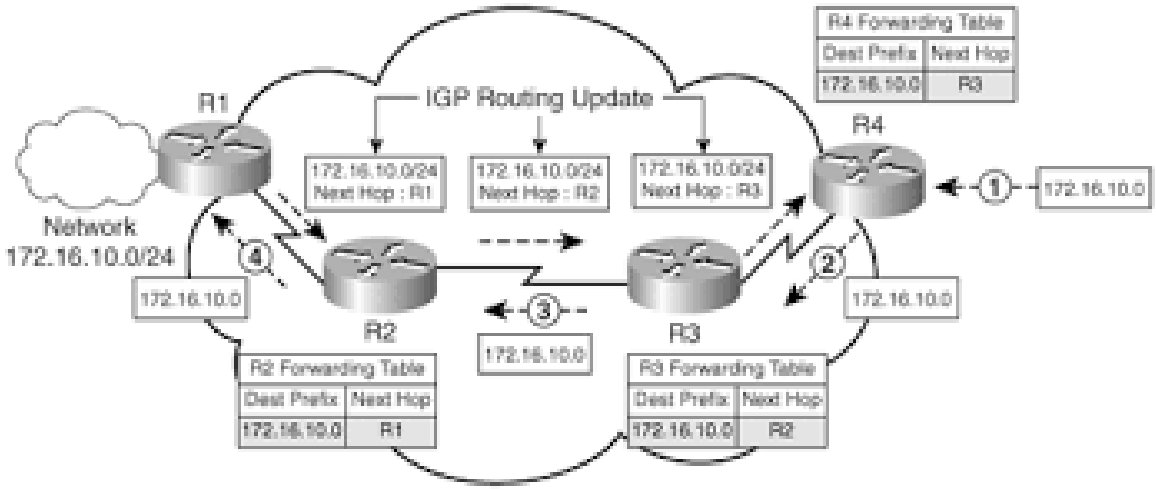


Fig No. 26 : MPLS Forwarding

As a packet flows across the network shown in Figure , it is processed by each hop as follows:

At the edge of the network, as shown in Figure, edge LSR A classifies a packet to its FEC and assigns (or imposes) label 17 to the packet. A label is of local significance on that interface just like an ATM VPI/VCI or a Frame Relay DLCI.

In the core, LSRs, such as LSR C and LSR B, swap label values. LSR C removes the old label, 17 in the example shown in Figure, and imposes the new one, 22. The values of the ingress label and interface are used to find the values of the egress label and interface.

LSR B, as the second-last hop in the MPLS network, removes the outermost label from the label stack, which is called penultimate hop popping (PHP). So, packets arrive at edge LSR D without any label, and standard IP routing is used to forward the packet. The process of removing a label is also called disposition. PHP avoids recursive lookups on edge LSR D.

After the label is removed, the packet is forwarded using standard IP routing.

Now the difference with standard IP forwarding should be clearer. FEC classification is done when a packet enters the MPLS network, not at every hop. An LSR needs to look only at the packet's label to know which outgoing interface to use. There can be different labels on an LSR for the same IP destination. Saying the same thing in a different way, there can be multiple LSPs for the same destination.

A key point to understand is that the control plane is identical in both the IP and MPLS cases. LSRs use IP routing protocols to build routing tables, just as routers do. An LSR then goes the extra step of assigning labels for each destination in the routing table and advertising the label/FEC mapping to adjacent LSRs. ATM switches can also be LSRs. They run IP routing protocols, just as a router LSR does, but label switch cells rather than packets.

What is missing from this description is how label information is propagated around the network.

How does LSR A in Figure 4-10 know what label to use? MPLS networks use a variety of signaling protocols to distribute labels:

LDP—Used in all MPLS networks iBGP—Used for L3 VPN service RSVP—Used for Traffic

Engineering

Directed LDP—Used for L2VPN service, such as VPLS

Label Distribution Protocol (LDP), which runs over tcp/646, is used in all MPLS networks to distribute labels for all prefixes in the nodes routing table. Referring again to Figure 4-10, LSR D and LSR B would bring up a LDP session (LSR B would have another session with LSR C and so forth). LSR D is connected to the customer 192.168.2.0/24 network and advertises this prefix to all its routing peers. LSR D also sends a label to LSR B for the 192.168.2.0 network. When LSR B's routing protocol converges and it sees 192.168.2.0 as reachable, it sends label 22 to LSR C. This process continues until LSR A receives a label from LSR C.

The complete end-to-end set of labels from LSR A to LSR D form an LSP. An LSP is unidirectional. There is another LSP, identified by a different set of labels, for return traffic from LSR D to LSR A.

Understand that two operations must complete for the LSP from LSR A to 192.168.2.0 to be functional:

The backbone routing protocol must converge so that LSR A has a route to 192.168.2.0. LDP must

converge so that labels are propagated across the network.

Figure 4-10 does not show a numeric value for the label between LSR B and LSR D. In fact, as already discussed, the packet on this link has no label at all, because of PHP. Never-theless, LSR D does still advertise a special value in LDP, called an implicit null (which has a reserved value of 3), so that LSR B performs PHP.

After LSR A has all the information it needs to forward data across the MPLS network, it encapsulates outgoing packets in a shim header, shown in Figure 4-11 and defined in RFC 3032, which is inserted between the Layer 2 and Layer 3 headers. Encapsulation stacks are defined in different RFCs for Ethernet, ATM, PPP, and other media.

Data-Path Virtualization Summary

We presented several different protocols that can be used for data-path virtualization. Two of them are suitable for Layer 2 traffic only: 802.1q, which is configured on each hop, and L2TPv3 which is configured end to end. IPsec is suitable for IP transport. Finally, GRE and MPLS LSPs can be used for either Layer 2 or Layer 3. GRE is another IP tunnel protocol, configured only on endpoints. MPLS creates a new forwarding path and is configured on all hops in a network.

Control-Plane Virtualization—Routing Protocols

Data-path virtualization essentially creates multiple separate logical networks over a single, shared physical topology. To move packets across these VNs, you need to need a routing protocol.

The most familiar virtualized control plane is probably *Per VLAN Spanning Tree* (PVST), which has a separate spanning-tree instance for each VLAN running on a switch. Even through PVST has been around longer than the term *virtualization*, it illustrates the central point we are making here very crisply. Different logical networks have different topologies and, therefore, different optimal paths. Switches have to run different spanning-tree calculations for each such network.

The remainder of this section deals with extensions to routing protocols to allow them to run with multiple routing instances. However, we will return to the topic of control-plane virtualization, because many different router and switch functions, such as NetFlow, DHCP, RADIUS, and so on, need to receive the same treatment and become VRF aware.

VRF-Aware Routing

Cisco's major *interior gateway protocol* (IGP) routing protocol implementations are VRF aware. This means that they understand that certain routes may be placed only in certain routing tables. The routing protocols manage this by peering within a constrained topology, where a routing protocol instance in a VRF peers with other instances in the same VN. No special information is added to the route advertisements to identify VRF names, so routing instances must communicate over private links.

With some protocols (for example, BGP), a single routing instance can manage multiple VRF tables; with others (for example, OSPF), a different routing process runs for every VRF. Remember that in both cases, every VRF requires a route optimization calculation, so increasing the number of VRFs does have a computational impact on a network device.

VRF per Process: OSPF

OSPF has a different routing process for each VRF. The first implementation was rather strict, with a maximum of 32 processes. Furthermore, two processes are reserved for static and connected routes. Recent software enhancements lift this limitation. You are now limited to 32 VRFs per process, but the number of processes is now fixed by the network devices' CPU and memory limitations (300 to 10,000 depending on the platform).

Example shows how an OSPF process is associated with the RED VRF. The networks advertised by this process should be in the same VRF.

Example *Per-VRF OSPF Configuration*

```
router ospf 2000 vrf RED log-adjacency-changes
```

network 20.0.0.0 0.0.0.255 area 0

network 40.0.0.0 0.0.0.255 area 0

VRF Address Families: EIGRP, RIP, and BGP

For the other routing protocols, a single process can manage all the VRFs, and the Cisco IOS **address-family** command is used to configure per-VRF route policy. Example 4-20 shows how to configure RIP for two VRFs, RED and GREEN. Each VRF has overlapping entries for network 13.0.0.0.

Example *Per-VRF RIP Configuration*

router rip version 2

!

address-family ipv4 vrf RED version 2

network 11.0.0.0

network 13.0.0.0 no auto-summary exit-address-family

!

address-family ipv4 vrf GREEN version 2

network 12.0.0.0

network 13.0.0.0 no auto-summary exit-address-family

!

Example shows an iBGP per-VRF Configuration.

Example *Per-VRF iBGP Configuration*

router bgp 100

no synchronization

bgp log-neighbor-changes no auto-summary

!

address-family ipv4 vrf RED redistribute connected neighbor 14.0.0.1 remote-as 100

neighbor 14.0.0.1 update-source loopback100 neighbor 14.0.0.1 activate

no auto-summary

no synchronization exit-address-family

!

address-family ipv4 vrf GREEN redistribute connected

neighbor 15.0.0.1 remote-as 100

neighbor 15.0.0.1 update-source loopback200 neighbor 15.0.0.1 activate

no auto-summary no synchronization exit-address-family

!

Multi-Topology Routing

Multi-Topology Routing (MTR) is a recent innovation at Cisco. As the name suggests, it creates multiple routing topologies across a shared, common infrastructure. However, MTR does not try to be yet another VPN solution. Instead, it creates paths through a network that you can map to different applications or classes of applications, with the understanding that, by separating traffic in this way, you can provide better performance characteristics to certain critical applications.

MTR bases its operation on the creation of separate RIBs and FIBs for each topology. The separate RIBs and FIBs are created within a common address space. Thus, MTR creates smaller topologies that are a subset of the full topology (also known as the base topology). The main difference between MTR and a VPN technology is that, with MTR, a single address space is tailored into many topologies that could overlap; whereas VPNs create totally separate and independent address spaces.

Thus, MTR must carry out two distinct functions:

At the control plane—Color the routing updates, so that the different topology RIBs are populated accordingly. Based on these RIBs, the corresponding FIBs are to be written.

At the forwarding plane—Identify the topology to which each packet belongs and use the correct FIB to forward the packet.

At each hop, there will be a set of prefixes and routes in the RIB for each topology. The contents of these RIBs are dynamically updated by routing protocol colored updates. Based on this RIB information, a separate FIB is built for each topology.

To forward traffic over different topologies, the router looks for a code point in each packet and chooses an FIB based on this code point. A first implementation of MTR uses *differentiated services* (DiffServ) *code point* (DSCP) as such a code point, but other code points could be used by future implementations. The DSCP value is used as a pointer to the correct forwarding table, and the packet's destination address is used to make a forwarding decision based on the information in the topology's FIB. MTR uses the terminology of color to refer to separate topologies. So, a RED value in a packet's DSCP field is recognized by the router, which will forward the packet

using the RED forwarding table (FIB).

MTR must run contiguously across a network, and the color mappings must be consistent (that is, you cannot use DSCP X as Green on one hop but as Red on the next). MTR does not allow you to double dip: If the destination route is not in the routing table of the color a packet is using, the packet can either be dropped or forwarded over the base topology—there are no lookups in "backup topologies" other than the base topology (which is equivalent to regular routing).

MTR does not change how routing works; it just runs across multiple topologies (using a single process with colored updates).

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – IV – VIRTUALIZATION TECHNIQUES SCSA7022

UNIT IV VIRTUALIZING STORAGE

SCSI - Speaking SCSI - Using SCSI buses - Fiber Channel - Fiber Channel Cables - Fiber Channel Hardware Devices - iSCSI Architecture - Securing iSCSI - SAN backup and recovery techniques - RAID - SNIA Shared Storage Model - Classical Storage Model - SNIA Shared Storage Model - Host based Architecture - Storage based architecture - Network based Architecture - Fault tolerance to SAN - Performing Backups - Virtual tape libraries.

VIRTUALIZATION

Virtualization provides a simple and consistent interface to complex functions. There is no need to understand the underlying complexity. For example, when driving an automobile the driver doesn't need to understand the workings of the internal combustion engine, the accelerator pedal virtualizes that operation. Virtualization is a technique of abstracting physical resources into logical view. It Increases utilization and capability of IT resource. Simplifies resource management by pooling and sharing resources and Significantly reduce downtime. Produces Improved performance of IT resources.

TYPES OF VIRTUALIZATION

Virtualization Comes in Many Forms and is classifies as below:

a. **Virtual Memory** - Each application sees its own logical memory, independent of physical memory

b. **Virtual Networks** - Each application sees its own logical network, independent of physical network

c. **Virtual Servers** - Each application sees its own logical server, independent of physical servers

d. **Virtual Storage** - Each application sees its own logical storage, independent of physical storage.

STORAGE VIRTUALIZATION

It is the process of presenting a logical view of physical storage resources to hosts. Logical storage appears and behaves as physical storage directly connected to host. Benefits of storage virtualization are Increased storage utilization, Adding or deleting storage without affecting application’s availability, Non-disruptive data migration.

DISK ARRAYS

In data centres disks are not available inside the computers. They all are external to the server in Disk Array. So that Data are easily accessible by other servers in case of a server failure. JBODs - Just a bunch of disks are not available in data centres since they are difficult to manage. Disk Arrays are easy to manage pool of disks with redundancy. The structure and component of Disk Array is represented below:

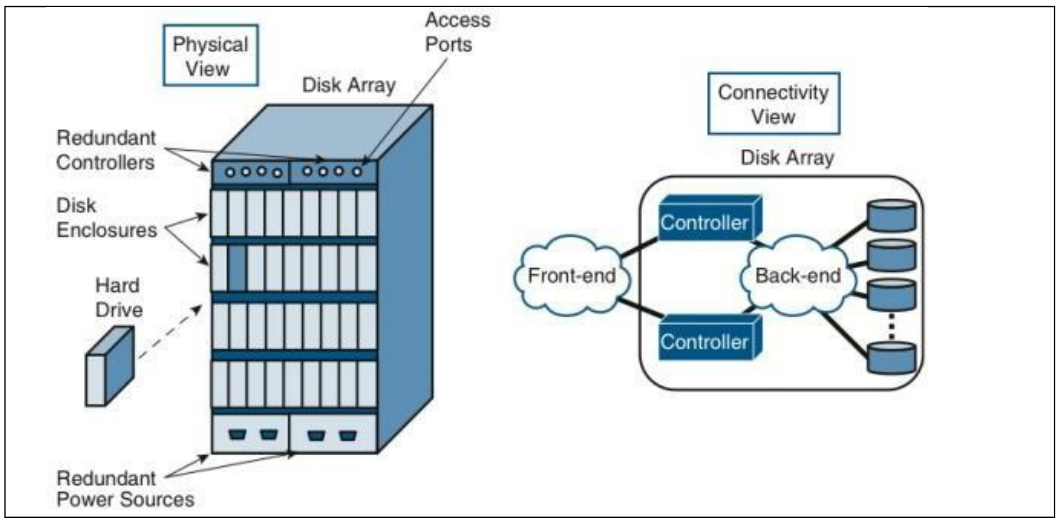


Fig No.27 structure and component of Disk Array

STRUCTURE AND COMPONENT OF DISK ARRAY DATA ACCESS METHODS

Three ways for applications to access data are

1. **Block Access**: A fixed number of bytes (block-size), e.g., 1 sector, 4 sectors, 16 sectors

2. **File Access**: A set of bytes with name, creation date, and other meta data. May or may not be contiguous. A file system, such as, FAT-32 (File Allocation Table) or NTFS (New Technology File System) defines how the meta-data is stored and files are organized. File systems vary with the operating systems.

3. **Record Access**: Used for highly structured data in databases. Each record has a particular format and set of fields. Accessed using Structured Query Language (SQL), Open Data Base Connectivity (ODBC), Java Data Base Connectivity (JDBC). Storage systems provide block access. A logical volume manager in the OS provides other "virtual" views, e.g., file or record

STORAGE VIRTUALIZATION CONFIGURATION

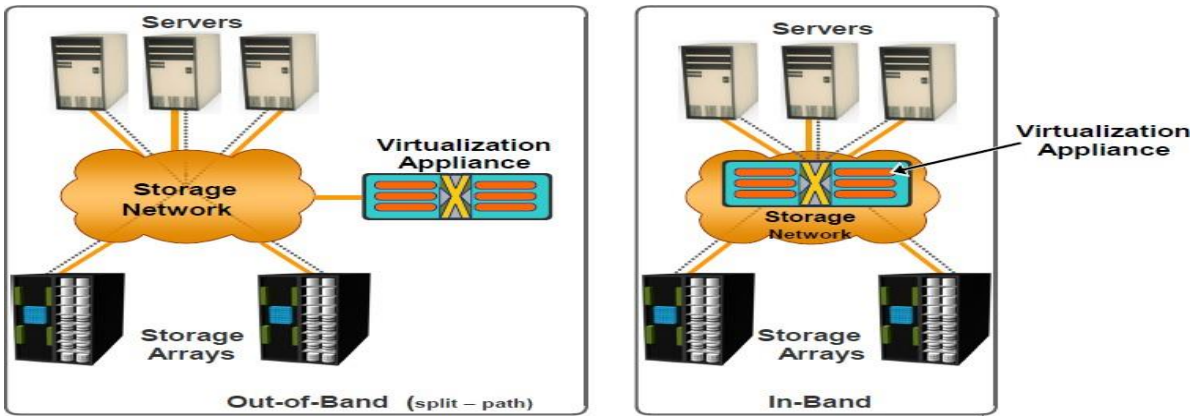


Fig No.28 Storage virtualization Configuration

Out-of-band implementation:

-Virtualized environment configuration is stored external to the data path
-Virtualization appliance is hardware-based and optimized for Fibre channel
-Enables data to be processed at network speed
-More scalable

In-band implementation:

-Virtualization function is placed in the data path
-Virtualization appliance is software-based and runs on general-purpose servers
-During processing, data storing and forwarding through the appliance results in additional latency
-Less scalable – only suitable for static environment with predictable workloads.

