# Xtended Volume Management

Colin Ngam, Silicon Graphics, Inc.

**ABSTRACT:** *This paper provides an overview of the design and architecture of Silicon Graphics next generation disk Logical Volume Manager(XVM). XVM is an extension of XLV, which is the current generation of SGI's disk Logical Volume Manager. This paper also addresses any differences between XVM and UNICOS/UNICOSmk disk Logical Device Drivers.*

## Introduction

Xtended Volume Managent(XVM) is a Logical Volume Manager for disk devices. XVM is a set of logical device drivers that provide the capability to glue together many pieces(slices) of Physical Disks into one(1) contiguous logical device that can be accessed via the standard character device(cdevsw()) and block device(bdevsw()) interfaces. This layer of abstraction provides a single contiguous logical address space that can span a large number of same/different physical devices e.g. SCSI disks, Disk Arrays, HIPPI Disks etc..

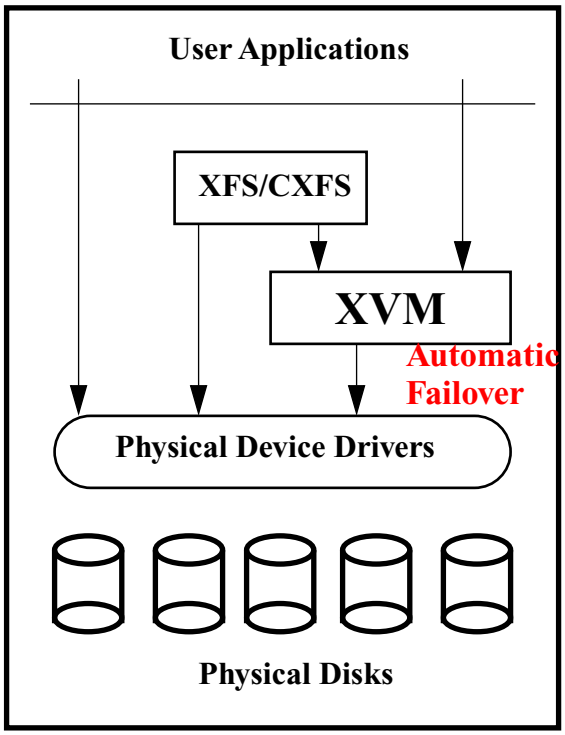
XVM is the next generation product of XLV with special emphasis on Cellular IRIX support.

The design of XVM also includes features available in UNICOS and UNICOS/mk logical device drivers.

## XVM IO Model

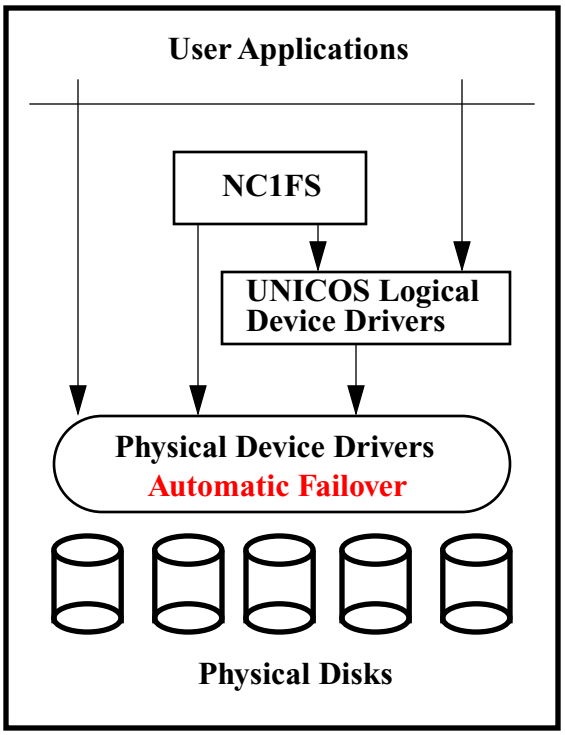
The diagram below presents the IO Model provided by XVM. Note that raw device access is still available and path failover(alternate pathing) is done by XVM.

Diagram 1 - XVM IO Model.




XVM IO Model is almost identical to the IO model supported by UNICOS or UNICOS/mk. The only real difference is that in UNICOS and UNICOS/mk, the path failover(alternate path) is handled by the Physical device drivers, while in XVM, the failover function is initiated by XVM. Therefore, in IRIX, if the application uses raw disk devices rather than XVM logical devices, path failover is not available to the application.

Diagram 2 - UNICOS IO Model.




## Logical Subvolume Concept

Subvolumes enable raw character and block devices to span multiple disk slices. Subvolumes behave like regular devices; they provide a character and block device interface to the application. File systems can be created on subvolumes, mounted and used. Just like a simple regular disk devices.

Applications can also use subvolumes as raw character or block devices. Subvolumes have names e.g. /hw/xvm/vol/<volname>/<subvol_name>/char for character devices and /hw/xvm/vol/<volname>/<subvol_name>/block for block devices. These names can be used in any interface that supports a device

## Features

There are many advantages for using XVM devices rather than interfacing directly with the underlying physical device drivers. These advantages are: user defined names, bigger file systems, increased aggregate bandwidth, redundant storage, error recovery, extended XFS/CXF support, distributed cellular IRIX support etc.

***User Defined Names***

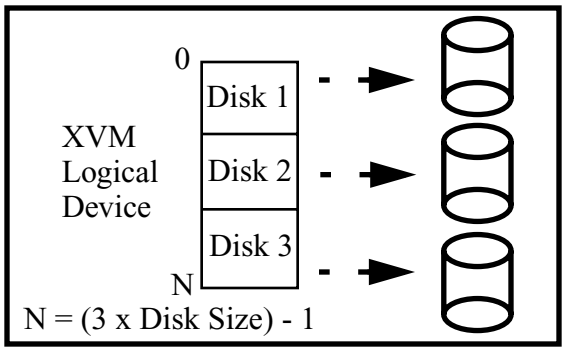
Devices in XVM can have User defined names. If a name is not given, XVM will provide a default name. All devices in XVM can have user defined names except for physical slices. Each physical slice name is generated from the name of the disk post fix with an s followed by a numeric number e.g. usrdisks202, where usrdisk is the name of the disk and s202 is system generated.

***Bigger Devices***

Without XVM, device sizes will be restricted to the actual size of the physical device. XVM provides the capability to glue together pieces of these physical devices together, providing 1 logical contiguous address space from 0 to (Aggregate Size of all the Physical Devices -1). This capability enables XFS and CXFS to create file systems that can span the size of any single physical device. This feature is implemented with XVM concatenated device.

The diagram below shows how we can create an XVM device(for a file system) that spans 3 physical disks

Diagram 3 - XVM Concatenated Device.