STORAGE VIRTUALIZATION CHALLENGES:

1. **Scalability -**Without virtualization, each storage array is managed independently to meet application requirements in terms of capacity and IOPS. With virtualization, the environment as a whole must be analyzed

2. **Functionality -**Virtualized environment must provide same or better functionality. Must continue to leverage existing functionality on arrays

3. **Manageability -** Virtualization device breaks end-to-end view of storage infrastructure. Must integrate with existing management tools

4. **Support -** Interoperability in multivendor environment.

SCSI (SMALL COMPUTER SYSTEM INTERFACE):

SCSI , the Small Computer System Interface, is a set of American National Standards Institute (ANSI) standard electronic interfaces that allow personal computers (PCs) to communicate with peripheral hardware such as disk drives, tape drives, CD-ROM drives, printers and scanners faster and more flexibly than previous parallel data transfer interfaces.

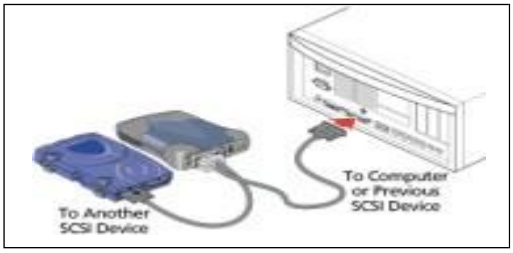


Fig No.29 SCSI

SCSI CONNECTIONS

Used to connect disk drives and tapes to computer. 8-16 devices can be connected on a single bus.

with any number of hosts on the bus at least one host must be with host bus adapter (HBA). Standard commands, protocols, and optical and electrical interfaces are used.

Peer-to-peer: supports host-to-device, device-to-device, host-to-host connections. But most devices implement only targets. Can't be initiators. Each device on the SCSI bus has a "ID". Each device may consist of multiple logical units (LUNs).LUNS are like apartments in a building. A direct access (disk) storage is addressed by a Logical Block Address (LBA). Each LB is typically 512 bytes. Initially used a parallel interface (Parallel SCSI) is Skew. Now Serial Attached SCSI (SAS) is used for higher speed.

Common SCSI components

There are several components used in SCSI storage systems as follows:

Initiator: An initiator issues requests for service by the SCSI device and receives responses. Initiators come in a variety of forms and may be integrated into a server's system board or exist within a host bus adapter.

Target: A SCSI target is typically a physical storage device (although software-based SCSI targets also exist). The target can be a hard disk or an entire storage array. It is also possible for non-storage hardware to function as a SCSI target. Although rare today, it was once common for optical scanners to be attached to computers through the SCSI bus and to act as SCSI targets.

Service delivery subsystem: The mechanism that allows communication to occur between the initiator and the target; it usually takes the form of cabling.

Expander: Only used with serial-attached SCSI (SAS); allows multiple SAS devices to share a single initiator port.

USING SCSI BUSES

SCSI supports faster data transfer rates than the commonly used IDE storage interface. SCSI also supports daisy-chaining devices, which means several SCSI hard drives can be connected to single a SCSI interface, with little to no decrease in performance.

The different types of SCSI interfaces are :

SCSI-1: Uses an 8-bit bus, supports data transfer speeds of 4 MBps.

SCSI-2: Uses a 50-pin connector instead of a 25-pin connector, and supports multiple devices. It is one of the most commonly used SCSI standards. Data transfer speeds are typically around 5 MBps.

Wide SCSI: Uses a wider cable (168 cable lines to 68 pins) to support 16-bit data transfers.

Fast SCSI: Uses an 8-bit bus, but doubles the clock rate to support data transfer speeds of 10

MBps.

Fast Wide SCSI: Uses a 16-bit bus and supports data transfer speeds of 20 MBps.

Ultra SCSI: Uses an 8-bit bus, supports data rates of 20 MBps.

SCSI-3: Uses a 16-bit bus, supports data rates of 40 MBps. Also called Ultra Wide SCSI.

Ultra2 SCSI: Uses an 8-bit bus, supports data transfer speeds of 40 MBps.

Wide Ultra2 SCSI: Uses a 16-bit bus, supports data transfer speeds of 80 MBps.

Ultra3 SCSI: Uses a 16-bit bus, supports data transfer rates of 160 MBps. Also known as Ultra-160.

Ultra-320 SCSI: Uses a 16-bit bus, supports data transfer speeds of 320 MBps.

Ultra-640 SCSI: Uses a 16-bit bus, supports data transfer speeds of 640 MBps.

Examples of commonly-used SCSI connectors on computers and devices are given below.

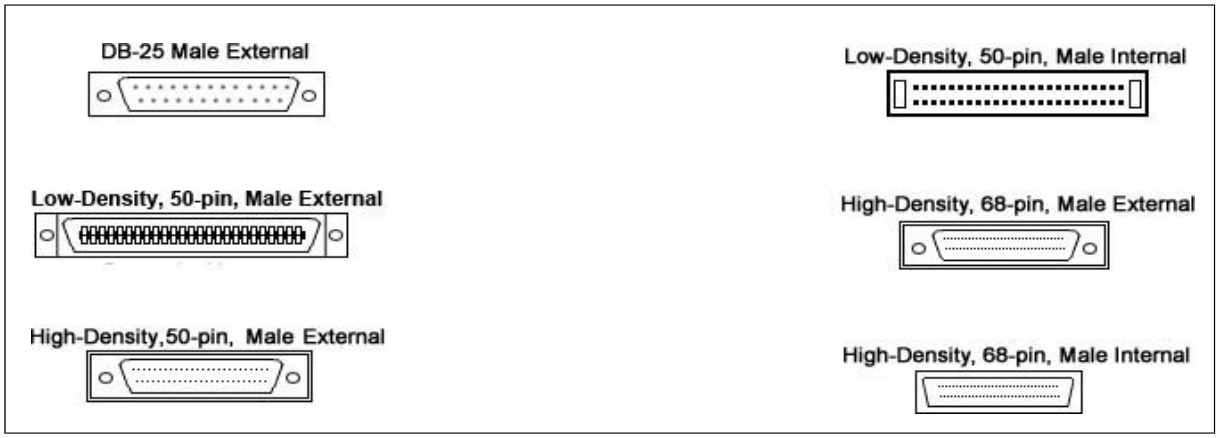


Fig No.30 commonly-used SCSI connectors on computers and devices

COMMONLY-USED SCSI CONNECTORS

while SCSI is still used for some high-performance equipment, newer interfaces have largely replaced SCSI in certain applications. For example, Firewire and USB 2.0 have become commonly used for connecting external hard drives. Serial ATA, or SATA, is now used as a fast interface for internal hard drives.

ADVANCED TECHNOLOGY ATTACHMENT (ATA)

Advanced Technology Attachment (ATA) is a standard physical interface for connecting storage devices within a computer. ATA allows hard disks and CD-ROMs to be internally connected to the motherboard and perform basic input/output functions. various versions are listed below:

Parallel Advanced Technology Attachment (PATA):

-Designed in 1986 for PCs. Controller integrated in the disk

-Integrated Device Electronics (IDE).

-133 Mbps using parallel ribbon cables

ATA Packet Interface (ATAPI):

- Extended PATA to CDROMS, DVD-ROMs, and Tape drives

Serial Advanced Technology Attachment (SATA):

- Designed in 2003 for internal hard disks. 6 Gbps.

PATA Enhancements: ATA-2 (Ultra ATA), ATA-3 (EIDE)

SATA Enhancements: external SATA (eSATA), mini SATA (mSATA)

FIBER CHANNEL:

Fibre Channel is a high-speed network technology used to connect servers to data storage area networks. Fibre Channel technology handles high-performance disk storage for applications on many corporate networks, and it supports data backups, clustering, and replication.

ANSI T11 standard for high speed storage area network(SAN) and Can run on TP or fiber with 2, 4, 8, 16, 32 GBps. It has three topologies namely point-to-point, arbitrated loop (ring), switched fabric.

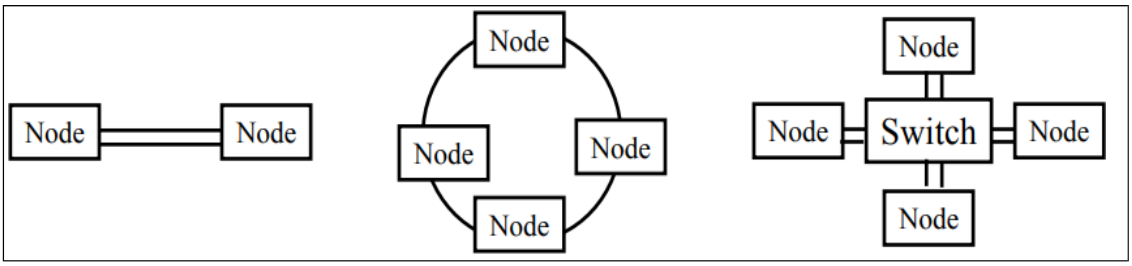


Fig No.31 Fiber Channel

TOPOLOGIES OF FIBER CHANNEL

FC host bus adapters (HBA) have a unique 64-bit World Wide Name (WWN) similar to 48-bit Ethernet MAC addresses with OUI, and vendor specific identifiers (VSID), e.g., 20000000C8328FE6. Several different network addressing authorities (NAA) exists.

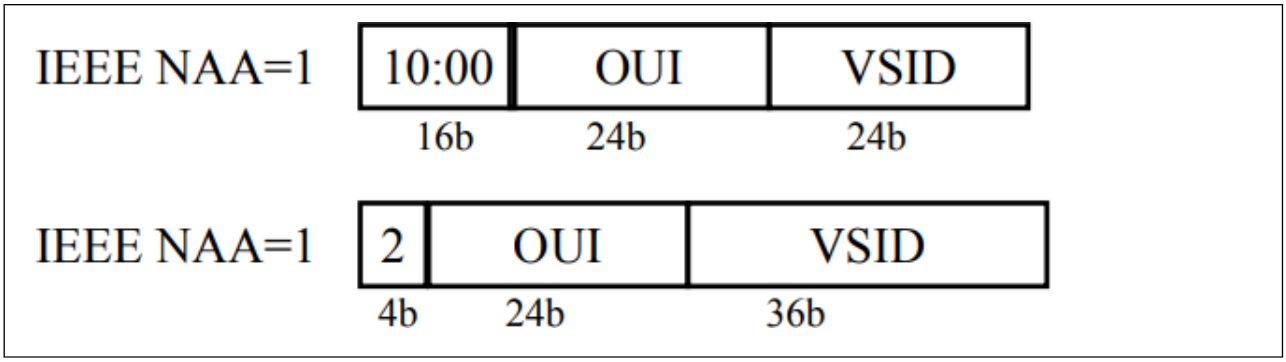


Fig No.32 Topologies of Fiber Channel

VARIOUS NETWORK ADDRESSING AUTHORITIES FIBER CHANNEL CABLES :

Fibre Channel hardware interconnects storage devices with servers and forms the Fibre Channel fabric. The fabric consists of the physical layer, interconnect devices, and translation devices. The physical layer consists of copper and fiber-optic cables that carry Fibre Channel signals between transceiver pairs. Interconnect devices, such as hubs, switches, and directors, route Fibre Channel frames at gigabit rates. Translation devices, such as Host Bus Adapters (HBA), routers, adapters, gateways, and bridges, are the intermediaries between Fibre Channel protocols and upper layer protocols such as SCSI, FCP, FICON, Ethernet, ATM, and SONET. Storage devices at one end of the fabric store trillions of bits of data while servers on the other end distribute the data to hungry users. Fibre Channel hardware stores and distributes data across the work group and the enterprise.

FIBER CHANNEL HARDWARE DEVICES :

Commonly used Devices are as follows:

1. Host Bus Adapters (HBA): Network interface card.

2. Gigabit Interface Converter (GBIC): Single mode fiber for long-distance. Multimode fiber for short distance. HBA ports are empty. Plug in GBIC.

3. Hubs: Physical layer Device. Like a active patch panel. Multiple hosts or storage devices. Only one host can talk to one device at a time using an arbitrated loop (FC-AL) protocol.

4. Switches: A link layer device. Forwards FC frames according to destination address.

5. Routers and Gateways: Connect FC to other types of storage (SCSI)

Fig No.33 Hardware Devices

FIBRE CHANNEL DEVICES

The Fibre Channel Protocol is a **FIBRE CHANNEL PROTOCOL LAYERS** transport protocol that consists of five layers as shown below.

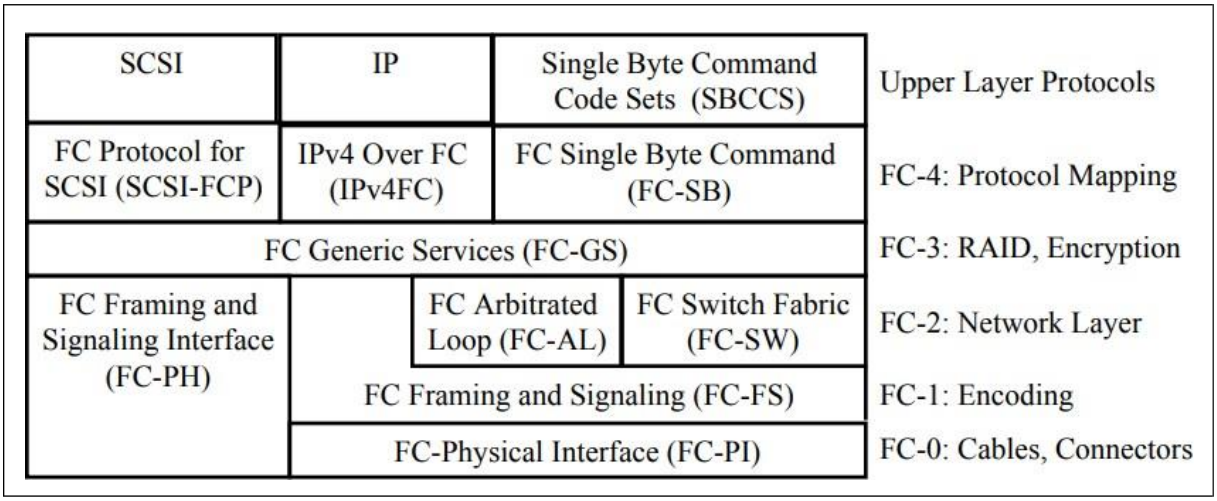


Fig no.34 Fibre channel protocol layers

FIBRE CHANNEL PROTOCOL LAYERS

Fibre Channel Protocol is split into five layers. It does not follow the OSI model, although it is quite similar.

FC-0 – Describes the physical media + the speed in which that device can communicate (1,2, 4, 8, 10, 16 Gbps). When it comes to cabling, Fiber is available as a single mode or multi-mode cable. single mode cable, which is 9 micron thick can be much longer than multi-mode cable, which is 50 micron

FC-1 – Contains specifications for encoding, ordered set, and link control communications. Encoding is a process where parallel electrical singal is encoded into serial optical signal. Speed 1,2,4 and 8 Gbps uses 8bit/10bit encoding. For every 8 bits (or 64bits) that are transmitted there are extra 2 bits that are sent in order to detect link control communication issues etc.

FC-2 – Create FC Frame and flow control. We need flow control to check how much data we can send at a given time, so the receive side can handle all the request. Flow control is handle by buffer-to-buffer control credit. Fibre Channel Frame has a header and may have a payload. Header contains control and addressing information. Where-as payload contains the information being transported form Upper Level Protocol (FC-4).Header has elements such as Destination_ID, Source_ID. Payload is max of 2112 Bytes, but the frame doesn't have to fill-up the payload with zero's if it is empty. The minimum Fibre Channel frame is 64 Bytes.

FC-3 – Largely unused. Common Fabric Services; Defines advanced features, such as stripping and hung group

FC-4 – Provides mapping of Fibre Channel capabilities to Upper Layer Protocols. The most popular will be SCSI. FCP – Fibre Channel Protocol is used to denote the SCSI over Fibre Channel.

The FC protocol layers are generally split into three groups:

-FC-0 and FC-1 are the physical layers.

- FC-2 is the protocol layer, similar to OSI Layer 3.

- FC-3 and FC-4 are the services layers.

The FCoE-FC gateway operates the physical layers and the protocol layer, and provides FIPand service redirection at the services layer. When the switch functions as an FCoE-FC gateway, the switch aggregates FCoE traffic and performs the encapsulation and de-encapsulation of native FC frames in Ethernet as it transports the frames between FCoE devices in the Ethernet network and the FC switch. In effect, the switch translates Ethernet to FC and FC to Ethernet.

- New extensions are named by adding a number, e.g., FC-SW-3 extends FC-SW-2, which extended FC-SW.

- Fibre Channel Shortest Path (FSPF) protocol is used to find routes through the fabric. It is a link-state protocol.

- Vendor specific equal cost path multiplexing

FIBRE CHANNEL FLOW CONTROL

Transmitter sends frames only when allowed by the receiver. It follows the Credit-based flow control. To optimal performance, the Credit value must be greater or equal to Round-trip path delay. supports both Hop-by-Hop and End-to-End networks.

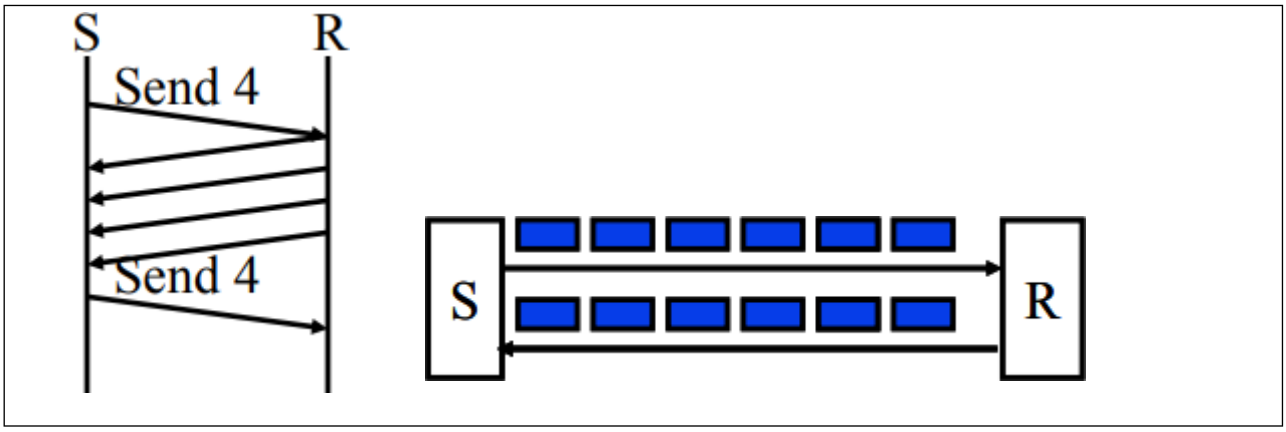


Fig No.35 Flow Control

FIBRE CHANNEL FLOW CONTROL

End to End indicates a communication happening between two applications. (Eg.Skype). It doesn't care what's in the middle, it just consider that the two ends are taking with one another. It generally is a Layer 4 (or higher) communication

Hop by Hop indicates the flow the communication follows. Data pass through multiple devices with an IP address, and each is genetically named "hop". Hop by Hop indicates analyzing the data flow at layer 3, checking all devices in the path

FIBRE CHANNEL CLASSES OF SERVICE:

Class 1: Connection-oriented dedicated physical links. Frame order guaranteed. Delivery confirmation. End-to-end flow control.

Class 2: Connectionless. Due to Multiple paths the order is not guaranteed. Hop-by-hop and end-to-end flow control.

Class 3: There is no dedicated connection in class 3 and the received frames are not acknowledged. Class 3 is similar to Class 2. The only exception is that it only uses buffer-to- buffer credit flow control and does not use end-to-end credit flow control. Class 3 is well suited for SCSI and a commonly used class of service in Fibre Channel networks

Class 4: Class 4 is a connection-oriented service like class 1. The main difference is that class 4 allocates only a fraction of the available bandwidth of path through the fabric that connects two N Ports. In class 4 Virtual circuits (VCs) are established between two N Ports with guaranteed quality of service (QoS), including bandwidth and latency. Class 4 uses only end-to-end credit flow control.

Class 5: Class 5 is called isochronous service, and is intended for applications that require

immediate delivery of the data as it arrives, with no buffering. However, it is still undefined, and possibly scrapped altogether. It is not mentioned in any of the FC-PH documents.

Class 6: Class 6 is a variant of class 1, and is known as a multicast class of service. It provides dedicated connections for a reliable multicast. When a connection is established, it is retained and guaranteed by the fabric until the initiator ends the connection. Class 6 was designed for applications like audio and video that require multicast functionality. It is included in the FC-PH-3 standard.

Class F: Packet-switched delivery with confirmation. For inter switch communication. Class F is used for switch to switch communication through inter-switch links (ISLs).Class F is similar to class 2. The main difference is that class 2 deals with N Ports that send data frames, while class F is used by E Ports for control and management of the fabric.

ISCSI ARCHITECTURE:

ISCSI is an IP-based standard for connecting data storage devices over a network and moving data by carrying SCSI commands over IP networks. In simple words, iSCSI is an interface for communication between initiator and target. IETF protocol to carry SCSI commands over traditional TCP/IP/Ethernet. It does not requires any dedicated cabling. Uses TCP end-to-end congestion control. Can use the same Ethernet port on the computers to connect to storage devices on different computers. iSNS (Internet Storage Name Service) can be used to locate storage resources.

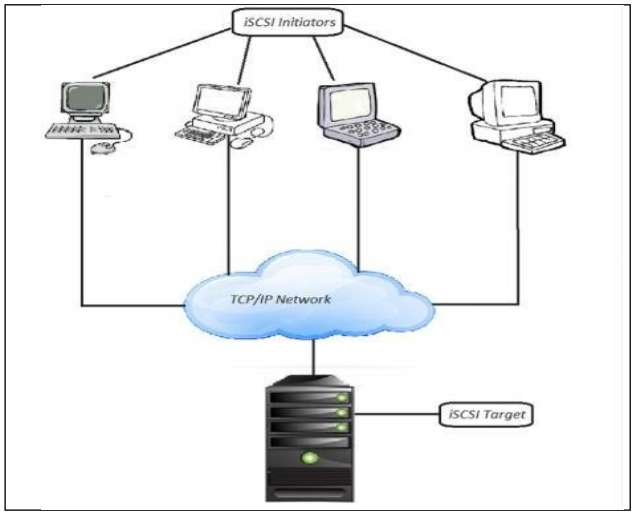


Fig No.36 ISCSI architecture

STRUCTURE OF ISCSI USING TCP IP NETWORK

iFCP (Internet Fiber Channel Protocol)

Interconnect FC devices using TCP/IP.Can connect native IP based storage and FC devices.

SAN frames are converted to IP packets at the source and sent to the destination. Uses TCP Congestion Control (end-to-end).

FCIP (Fibre Channel over IP)

Tunnelling protocol is used for passing FC frames over TCP/IP. SAN packets are encapsulated in IP packets at the source and decapsulated back at the destination. It doesn't allow to directly interface with a FC device. Some FC switches have FCIP ports.

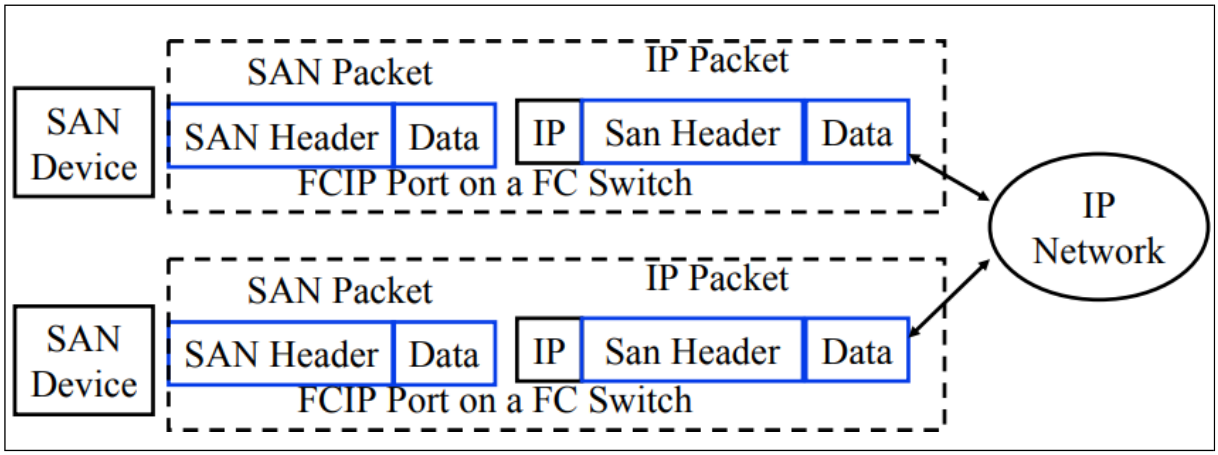


Fig No.37 FCIP

FCIP- FC FRAMES SECURING ISCSI

iSCSI configuration is only as secure as your IP network. By enforcing good security standards while network set up the iSCSI storage is safeguarded. iSCSI SANs support efficient use of the existing Ethernet infrastructure to provide hosts access to storage resources that they can dynamically share. iSCSI SANs are an economical storage solution for environments that rely on a common storage pool to serve many users.

Securing iSCSI Devices

To secure iSCSI devices, requires initiator, that authenticate to the iSCSI device, or target, whenever the host attempts to access data on the target LUN. Authentication ensures that the initiator has the right to access a target. You grant this right when you configure authentication on the iSCSI device. ESXi does not support Secure Remote Protocol (SRP), or public-key authentication methods for iSCSI. You can use Kerberos only with NFS 4.1. ESXi supports both CHAP and Mutual CHAP authentication.

Protecting an iSCSI SAN

The following are some specific suggestions for enforcing good security standards.

1. Protect Transmitted Data

A primary security risk in iSCSI SANs is that an attacker might sniff transmitted storage data. Take additional measures to prevent attackers from easily seeing iSCSI data. Neither the hardware iSCSI adapter nor ESXi iSCSI initiator encrypts the data that they transmit to and from the targets, making the data more vulnerable to sniffing attacks.

Allowing your virtual machines to share standard switches and VLANs with your iSCSI configuration potentially exposes iSCSI traffic to misuse by a virtual machine attacker. To help ensure that intruders cannot listen to iSCSI transmissions, make sure that none of your virtual machines can see the iSCSI storage network.

If you configure iSCSI directly through the ESXi host, you can accomplish this by configuring iSCSI storage through a different standard switch than the one used by your virtual machines.

2. Secure iSCSI Ports

When you run iSCSI devices, ESXi does not open any ports that listen for network connections. This measure reduces the chances that an intruder can break into ESXi through spare ports and gain control over the host. Therefore, running iSCSI does not present any additional security risks at the ESXi end of the connection.

Any iSCSI target device that you run must have one or more open TCP ports to listen for iSCSI connections. If any security vulnerabilities exist in the iSCSI device software, your data can be at risk through no fault of ESXi. To lower this risk, install all security patches that your storage equipment manufacturer provides and limit the devices connected to the iSCSI network.

SAN BACKUP AND RECOVERY TECHNIQUES

SANs are particularly helpful in backup and disaster recovery settings. Within a SAN, data can be transferred from one storage device to another without interacting with a server. This speeds up the backup process and eliminates the need to use server CPU cycles for backup. Also, many SANs utilize Fibre Channel technology or other networking protocols that allow the networks to span longer distances geographically. That makes it more feasible for companies to keep their backup data in remote locations.

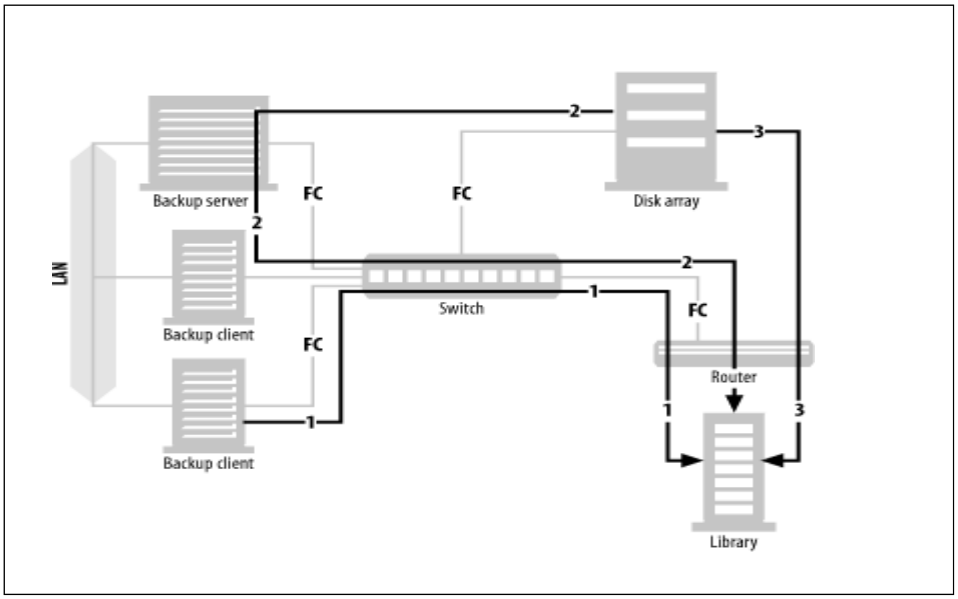


Fig No.38 Backup and Recovery Techniques

SAN ELEMENTS IN ONE REPRESENTATION

A SAN typically consists of multiple servers, online storage (disk), and offline storage (tape or optical), all of which are connected to a Fibre Channel switch or hub—usually a switch. Once the three servers in above diagram are connected to the SAN, each server in the SAN can be granted full read/write access to any disk or tape drive within the SAN. This allows for LAN- free, client-free, and server-free backups, each represented by a different-numbered arrow in above diagram.

LAN-free backups

LAN-free backups occur when several servers share a single tape library. Each server connected to the SAN can back up to tape drives it believes are locally attached. The data is transferred via the SAN using the SCSI-3 protocol, and thus doesn't use the LAN. All that is needed is software that will act as a "traffic cop." LAN-free backups are represented above by arrow number 1, which shows a data path starting at the backup client, travelling through the SAN switch and router, finally arriving at the shared tape library.

Client-free backups

Although an individual computer is often called a server, it's referred to by the backup system as a client . If a client has its disk storage on the SAN, and that storage can create a mirror that can be split off and made visible to the backup server, that client's data can be backed up via the backup server; the data never travels via the backup client. Thus, this is called client- free backup. Client-free backups are represented by arrow number 2, which shows a data path starting at the disk array, traveling through the backup server, followed by the SAN switch and router, finally arriving at the shared tape library.

The backup path is similar to LAN-free backups, except that the backup server isn't backing up its own data. It's backing up data from another client whose disk drives happen to reside on the SAN. Since the data path doesn't include the client that is using the data, this is referred to as client-free backups.

Server-free backups

If the SAN to which the disk storage is connected supports a SCSI feature called extended copy , the data can be sent directly from disk to tape, without going through a server. There are also other, more proprietary, methods for doing this that don't involve the extended copy command. This is the newest area of backup and recovery functionality being added to SANs. Server-free backups are represented by arrow number 3, which shows a data path starting at the disk array, traveling through the SAN switch and router, and arriving at the shared tape library. You will notice that the data path doesn't include a server of any kind. This is why it's called server-free backups.

RAID

RAID is a technology that is used to increase the performance and/or reliability of data storage. The abbreviation stands for either Redundant Array of Inexpensive Disks or Redundant Array of Independent Drives. A RAID system consists of two or more drives working in parallel. These can be hard discs, but there is a trend to also use the technology for SSD (Solid State Drives). There are different RAID levels, each optimized for a specific situation.

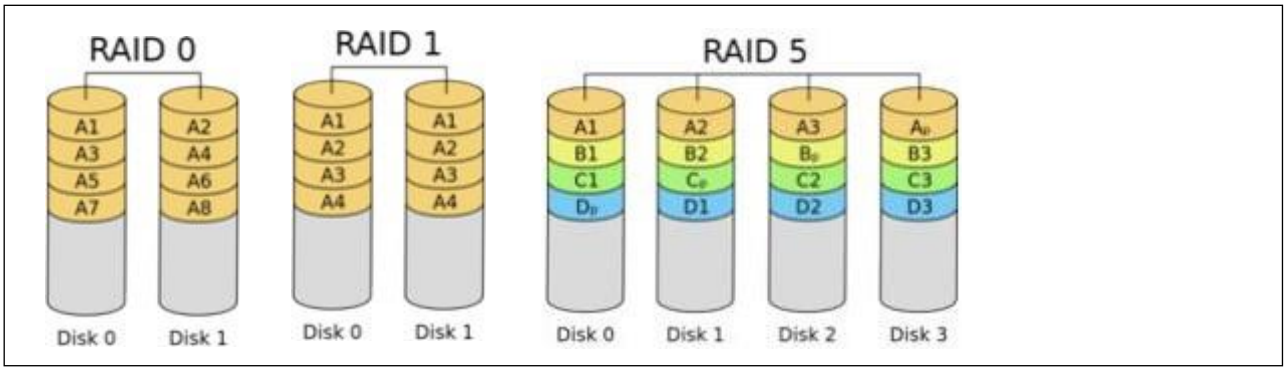


Fig No.39 RAID

Various RAID levels available are

RAID 0 – block-level striping without parity. Zero redundancy. Higher performance and capacity. In a RAID 0 system data are split up into blocks that get written across all the drives in the array. By using multiple disks (at least 2) at the same time, this offers superior I/O performance. This performance can be enhanced further by using multiple controllers, ideally one controller per disk.

RAID 1 – Mirroring without parity. Higher read performance. Two or more mirrors. Data are stored twice by writing them to both the data drive (or set of data drives) and a mirror drive (or set of drives). If a drive fails, the controller uses either the data drive or the mirror drive for data

recovery and continues operation. You need at least 2 drives for a RAID 1 array.

RAID 2 - Bit-level striping with dedicated Hamming code parity. Each sequential bit is on a different drive. Not used in practice.

RAID 3 - Byte-level striping with dedicated Hamming code parity. Not commonly used.

RAID 4 - Block-level striping with dedicated parity. Allows I/O requests to be performed in parallel.

RAID 5 – Block-level striping with distributed parity. Masks failure of 1 drive. RAID 5 is the most common secure RAID level. It requires at least 3 drives but can work with up to 16. Data blocks are striped across the drives and on one drive a parity checksum of all the block data is written.

RAID 6 – Block-level striping with double distributed parity. Masks up to two failed drives. Better for large drives that take long time to recover. RAID 6 is like RAID 5, but the parity data are written to two drives. That means it requires at least 4 drives and can withstand 2 drives dying simultaneously. The chances that two drives break down at exactly the same moment are of course very small. However, if a drive in a RAID 5 systems dies and is replaced by a new drive, it takes hours or even more than a day to rebuild the swapped drive. If another drive dies during that time, you still lose all of your data. With RAID 6, the RAID array will even survive that second failure.

RAID 10 – combining mirroring and striping

Nested RAIDs:

RAID of RAID drives. In RAID 01: Stripe and then mirror = RAID 0+1. Data is striped across primary disks that are mirrored to secondary disks. In RAID 10: Mirror then stripe = RAID 1+0. The order of digits is the order in which the set is built. RAID 0+1 implies Stripping first and then mirroring. Mirrored striped set with distributed parity = RAID 5+3 or RAID 53

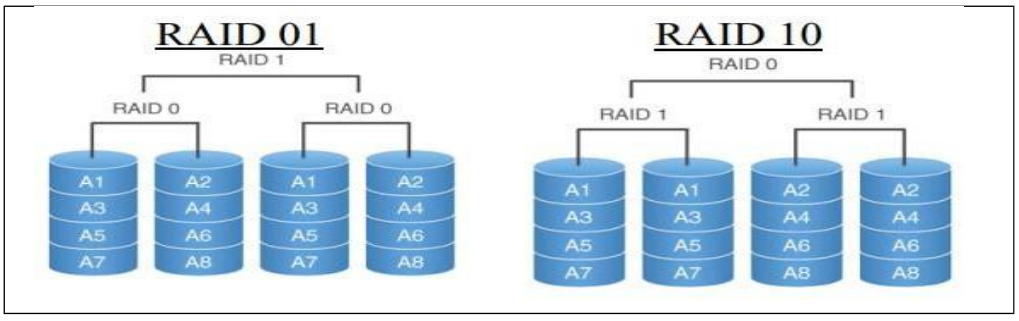


Fig No.40 Nested RAID

Advantages of RAID

1. RAID 1 is useful in systems where complete redundancy of data is necessary and disk

space is readily available.

2. With RAID1 the writes to the memory are much faster.

3. Inclusion of RAID improves the system performance by distributing I/O load equally on several physical servers.

4. RAID supports creation of logical files that are larger than the maximum size supported by the operating system.

Disadvantages of RAID

1. RAID 1 is not suited for systems with large data files and less disk space.
2. Data striping might not allow you to locate single data file on specific physical drive.
3. Some of the application tuning capabilities might be lost due to data striping.

4.Data recovery becomes time consuming, as all the disks that are a part of the logical RAID device must be involved (for read and write operating) in the recovery process.

The classic storage model

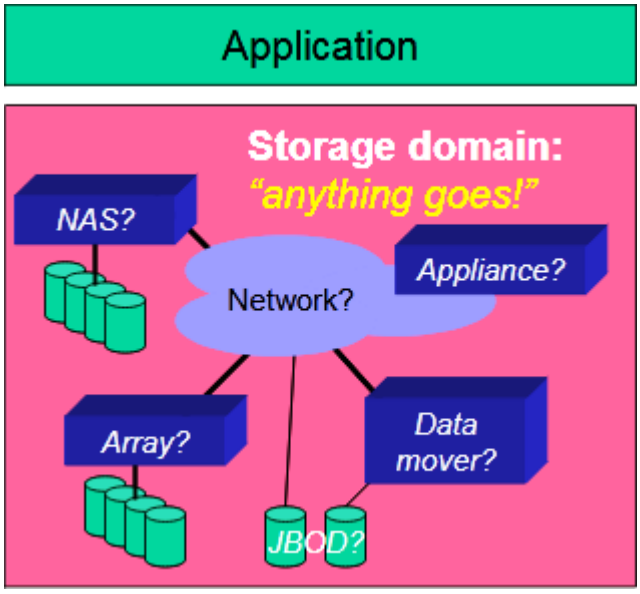


Fig No.41 Storage Model

All too often, the picture shown here represents the current state of conversations about storage networking: vendors, system designers, and customers try to describe what they want – or what they have – using a set of inconsistent, ad hoc, languages. Things are made worse by there being a great many network storage components, with relatively small differences between them. This causes designs that are actually the same to be described in different ways; and different designs to be described sometimes using identical forms of words. This is clearly undesirable, and results

in many problems: it's often not obvious what is being proposed or described; tradeoffs between alternatives are harder to identify than they could – or should – be; and it's harder for everybody to make high quality decisions. These confusions are not accidental: the wide variety of the range of system architectures that have been developed exhibit a great deal of complexity because they are trying to accommodate a great deal of information, and cover many different elements and functions. Some of those elements are physical – boxes, wires, computers – and it is often the case that architectures are presented by describing the physical components in some detail, coupled with an explanation of what functions they perform. That is, the traditional approach focuses first on the physical partitioning that a particular vendor has selected, rather than on the range of options that may be possible. And because this is "box-centric" rather than "function- centric" it is all too easy to misunderstand precisely what has been included. The SNIA Shared Storage Model is an approach to removing these difficulties. It does so by taking a slightly different approach: it first identifies the functions that can be provided, and then describes a range of different architectural choices for placing those on physical resources. As a result, the SNIA Shared Storage Model makes it easier to compare alternative architectures and designs, it lets architects think about functions independently of implementations, and it makes it simpler to anticipate new implementations or combinations of architectures, designs, and implementations.

The SNIA Shared Storage Model

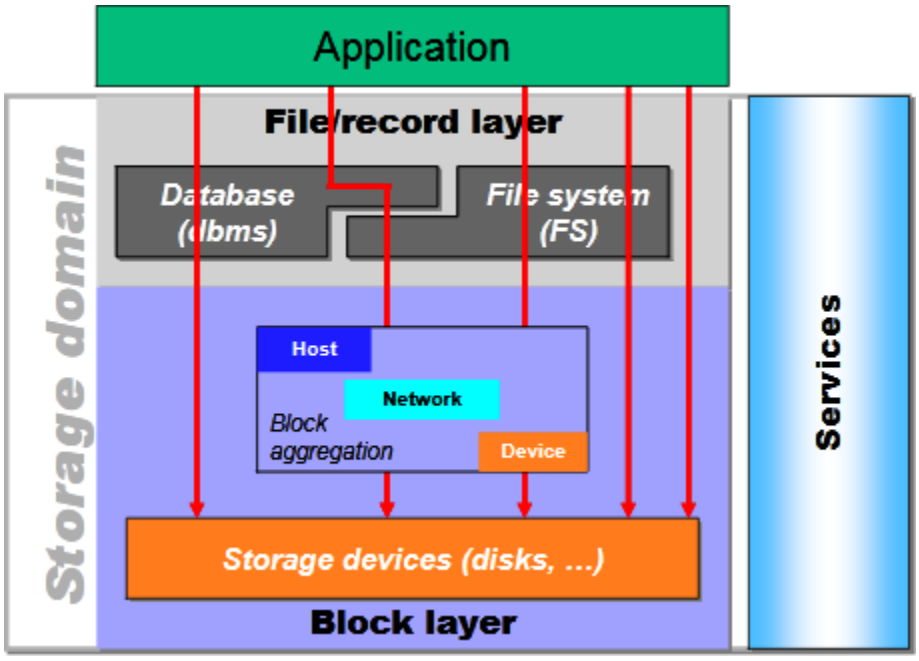


Fig No.42 SNIA Shared Storage Model

This is the highest-level picture of the SNIA Shared Storage Model. It has three main components within its scope:

1. The file/record layer, which includes databases and file systems.

2. The block layer, which includes both low-level storage devices and block-based aggregation.

3. A services subsystem, which provides functions such as the management of the other components. Note that applications lie outside the scope of the model – they are viewed as "clients" of the storage domain, in the broadest sense.

Storage system components

The SNIA Shared Storage Model supports the following kinds of components:

•Interconnection network– the network infrastructure that connects the elements of the shared storage environment. This network may be a network that is primarily used for storage access, or one that is also shared with other uses. The important requirement is that it provides an appropriately rich, high-performance, scalable connectivity upon which a shared storage environment can be based. The physical-layer network technologies that are used (or have been used) for this function include Fibre Channel, Fast- and Gigabit-Ethernet, Myrinet, the VAX CI network, and ServerNet. Network protocols that are used at higher layers of the protocol stack also cover a wide range, including SCSI FCP, TCP/IP, VI, CIFS, and NFS.

Redundancy in the storage network allows communication to continue despite the failure of various components; different forms of redundancy protect against different sorts of failures. Redundant connections within an interconnect may enable it to continue to provide service by directing traffic around a failed component. Redundant connections to hosts and/or storage enable the use of multi-path I/O to tolerate interface and connection failures; multi-path I/O implementations may also provide load balancing among alternate paths to storage. An important topology for multi-path I/O uses two completely separate networks to ensure that any failure in one network cannot directly affect the other.

•Host computer –a computer system that has some or all of its storage needs supplied by the shared storage environment. In the past, such hosts were often viewed as external to the shared storage environment, but we take the opposite view, and will show examples of function mappings that place key components in such hosts. A host typically attaches to a storage network with a host-bus adapter (HBA) or network interface card (NIC). These are typically supported by associated drivers and related software; both hardware and software may be considered part of the shared storage environment. The hosts attached to a shared storage environment may be largely unaware of each other, or they may explicitly cooperate in order to exploit shared storage environment resources. Most commonly this occurs across subsets of the hosts ("clusters"). One of the advantages of separating hosts from their storage devices in a shared storage world is that the hosts may be of arbitrary and differing hardware architecture and run different versions and types of operating system software.

•Physical storage resource – a non-host element that is part of the shared storage environment, and attached to the storage network. Examples include disk drives, disk arrays, storage controllers, array controllers, tape drives and tape libraries, and a wide range of storage appliances. (Hosts are not physical storage resources.) Physical storage resources often have a high degree of redundancy, including multiple network connections, replicated functions, and data redundancy via RAID and other techniques – all to provide a highly available service.

•Storage device – a special kind of physical-storage resource that persistently retains data.

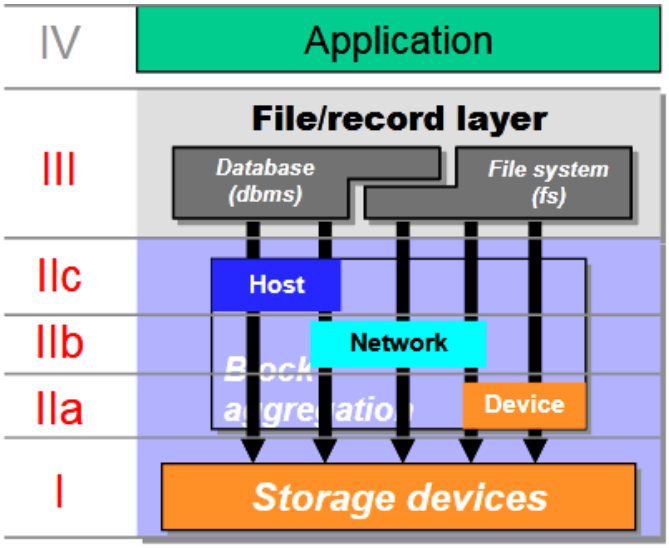
• Logical storage resource – a service or abstraction made available to the shared storage

environment by physical storage resources, storage management applications, or combination thereof. Examples include volumes, files, and data movers.

Storage management – functions that observe, control, report, or implement logical storage resources. Typically these functions are implemented by software that executes in a physical storage resource or host.

The layering scheme of the SNIA Shared Storage Model

The SNIA Shared Storage Model is a layered one. The figure shows a picture of the stack with a numbering scheme for the layers. Roman numerals are used to avoid confusion with the ISO and IETF networking stack numbers



The layers are as follows:

- IV. Application
- III. File/record layer
  - IIIb. Database
  - IIIa. File system
- II. Block aggregation layer, with three function-placements:
  - IIc. Host
  - IIb. Network
  - IIa. Device
- I. Storage devices

We will now describe each of these layers, from the topmost layer downwards.

Fig No.43 layering scheme of the SNIA

The file/record layer

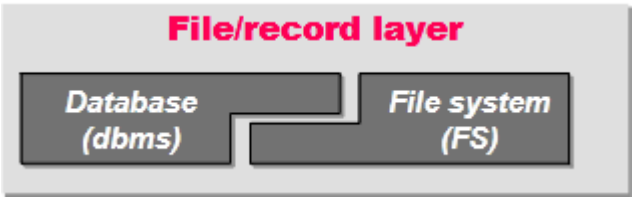


Fig No.44 File /Record layer

bytes → files → volumes Because a byte vector can be used to emulate a block vector, the volumes that a database is mapped to can sometimes be files. This is most often done for small database systems where the performance penalties of the two levels of mapping it entails are outweighed by the simpler management that results from exploiting the naming and access control mechanisms offered by the file system. Secondary functionality provided by this layer may include content indexing, aggressive prefetching and write-behind techniques to improve performance, hierarchy management, and providing coherency across multiple copies in distributed systems. In the future, we expect to see new implementations at this layer, such as file systems explicitly designed for

replaying isochronous streams against multimedia objects (e.g., videos). Indeed, an http web cache might be considered a new kind of distributed file system.
Where can it be done?

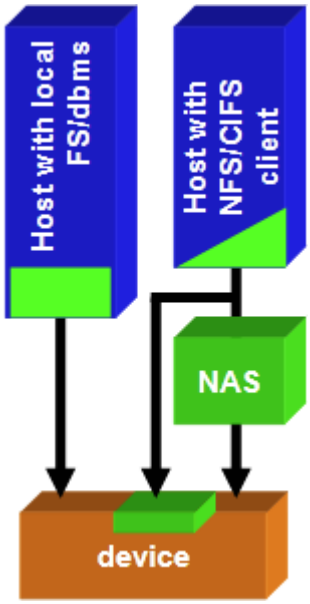


Fig No.45 File /Record layer

The functions provided by the file/record layer can be implemented in several different places:
•Solely in the host: these are the traditional host-based file systems and databases (the left hand column in the figure shown here). In such systems, the implementation of the file system or database resides completely in the host, and the interface to the storage device is at the block-vector level.
• In both client and server: these are the standard "network" file systems such as NFS, CIFS, etc. (The right-hand column of the figure.) Such implementations split their functions between the client (host) and the server system. (Note that this split happens inside the file system layer in the SNIA Shared Storage Model.) The client side always resides in a host computer. The server side, however, can reside in a: ofile server (sometimes: database server) – typically a host computer with local attached block storage devices that may be dedicated to this function. oNAS head – a dedicated-function computer acting as a file server and relying on external block storage devices connected through a storage network. ostorage device – such as a disk array or "smart disk".

The block layer

The block layer provides low-level storage to higher layers, typically with an access interface that supports one or more linear vectors of fixed-size blocks. In SCSI, these logical address spaces are called logical units (LUs); a single SCSI storage device may support several such logical units.

Ultimately, data is stored on "native" storage devices such as disk drives, solid-state disks, and tape drives. These devices can be used directly, or the storage they provide can be aggregated into

one or more block vectors to increase or decrease their size, or provide redundancy. This aggregation can occur in many places – more on this below. Secondary responsibilities of the block layer include a simple form of naming, such as SCSI Logical Unit Names (LUNs), caching (and, in particular, non-volatile caches for write-behind data), and (increasingly) simple access control.

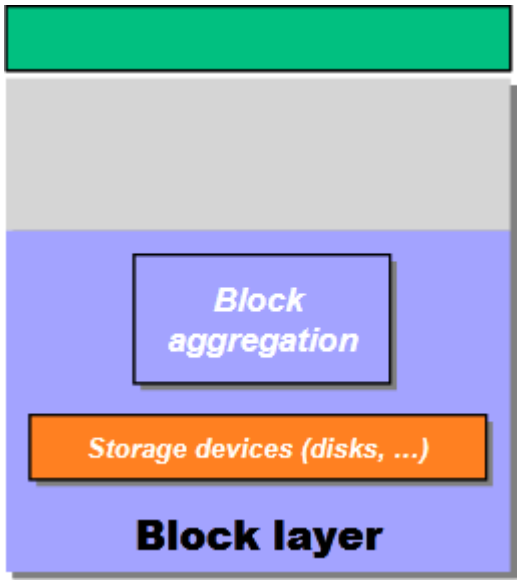


Fig No.46 Block layer

- Block aggregation

The block layer provides low-level storage to higher layers, typically with an access interface that supports one or more linear vectors of fixed-size blocks. In SCSI, these logical address spaces are called logical units (LUs); a single SCSI storage device may support several such logical units.

Ultimately, data is stored on "native" storage devices such as disk drives, solid-state disks, and tape drives. These devices can be used directly, or the storage they provide can be aggregated into one or more block vectors to increase or decrease their size, or provide redundancy. This aggregation can occur in many places – more on this below. Secondary responsibilities of the block layer include a simple form of naming, such as SCSI Logical Unit Names (LUNs), caching (and, in particular, non-volatile caches for write-behind data), and (increasingly) simple access control Block aggregation comprises a powerful set of techniques that are used to serve many purposes. These include:

•Space management: constructing a large block vector by assembling several smaller ones, or packing many small block vectors into one large one, or both. (This "slicing and dicing" has historically been one of the most important functions of host-based logical volume managers.) •Striping: apportioning load across several lower-level block vectors and the systems that provide them. The typical reason for doing this is to increase throughput by increasing the amount of parallelism available; a valuable secondary benefit may be the reduction in average latency that

can result as a side effect.

•Providingredundancy for increasing availability in the face of storage device failures. This can be full redundancy (e.g., local & remote mirroring, RAID-1, -10...); or partial redundancy (RAID-3, -4, -5, ...). Additional features like point-in-time copy (various versions of this are sometimes called “snapshot”) can be provided, which can be used to increase the effective redundancy level of a system, and to help recover from other kinds of failures. In practice, several of these functions are often combined together. For example, a system that can handle striping is often also capable of performing mirroring (a combination sometimes referred to as RAID-10).

- **Where can it be done?**

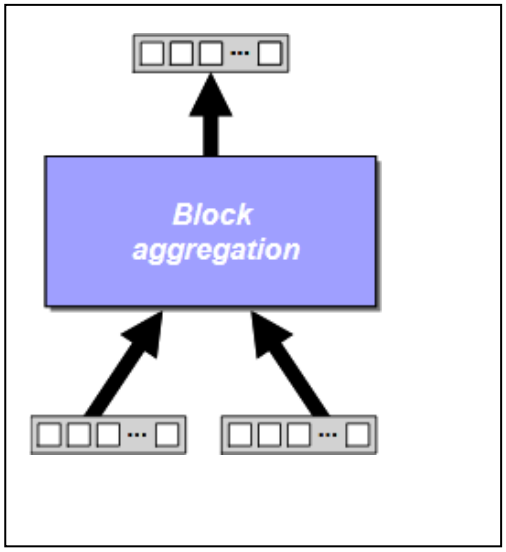


Fig No.47 Block aggrigation

The block aggregation functions can be performed at several of the storage components described in the model. Indeed, it is common to find more than one being used. •Host- side, such as in logical volume managers, device drivers, and host bus adapters (HBAs) • In components of the storage network itself, such as specialized “SN appliances”. In addition, some HBAs are better thought of as part of the storage network.

And, very commonly, in the storage devices themselves: disk array controllers (e.g., RAID) are classic examples of this. Modern disk drive controllers provide some level of this functionality too, such as a logical-to-physical block mapping for supporting sparing.

- **How is it done?**

The figure shown here offers a simple, visual model of how block aggregation operates. It also illustrates the reason that block-based aggregation functions at different components can be composed together. You can think of each kind of aggregation function as a building block that

imports (uses) one or more block vectors from “below”, and exports to its clients one or more block-vectors at its upper interface that are constructed (i.e. “aggregated” or “virtualized”) from those imported ones. The construction can embody any or al of the functions described above. Because the interfaces to both imported and exported block vectors are the same, these building blocks can often be stacked on top of one another: for example, mirroring across two disk arrays could be performed in a host logical volume manager, and RAID 5 redundancy applied within each of the disk arrays. Each layer could in theory also be internally constructed in this same way.

Sample architectures

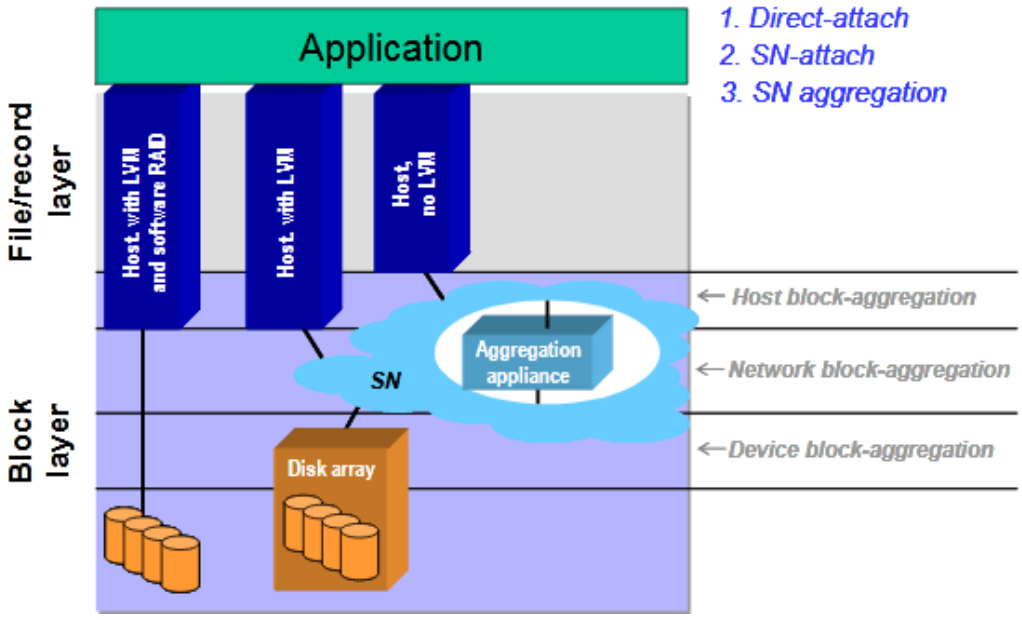


Fig No.48 Block-based storage architecture

This illustration shows the first application of the model to a number of different block-based storage architectures. •direct-attach: the leftmost column illustrates this. A host is connected to some private, non-shared storage devices (e.g., disk drives attached by a local SCSI cable). In the figure, the host is shown running a logical volume manager (LVM) – perhaps capable of providing a software RAID function to provide protection against disk failure.

•storage-network attach: the second and third hosts, plus the storage network and disk array in the second column illustrate this. This scheme introduces a storage network (the pale blue cloud) connecting one or more hosts – perhaps still running LVM software – to a disk array that is providing a further set of block aggregation functions. The disk array resources can now be shared between multiple hosts.

•storage-network aggregation: the final example embeds a block-aggregation function into the storage network in an aggregation appliance that might be providing access control and (say) striping aggregation functions.

Putting it all together – combining the block & file/record layers:

This picture puts together both block-based and file-based storage architectures. •direct-attach: the

leftmost column shows local, private storage directly connected to a single host. •storage- network attach: the second column shows a representative host connected to a (shared) disk array through a storage network (SN). •NAS head: the third column illustrates a dedicated-function "NAS head" (file server controller) interposed between the lower-level, block-based storage network and its clients, which are connected to it through a second network (generically an arbitrary second storage network, but shown here as a LAN, as that is the most common form), and operate using client-server file system protocols. Note that block-aggregation functions can be used to support several NAS heads, as well as regular block-level hosts. •NAS server: this is shown in the right-most (fourth) column and logically consists of a combined NAS head and its own local, private storage. Access paths

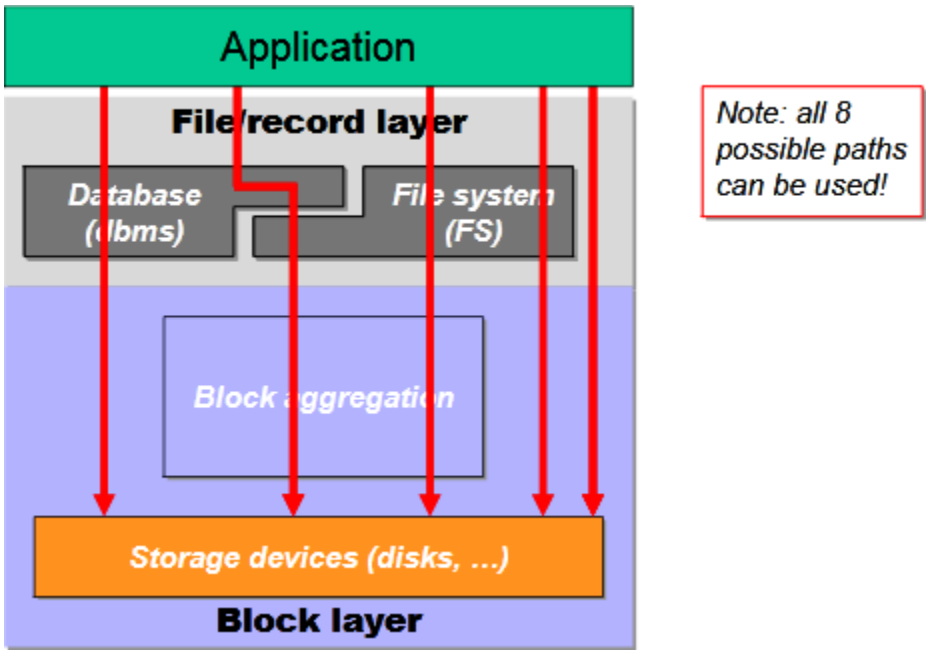


Fig No.49 Block-based and file-based storage architecture

An access path is the list of components (hardware and software) that are traversed by read and write requests to the storage system and their responses. If we restrict ourselves to avoid cycles, there are eight possible paths from the application layer to the lowest storage layer through the elements of the SNIA Shared Storage Model. Five examples of common paths are shown in the figure here, reading from right to left:

- direct to the storage device (e.g., disk drive or disk array)
- via a file system
- via a file system that sits on top of a block aggregation function

- via a database on top of a file system on top of a block aggregation function

•via a database Caching:

Caching is designed to shorten access paths for frequently referenced items, and so improve the performance of the overall storage system. Most elements of a storage system can provide a cache, and so such caches can be performed at the block or file/record layer – or both. Indeed, it is common to see several caches in operation simultaneously. For example: a read cache in a file system, coupled with a write-back cache in a disk array, and a readahead cache in a disk drive.

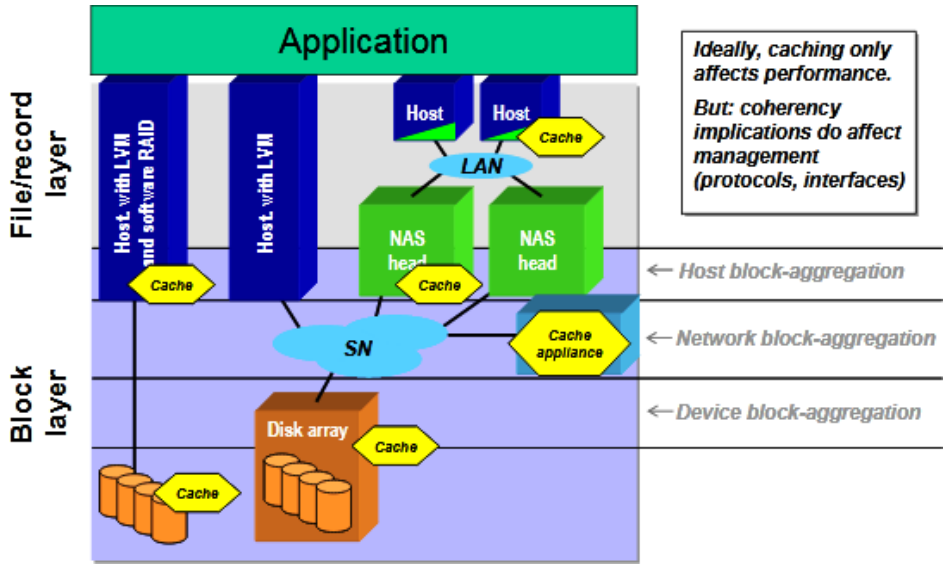


Fig No.50 Caching

The figure here illustrates this: almost any of the components in the system can be augmented with a cache. The figure also introduces a new component: a dedicated caching appliance, added to the storage network solely to provide caching functions. Ideally, all that adding caching does is speed things up, but making this true for shared storage networks requires taking account of a multitude of detailed issues that occur in distributed systems, such as maintaining the coherency of multiple copies of data or metadata (data about data), and tolerating partial failures of the system. In particular, cache management (deciding which cache should – or does – hold what) is significantly more complicated when data may be cached in several places.

Access control:

A shared storage environment bestows the benefit of hosts being able to access their storage devices directly. But the existence of an access path should not be taken as equivalent to permission to exercise it. Access control is the set of techniques that enforce the decisions that encapsulate these permissions. There are many different kinds of unwanted accesses possible, and the protection used against them has perforce to trade off the degree of protection against efficiency, and the complexity of enforcement.

The basic goals are to provide:

•authentication (“proof that I am who I say I am”),

•authorization (“proof that I am allowed to do this”), and

•privacy (“proof that I am allowed to see the contents”).

Historically, storage systems have provided little or no support for any of these, other than via simple physical security – locking up the storage devices and the hosts that access them. This is likely to change significantly in the storage network world because the number of different threat types is so much large . ltimately, all approaches to access control rely on some form of secure channel being established between the provider of data (or operations on that data) and its destination. The least secure, but also easiest to implement, solution imposes simple, coarse-grained accessor checks (of the form “is this host permitted to send requests to this storage device?”); at the other extreme lies cryptographic protection mechanisms that are resistant to a wide variety of impersonation, monitoring, and replay attacks, and capable even of securely storing data on storage devices that cannot be trusted not to divulge (or lose) their contents. Preventing unwanted accesses can be performed in several places:

•At the host: this offers convenience of implementation, easy scalability, low execution cost, and potentially fine-grained control, but it must be pervasively deployed, and the enforcement mechanism should be resistant to tampering. With support from their host operating systems, file systems and databases commonly enforce access controls on data, and similar kinds of solutions are appearing at the block vector layer. Networking stacks in the host can also use encryption to provide secure channels across various forms of network (e.g., IPsec). The performance of software versions of these schemes means today that they are best suited to use on relatively low–speed network links (such as a wide area network), but this is likely to change with the deployment of hardware accelerators.

•In the storage network: today, this is largely restricted to relatively low-level approaches that offer the illusion of private, dedicated sub-networks that permit a set of host and storage device ports access only to one another, hiding any other ports. (These are usually called zones in Fibre Channel, or VLANs in the Ethernet world.) Because they operate on whole ports, such solutions are quite coarse-grained. But, by analogy with the migration of functions such as load balancing into traditional IP networking components, it is reasonable to expect finer-grain controls appearing in storage network switches, such as the ability to enforce such virtual networks on per-logical-unit (LU) boundaries.

•At the storage devices: ultimately, storage devices will probably have to accept as much responsibility for enforcing access controls as the hosts do: they are, after all, the primary shared resource that storage networking is trying to make available. This has long been true of servers at the file/record layer such as file servers and NAS heads; and now solutions are appearing that perform a simple form of "LU masking" at the block layer, where only certain host (or host ports) are allowed access to a particular LU. As storage networks span ever-greater numbers of devices, and encompass greater heterogeneity of host types, host software, and distances, the importance of this issue will greatly increase.

The services subsystem:

A very important part of the model is the set of services that lie "off to one side" of the critical data-access paths. The list provided in the graphic here is not meant to be exhaustive, but to give a flavor of the kinds of things that are handled by this services subsystem. Many of the services are "management" tasks, and need to be tied into the larger system-wide service management tool processes and tools. Although such services are vital for successful implementations, they are not further discussed here: this version of the SNIA Shared Storage Model deliberately focuses on the data-access portion, in order to allow us to communicate the model for discussion and use in a timely manner. Refining the definitions, terminology, specifications, and interfaces associated with the services subsystem represents a major opportunity for the SNIA community. This issue will be explored in detail in future SNIA Technical Council reports.

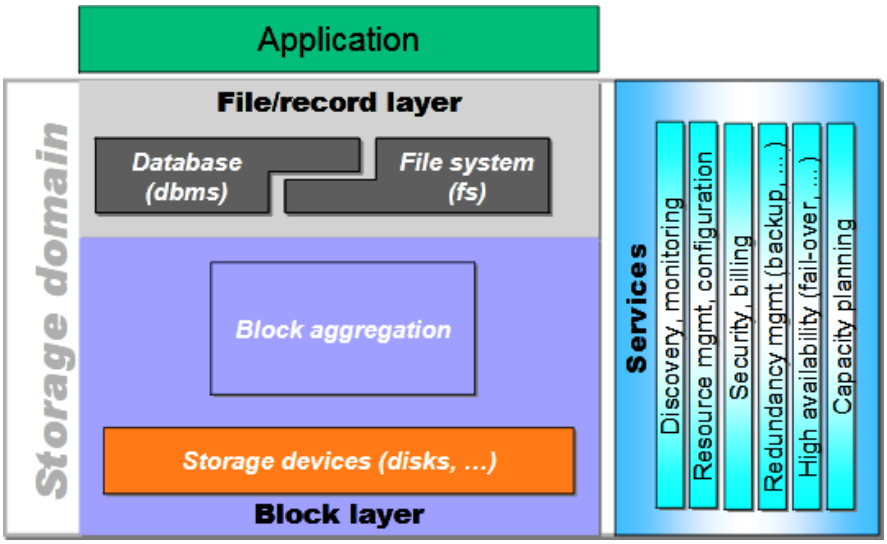


Fig No.51 storage domain

Storage based architecture:

Storage network-attached block storage with metadata server ("asymmetric block service")

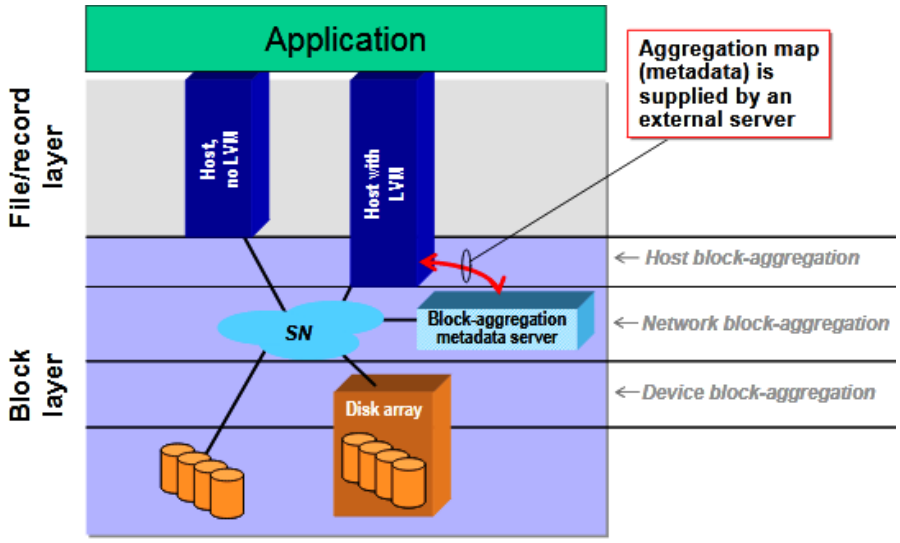


Fig No.52 asymmetric block service

Storage network-attached block storage with metadata server is characterized by

• having multiple hosts and devices attached to a shared storage interconnect,

• employing a block interface protocol over that interconnect,

• having the hosts communicates directly to the storage devices, while

•employing a metadata server to provide layout information ("block metadata") to the hosts on

the current layout of block data on those storage devices. By comparison to the "SAN appliance" architecture, this does not impose additional physical resources in the data access path, but data placement changes require coherent updates of any cached copies of the metadata (layout information) held at the hosts.

Network based Architecture:

Multi-site block storage

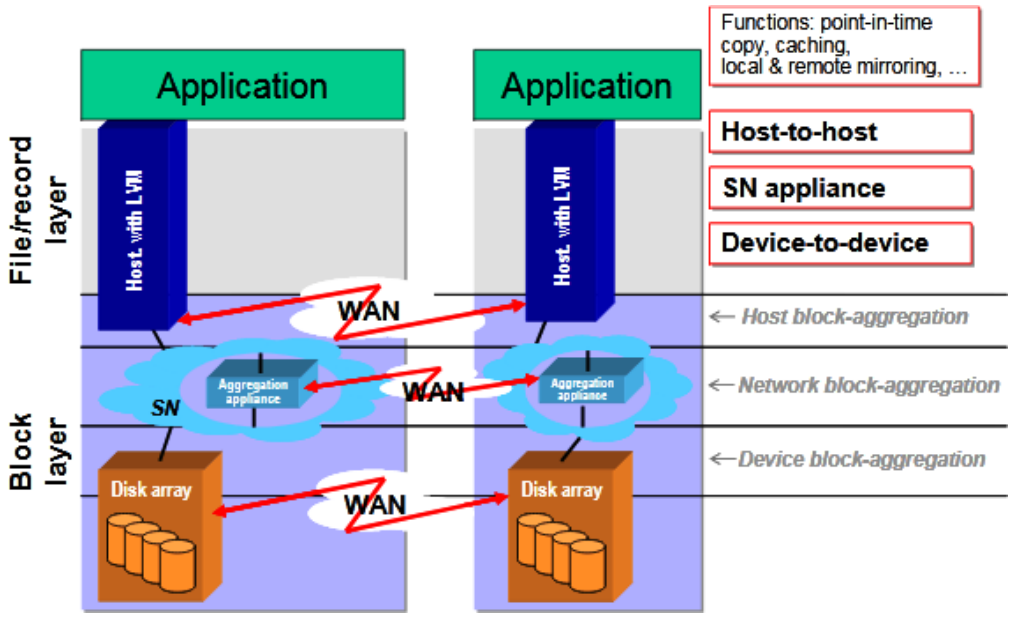


Fig No.53 Multi-site block storage

Multi-site block storage is characterized by the use of peer-to-peer protocols between like components of two or more systems at different sites to maintain data replicas at each site. This addresses the increasing need for geographic separation and appropriate decoupling between two or more data sites. In turn, this can be used to enhance data availability in the presence of site disasters, while – with careful caching and update protocols – retaining the performance advantages of having access to a local copy of data. (This is particularly important in the presence of the larger propagation delays and lower bandwidths of long-haul networks.) The peer-to-peer protocols can be implemented at several different levels, such as between pairs of logical volume managers, SAN appliances (e.g., remote mirroring boxes), and between storage devices themselves, such as disk arrays. The type of network used between the sites is frequently different than the network used within each site, so gateways or protocol conversion boxes may need to be employed to achieve the desired connectivity.

File server:

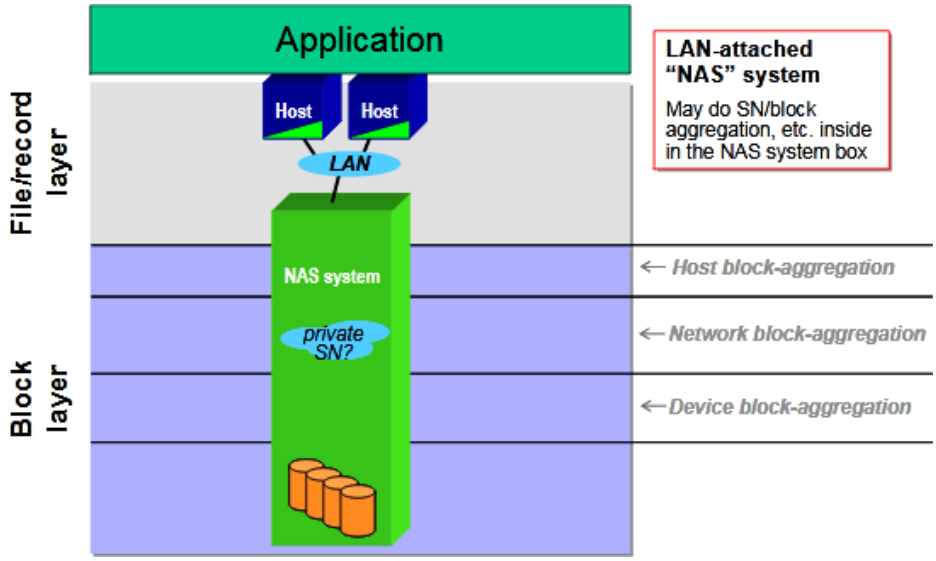


Fig No.54 File servers

File servers ("NAS systems") are characterized by

• bundling storage devices and a file/record subsystem controller into one package,

• employing a client:server file/record protocol to access the data,

•and using a network that is typically not specialized for, or dedicated to, storage traffic, such as a LAN Of the approaches to shared, network storage, this is probably the commonest, most mature, easiest to deploy, and most capable today of supporting heterogeneous hosts. The price is that the file server can sometimes be a performance, capacity or availability bottleneck. Some database servers exist with a similar architecture.

File server controller ("NAS head"):

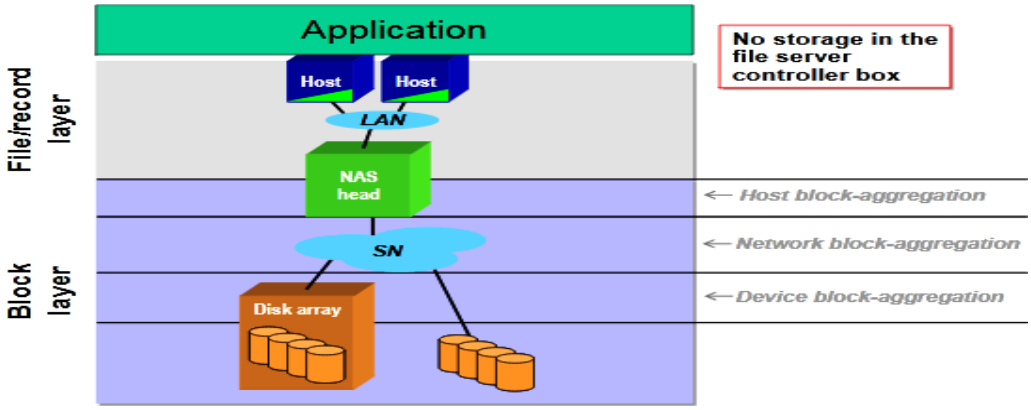


Fig No.55 File server controllers

File server controllers are characterized by:

• decoupling storage devices from the file/record subsystem controller that provides access to them,

•employing a client:server file/record protocol to access the file/record subsystem from the client hosts, and using a network that is typically not specialized for, or dedicated to, storage traffic, such as a LAN for the host to file/record subsystem traffic,

•having the file/record subsystem controller, as well as multiple hosts and devices, attached to a shared storage interconnect that employs a block interface protocol. This variation of the classic file server model has several potential benefits:

•the block storage devices can be used directly (as shared block storage resources) by both the file/record service and the hosts;

• the hosts can be bound to both block and file services from common resources at the same time;

•easier independent scalability of file/record subsystem performance and block-storage performance and capacity. The cost is largely that of increased complexity of managing the larger numbers of components that are exposed compared to the integrated file server approach.

Fault tolerance to SAN:

High Availability Using Fault Tolerance in the SAN:

This session will appeal to those seeking a fundamental understanding of the role fault tolerance plays in High Availability (HA) configurations. Modern SANs have developed numerous methods using hardware and software fault tolerance to assure high availability of storage to customers. The session will explore basic concepts of HA, move through a sample configuration from end-to-end, and discuss some of the challenges faced in testing HA configurations.

High Availability Using Fault Tolerance in the SAN

- Promises a certain amount of uptime
- Promises access to critical functions of the system
- Allows system to handle faults or failures in a system
- Involves redundant components
- Allows component upgrades or maintenance without impacting availability

What This Is Not

- A guarantee of 100% availability
- An inexpensive solutionA DR solution

The Language of HA

Uptime:

Measure of the time a computer system has been “up” and running (does not imply availability)

Availability:

The proportion of time a system is production capable

High Availability:

System design protocol and associated implementation that ensures a certain absolute degree of operational continuity during a given measurement period

Fault Tolerance:

The ability to continue properly when a hardware or software fault or failure occurs. Designed for reliability by building multiples of critical components like controllers, adapters, memory and disk drives.

Redundancy:

The duplication of components to ensure that should a primary resource fail, a secondary resources can take over its function

And Now For The Parts

Storage Controller/Controller

The control logic in a storage subsystem that performs, among other things, command transformation and routing, I/O prioritization, error recovery and performance optimization

Fabric

Interconnection method that allows multiple hosts and/or storage devices connected with a multi-port hub, simultaneous and concurrent data transfers

Adapter

-Circuit board that provides I/O processing and physical connectivity between a server and storage deviceMultipathingThe use of redundant storage networking components (adapters, cables, switches) responsible for the transfer of data between the server and the storage

Availability	Downtime/Yr	Downtime/Mo	Downtime/Wk
90%	36.5 days	72 hours	16.8 hours
95%	18.25 days	36 hours	8.4 hours
98%	7.30 days	14.4 hours	3.36 hours
99%	3.65 days	7.20 hours	1.68 hours
99.5%	1.83 days	3.60 hours	50.4 minutes
99.8%	17.52 hours	86.23 minutes	20.16 minutes
99.9%	8.76 hours	43.2 minutes	10.1 minutes
99.95%	4.38 hours	21.56 minutes	5.04 minutes
99.99%	52.6 minutes	4.32 minutes	1.01 minutes
99.999%	5.26 minutes	25.9 seconds	6.05 seconds
99.9999%	31.5 seconds	2.59 seconds	0.605 seconds

Table.No.2 Redundancy Based Fault Tolerance

Redundancy Based Fault Tolerance

Designed to ensure it takes two independent local faults, in a short time period, to cause end-to-end failure

Even the best designs can be defeated

- Undetected faults : single local fault not detected and addressed
- Dependent faults : two faults assumed to be independent are actually related
- Out-of-scope Faults : additional categories of faults not addressed by the design

The Two of Us Together

- Dependent Faults
- Two faults assumed to be independent are actually dependent
- Operations often performed twice, once for each redundant system
    - Operation error initiated in one system can be replicated to redundant system

ExampleApplication

downtime caused by zoning errors made in one fabric repeated across redundant dual-fabrics

That Can’t Happen Here

Out-Of-Scope FaultsFaults that were not anticipated in the original fault tolerant designMisconfigurations or failure to clean up old configurationsExampleLUN accidentally assigned to two different hosts, resulting in data corruptionReuse of an old HBA in a new server caused downtime because previous zonings using that HBA had not been cleaned up

A “Simple” Fault Tolerant SAN:

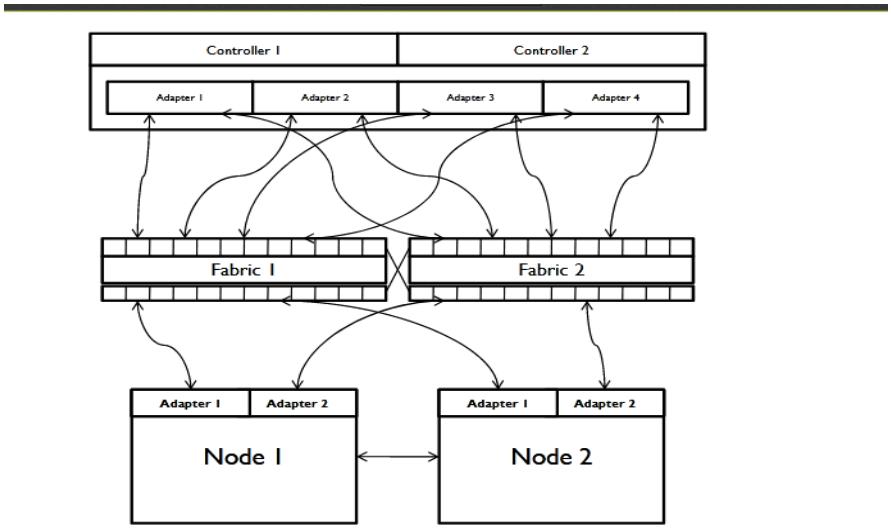
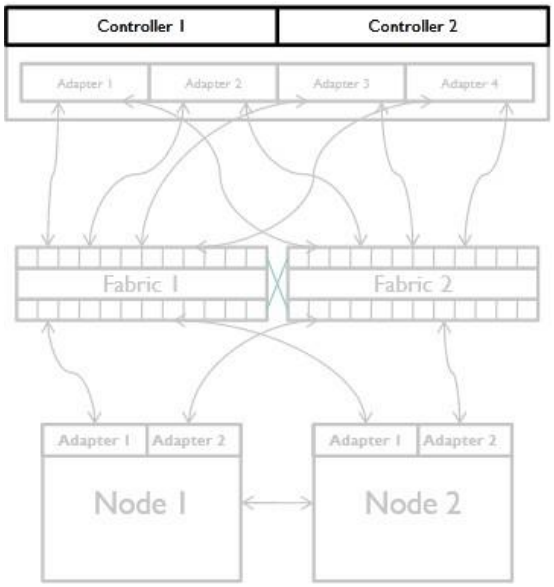


Fig No.56 A “Simple” Fault Tolerant SAN

Can’t Have Too Much Control:



- Storage Controller/Controller
  - Typical features
    - Redundant internal HDs
    - Fault Tolerant Internal Fabric
    - Hot swappable components
    - Predictive failure analysis
  - Redundant controllers allow for
    - Scheduled maintenance
    - Single controller faults
    - System upgrades
  - Single controller boxes can result in downtime if they experience a fault

Fig No.57 Storage controller

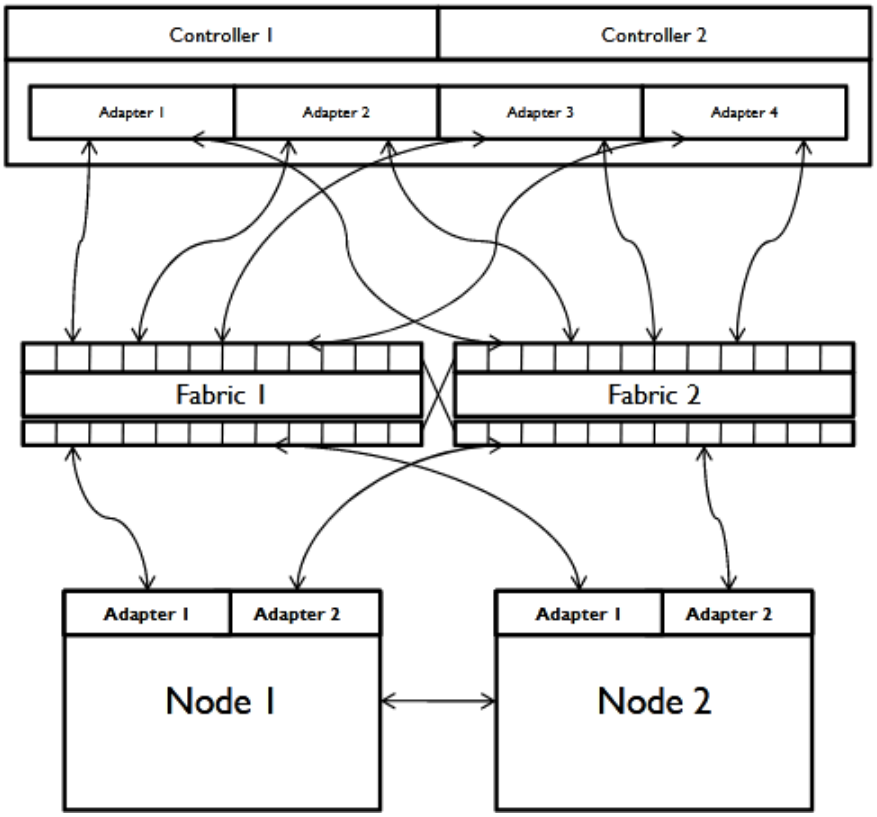


Fig No.58 Storage controller

High Availability Failover Cluster:

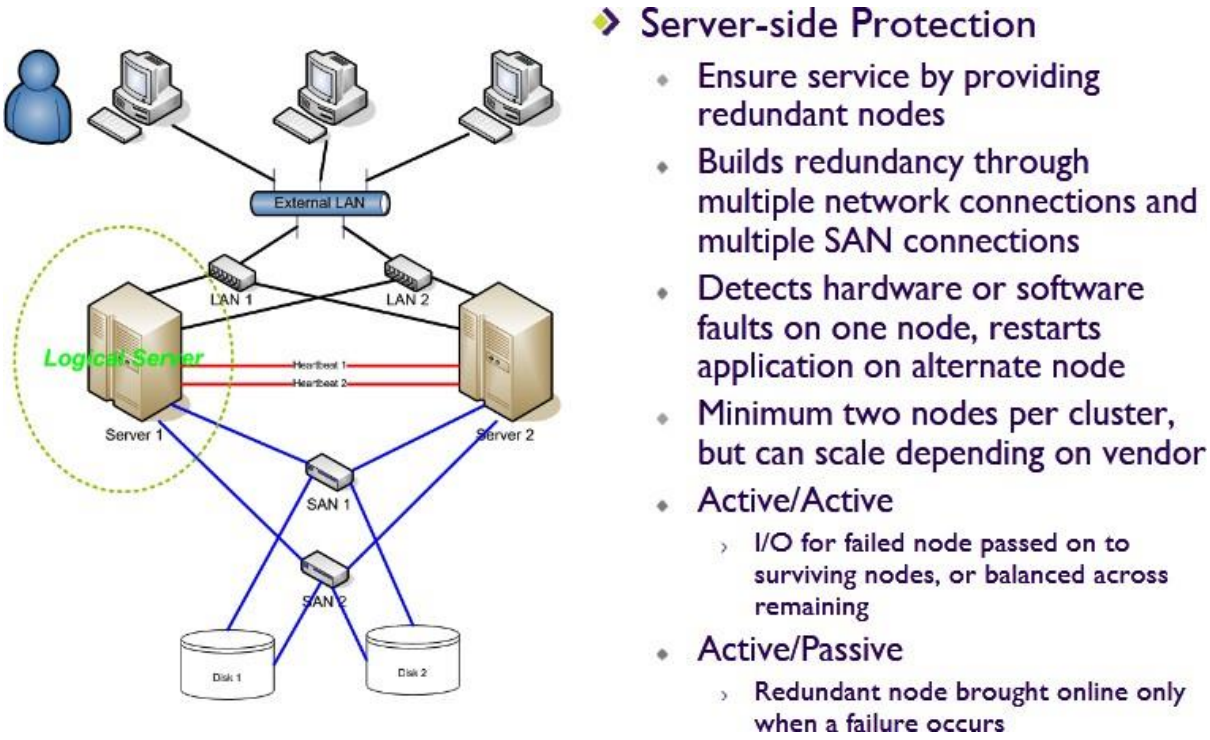


Fig No.59 Server side protection

Virtual tape library

A virtual tape library (VTL) is a technology for data backup and recovery that uses tape libraries or tape drives along with their existing software for backup. The virtual tape library system emulates the former magnetic tape devices and data formats, but performs much faster data backups and recovery. It is able to avoid the data streaming problems that often occur with tape drives as a result of their slow data transfer speeds. The VTL technology does not include physically removable disk drives, and the drives are always powered and connected to data sources. Therefore, removal to a different physical location for safe disaster recovery and storage is not possible, and the powered disk drives are always susceptible to damage and corruption from electrical power fluctuations or lightening strikes. Thus, they are never physically electrically isolated. Both of these factors are disadvantages compared to magnetic tape.

To address these disadvantages, some systems use a VTL and then back up the second hard drive disk to magnetic tape for disaster recovery protection; this is referred to as a disk-to-disk-to-tape (D2D2T) system.

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – V – VIRTUALIZATION TECHNIQUES SCSA7022

UNIT V VIRTUAL MACHINES PRODUCTS

Xen Virtual machine monitors - Xen API - VMware - VMware products - Vmware Features - Microsoft Virtual Server - Features of Microsoft Virtual Server.

XEN VIRTUAL MACHINE MONITORS:

Virtual Machines (VM):

Virtualization technology enables a single PC or server to simultaneously run multiple operating systems or multiple sessions of a single OS. A machine with virtualization software can host numerous applications, including those that run on different operating systems, on a single platform. The host operating system can support a number of virtual machines, each of which has the characteristics of a particular OS. The solution that enables virtualization is a virtual machine monitor (VMM), or hypervisor.

Xen:

ne**X**t g**EN**eration virtualization(Xen) is a virtualization system supporting both paravirtualization and hardware-assistant full virtualization. It is an open source Licensed under GPL2,

Reason to use Xen:

Xen is powered by a growing and active community and a diverse range of products and services. It offers high performance and secure Architecture.

Paravirtualization:

A software assisted virtualization technique that uses specialized APIs to link virtual machines with the hypervisor to optimize their performance. The operating system in the virtual machine, Linux or Microsoft Windows, has specialized paravirtualization support as part of the kernel, as well as specific paravirtualization drivers that allow the OS and hypervisor to work together more efficiently with the overhead of the hypervisor translations. Support has been offered as part of many of the general Linux distributions since 2008.

XEN API:

XenAPI and its associated object model has the following key features:

Management - Manages all aspects of the XenServer Host. The API allows you to manage VMs, storage, networking, host configuration and pools. Performance and status metrics can also be queried from the API.

Persistent Object Model - The results of all side-effecting operations (e.g. object creation, deletion and parameter modifications) are persisted in a server-side database that is managed by

the XenServer installation.

An event mechanism - Through the API, clients can register to be notified when persistent (server-side) objects are modified. This enables applications to keep track of datamodel modifications performed by concurrently executing clients.

Synchronous and asynchronous invocation - All API calls can be invoked synchronously (that is, block until completion); any API call that may be long-running can also be invoked asynchronously. Asynchronous calls return immediately with a reference to a task object. This task object can be queried (through the API) for progress and status information. When an asynchronously invoked operation completes, the result (or error code) is available from the task object.

Remotable and Cross-Platform - The client issuing the API calls does not have to be resident on the host being managed; nor does it have to be connected to the host over ssh in order to execute the API. API calls make use of the XML-RPC protocol to transmit requests and responses over the network.

Secure and Authenticated Access - The XML-RPC API server executing on the host accepts secure socket connections. This allows a client to execute the APIs over the https protocol. Further, all the API calls execute in the context of a login session generated through username and password validation at the server. This provides secure and authenticated access to the XenServer installation.

OBJECT MODEL OVERVIEW:

A detailed description of the parameters and methods of each class of the object model are described below.

VM - A VM object represents a particular virtual machine instance on a XenServer Host or Resource Pool.

VIF - A VIF (Virtual network InterFace) object represents an attachment between a VM and a Network object. When a VM is booted its VIF objects are queried to determine which network devices should be created. Example methods of the VIF class include "plug" (which hot plugs a network device into a running VM) and "unplug" (which hot unplugs a network device from a running guest).

VBD - A VBD (Virtual Block Device) object represents an attachment between a VM and a VDI. When a VM is booted its VBD objects are queried to determine which disk images (VDIs) should be attached. Example methods of the VBD class include "plug" (which hot plugs a disk device into a running VM, making the specified VDI accessible therein) and "unplug" (which hot unplugs a disk device from a running guest); example fields include "device" (which determines the device name inside the guest under which the specified VDI will be made accessible).

VDI - A VDI object represents a Virtual Disk Image. Virtual Disk Images can be attached to VMs, in which case a block device appears inside the VM through which the bits encapsulated by the Virtual Disk Image can be read and written. Example methods of the VDI class include "resize" and "clone".

SR - An SR (Storage Repository) aggregates a collection of VDIs and encapsulates the properties of physical storage on which the VDIs' bits reside.

PBD - A PBD (Physical Block Device) object represents an attachment between a Host and a SR (Storage Repository) object. Fields include "currently-attached" (which specifies whether the chunk of storage represented by the specified SR object) is currently available to the host

PIF - A PIF (Physical InterFace) object represents an attachment between a Host and a Network object. If a host is connected to a Network (over a PIF) then packets from the specified host can be transmitted/received by the corresponding host.

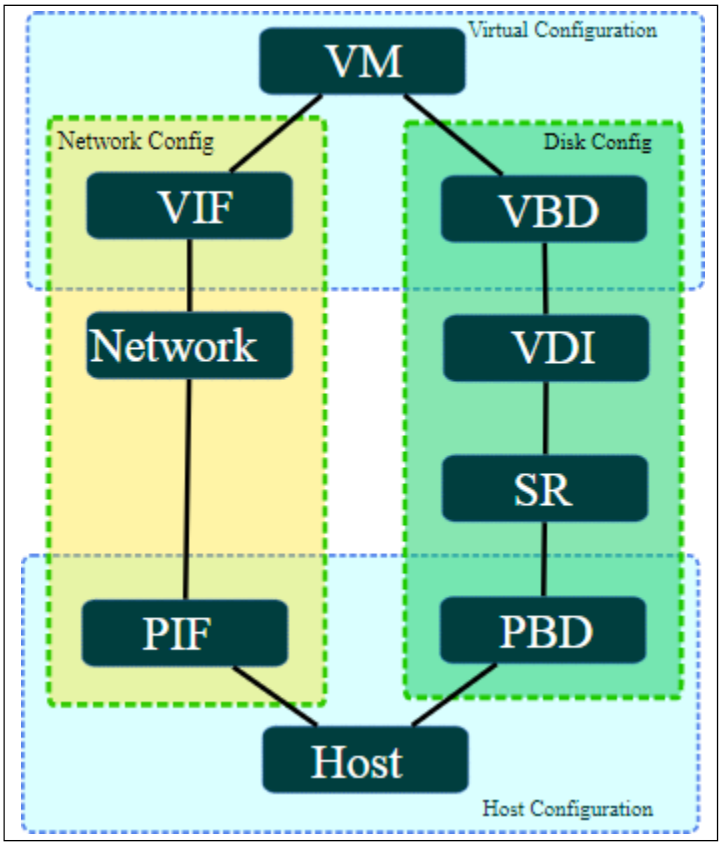


Fig No.60 Object Model

OBJECT MODEL

XAPI adds additional functionalities like

- Extending the software to cover multiple hosts.

- Enhancing the VM lifecycle, including live snapshots, VM checkpointing, and VM migration.
- Enabling resource pools to include live migration, auto configuration, and disaster recovery.
- Allowing flexible storage and networking including integrated Open vSwitch support and storage XenMotion® live Migration (cross-pool migration, VDI migration)
- Enabling event tracking, with progress and notification.
- Creating upgrade and patching capabilities.
- Facilitating real-time performance monitoring and alerting.
- Integrations with cloud orchestration stacks.
- Built-in support and templates for Windows and Linux guests.

VMWARE:

VMware Infrastructure is a full infrastructure virtualization suite that provides comprehensive virtualization, management, resource optimization, application availability, and operational automation capabilities in an integrated offering. VMware Infrastructure virtualizes and aggregates the underlying physical hardware resources across multiple systems and provides pools of virtual resources to the data center in the virtual environment.

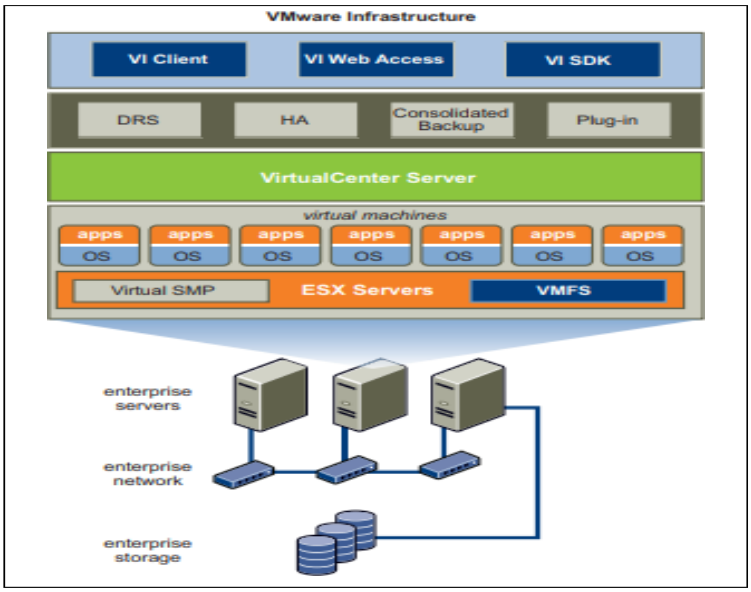


Fig No.61 VM infrastructure

VMWARE INFRASTRUCTURE ARCHITECTURE

VMware ESX Server – A robust, production-proven virtualization layer run on physical servers that abstracts processor, memory, storage, and networking resources into multiple virtual machines. Two versions of ESX Server are available:

„ 1. ESX Server 3 contains a built-in service console. It is available as an installable CD- ROM boot image.

„ 2. ESX Server 3i does not contain a service console. It is available in two forms.ESX Server 3i Embedded and ESX Server 3i Installable. ESX Server 3i Embedded is firmware that is built into a server's physical hardware. ESX Server 3i Installable is software that is available as an installable CD-ROM boot image.

VirtualCenter Server – The central point for configuring, provisioning, and managing virtualized IT environments.

VMware Infrastructure Client (VI Client) – An interface that allows users to connect remotely to the VirtualCenter Server or individual ESX Servers from any Windows PC.

VMware Infrastructure Web Access (VI Web Access) – A Web interface that allows virtual machine management and access to remote consoles.

VMware Virtual Machine File System (VMFS) – A high-performance cluster file system for ESX Server virtual machines.

VMware Virtual Symmetric Multi-Processing (SMP) – Feature that enables a single virtual machine to use multiple physical processors simultaneously.

VMware VMotion and VMware Storage VMotion – VMware VMotion enables the live migration of running virtual machines from one physical server to another with zero down time, continuous service availability, and complete transaction integrity. VMware Storage VMotion enables the migration of virtual machine files from one data store to another without service interruption.

VMware High Availability (HA) – Feature that provides easy-to-use, cost-effective high availability for applications running in virtual machines. In the event of server failure, affected virtual machines are automatically restarted on other production servers that have spare capacity.

VMware Distributed Resource Scheduler (DRS) – Feature that allocates and balances computing capacity dynamically across collections of hardware resources for virtual machines. This feature includes distributed power management (DPM) capabilities that enable a datacenter to significantly reduce its power consumption.

VMware Consolidated Backup (Consolidated Backup) – Feature that provides an easy-to-use, centralized facility for agent-free backup of virtual machines. It simplifies backup administration

and reduces the load on ESX Servers.

VMware Infrastructure SDK – Feature that provides a standard interface for VMware and third-party solutions to access the VMware Infrastructure.

PHYSICAL TOPOLOGY OF VI DATACENTER:

Typical VMware Infrastructure datacenter consists of basic physical building blocks such as x86 computing servers, storage networks and arrays, IP networks, a management server, and desktop clients.

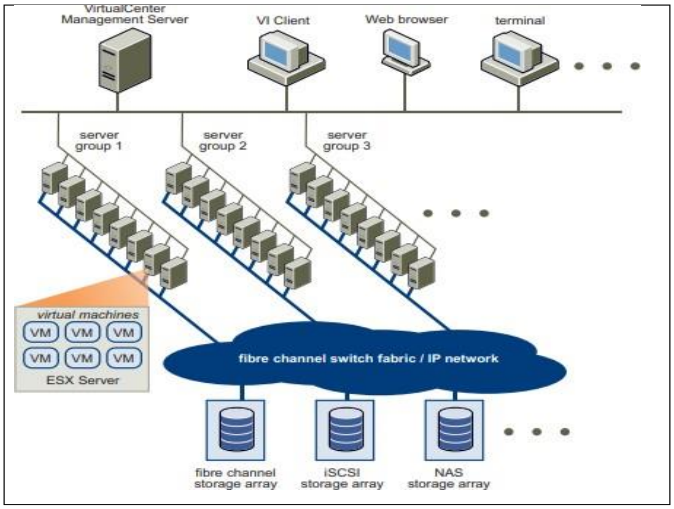


Fig No.62 Physical Topology

Computing Servers

The computing servers are industry standard x86 servers that run VMware ESX Server on the bare metal. ESX Server software provides resources for and runs the virtual machines. Each computing server is referred to as a standalone host in the virtual environment. A number of similarly configured x86 servers can be grouped together with connections to the same network and storage subsystems to provide an aggregate set of resources in the virtual environment, called a cluster.

Storage Networks and Arrays

Fiber Channel SAN arrays, iSCSI SAN arrays, and NAS arrays are widely used storage technologies supported by VMware Infrastructure to meet different datacenter storage needs. Sharing the storage arrays between (by connecting them to) groups of servers via storage area networks allows aggregation of the storage resources and provides more flexibility in provisioning them to virtual machines.

IP Networks

Each computing server can have multiple Ethernet network interface cards (NICs) to provide high bandwidth and reliable networking to the entire datacenter.

VirtualCenter Server

The VirtualCenter Server provides a convenient single point of control to the datacenter. It provides many essential datacenter services such as access control, performance monitoring, and configuration. It unifies the resources from the individual computing servers to be shared among virtual machines in the entire datacenter. It accomplishes this by managing the assignment of virtual machines to the computing servers and the assignment of resources to the virtual machines within a given computing server based on the policies set by the system administrator. Computing servers will continue to function even in the unlikely event that VirtualCenter Server becomes unreachable (for example, the network is severed). They can be managed separately and will continue to run the virtual machines assigned to them based on the resource assignment that was last set. After the VirtualCenter Server becomes reachable, it can manage the datacenter as a whole again.

Desktop Clients

VMware Infrastructure provides a selection of interfaces for datacenter management and virtual machine access. Users can choose the interface that best meets their needs:VMware Infrastructure Client (VI Client), Web Access through a Web browser, or terminal services (such as Windows Terminal Services) becomes reachable, it can manage the datacenter as a whole again.

VMWARE PRODUCTS:

VMware's product line provides a sometimes-confounding array of packages and features, which can be difficult for a virtualization administrator to navigate. This VMware products and features overview demystifies the virtualization platform by breaking down the capabilities of VMware's feature sets as well as how VMware bundles these features into its various virtualization products. VMware's product line is divided into two categories.

1. Data center / server virtualization
2. Desktop virtualization

Data center / server virtualization:

1. **Free ESXi edition:** the free version of VMware ESX/ESXi that allows you to consolidate servers while still using VMware's enterprise-grade hypervisor

2. **vSphere 4/ESX Server:** includes ESX and ESXi plus a number of features, depending on the edition of the vSphere suite that you select. Some vSphere features are:

VMotion: moved running virtual machines (VMs) from one server to another

Storage VMotion (SVMotion): moves the virtual disks of a running virtual machine from one data store to another

VMware High Availability (or VMware HA, VMHA): reboots running VMs on another ESX server if an ESX host goes down

Fault Tolerance (FT): moves a running VMs from one ESX server to another if the server they run on goes down

Distributed Power Management (DPM): when demand is low on a virtual infrastructure, running VMs are consolidated onto fewer servers, and unused servers are powered off

VMware Consolidated Backup (or VCB): this VMware backup tool enables you to back up running virtual machines using an existing backup application

vShield Zones: creates a virtual firewall within your virtual infrastructure.

3. vCenter Server: the centralized management server that manages all ESX servers and enables most vSphere features

4. VMware Server: a free virtualization platform that runs in an existing Windows or Linux operating system.

Desktop virtualization:

1. VMware View (includes VMware VDI): used to consolidate desktop PCs into your virtual infrastructure

2. VMware Workstation: allows you to run multiple operating systems on your desktop PC. Few other

products most popularly used are vSphere (server virtualization), Horizon
View (Desktop virtualization), ThinApp (Application virtualization), vCloud Director (Cloud),vRA (Private Cloud), vCloud Air (Public cloud), VMware Workstation (Type 2 virtualization), NSX (Network virtualization),vSan (Storage Virtualization).

VMWARE FEATURES:

There are some advanced features VSystems uses regularly or has implemented within VMware that greatly increase the desirability of a virtual infrastructure over a traditional physical server environment. The following features work within a VMware environment to improve performance, streamline efficiency and improve consistency that few non-virtual environments can manage without taking on massive resources:

- **High Availability** (HA) : High Availability improves the reliability of the environment. The

purpose of Highly Available servers is to reduce downtime in case of a hardware failure. VMware vSphere Hypervisor ESXi hosts that are arranged in clusters share resources of the machines held within them. If a host experiences unexpected downtime, the virtual machines on that host automatically begin running on an alternate ESXi host within that cluster. When HA comes into play, a machine is migrated to (and restarted on) an alternate host. High Availability provides a method to keep virtual machines functioning even in the case of a hardware failure. This is an incredibly important feature for environments that cannot live with downtime on their machines.

- **Fault Tolerance** (FT) : Like High Availability,Fault Tolerance allows a virtual machine (VM) to persist through a hardware failure.

High Availability Fault Tolerance: Utilizes resources so in case of a hardware failure VMs can be powered on from a new host

Fault Tolerance : A live shadow instance of a VM running on a secondary host.Fault Tolerance allows the VM to continue to run, even if a host fails suddenly, without any loss of data or connectivity to the end user or the VM.

- **vMotion :** vMotion is a feature within VMware that allows the live migration of a virtual machine from one ESXi host to another without interrupting the services the VM provides. There is little to no interruption in service while using vMotion to migrate a VM – usually only a few packets are lost, and the end user should not even notice the transition. It allows administrators to remove VMs from a host that may be failing or not quite performing as well as it should be.

- **Storage vMotion :** Storage vMotion is a similar feature to vMotion, however, it is used for migrating data to another datastore on a connected disk. This feature performs a similar role to vMotion, but provides administrators with the ability to manage storage issues, such as high latency, before they become an issue within an environment.

- **Distributed Resource Scheduler** (DRS) : Distributed Resource Scheduler (DRS) is a load-balancing feature. DRS utilizes vMotion to automatically allow a cluster of VMware ESXi hosts to distribute the compute workload across the environment. DRS gauges the combined vCPU and vRAM usage amongst virtual machines running in the environment and spreads them across the hosts in the most efficient manner. This makes sure that the resources of an individual host are not being overburdened, while another host is performing under a light load.

MICROSOFT VIRTUAL SERVER:

Microsoft released Hyper-V in 2016 and it is considered a hot competitor to VMware's Fusion and Oracle's VM VirtualBox. Hyper-V is virtualization software that, well, virtualizes software. It can not only virtualize operating systems but also entire hardware components, such as hard drives and network switches. Unlike Fusion and Virtualbox, Hyper-V is not limited to the user's device. You can use it for server virtualization, too.

Hyper-V is available in three versions.

1. Hyper-V for Windows Servers
2. Hyper-V Servers
3. Hyper-V on Windows 10

Hyper-V for Windows Servers is an add-on to the Windows Server OS. Hyper-V Servers, on the other hand, is a standalone solution that can be used to manage virtual and dedicated server instances, just like Hyper-V for Windows Servers. Hyper-V on Windows 10 is the version that runs on your laptop and the subject of this article. To enable Hyper-V on your Windows device, you need a 64-bit OS. It doesn't have to be Windows 10, though. Windows 8.1 works too.

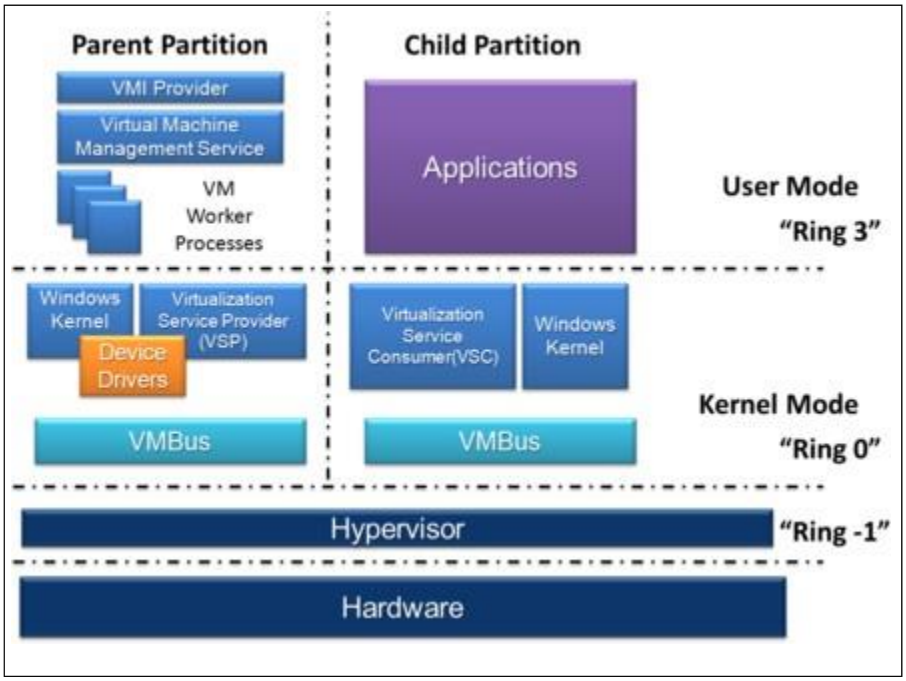


Fig no.63 Microsoft virtual server

Hyper V Architecture

Hyper-V implements isolation of virtual machines in terms of a partition. A partition is a logical unit of isolation, supported by the hypervisor, in which each guest operating system executes. There must be at least one parent partition in a hypervisor instance, running a supported version of Windows Server. The virtualization software runs in the parent partition and has direct access to the hardware devices. The parent partition creates child partitions which host the guest OSs. A parent partition creates child partitions using the hypercall API, which is the application programming interface exposed by Hyper-V.

A child partition does not have access to the physical processor, nor does it handle its real interrupts. Instead, it has a virtual view of the processor and runs in Guest Virtual Address, which, depending on the configuration of the hypervisor, might not necessarily be the entire virtual address space. Depending on VM configuration, Hyper-V may expose only a subset of the processors to each partition. The hypervisor handles the interrupts to the processor, and redirects them to the respective partition using a logical Synthetic Interrupt Controller (SynIC). Hyper-V can hardware accelerate the address translation of Guest Virtual Address-spaces by using second level address translation provided by the CPU, referred to as EPT on Intel and RVI (formerly NPT) on AMD.

Child partitions do not have direct access to hardware resources, but instead have a virtual view of the resources, in terms of virtual devices. Any request to the virtual devices is redirected via the VMBus to the devices in the parent partition, which will manage the requests. The VMBus is a logical channel which enables inter-partition communication. The response is also redirected via the VMBus. If the devices in the parent partition are also virtual devices, it will be redirected further until it reaches the parent partition, where it will gain access to the physical devices. Parent partitions run a Virtualization Service Provider (VSP), which connects to the VMBus and handles device access requests from child partitions. Child partition virtual devices internally run a Virtualization Service Client (VSC), which redirect the request to VSPs in the parent partition via the VMBus. This entire process is transparent to the guest OS.

Virtual devices can also take advantage of a Windows Server Virtualization feature, named Enlightened I/O, for storage, networking and graphics subsystems, among others. Enlightened I/O is a specialized virtualization-aware implementation of high level communication protocols, like SCSI, that allows bypassing any device emulation layer and takes advantage of VMBus directly. This makes the communication more efficient, but requires the guest OS to support Enlightened I/O.

FEATURES OF MICROSOFT VIRTUAL SERVER.:

The following additional features are specific to Microsoft Virtual Server. These include:

1. Loads and runs during start-up - Virtual Server runs as a service in Windows Server 2003 and will automatically load and run during start-up before any administrator user has logged on.
2. Large memory support for virtual machine sessions - Virtual Server provides support for up to 3.6 gigabytes of RAM per virtual machine session, up to the limit of RAM supported by the server operating system running Virtual Server.

3. Support of up to 64 sessions - Virtual Server can run up to 64 virtual machine sessions simultaneously, provided the server running Virtual Server has adequate resources.
4. Virtual networking - Virtual Server supports up to 4 virtual Ethernet NICs per virtual machine session, and an unlimited number of virtual networks.
5. Virtual SCSI disk support - Virtual Server supports up to 4 virtual SCSI controllers per

virtual machine session, and each controller can support up to 7 virtual hard disks. These are in addition to the virtual IDE hard disks supported by Virtual Server.

6. Resource limiting and reservations - Virtual Server can set minimum and maximum CPU usage limits separately for each virtual machine session.
7. Web interface controls and virtual machine Remote Desktop administration - Virtual Server provides a native Virtual Machine Remote Control (VMRC) that allows remote users to manage any type of virtual machine session desktop from within a browser.
8. Virtual machine threading- All virtual machine sessions are threaded to take advantage of multiple processors in the Virtual Server's host server hardware.

9. COM-based architecture and scripting support- Virtual Server is based entirely on a COM architecture. This allows developers to control Virtual Server with any language that can control COM objects, such as Visual Basic.NET, Visual C#, and so on.
10. Crash detection, recovery, logging, and monitoring- Virtual Server can detect when a virtual machine session crashes and initiate event notices to external scripts. Extensive logging is available for each virtual machine session.
11. Pmon and Microsoft Management Console (MMC) integration - Extensive logging, performance monitoring, and resource monitoring are directly accessible from Pmon.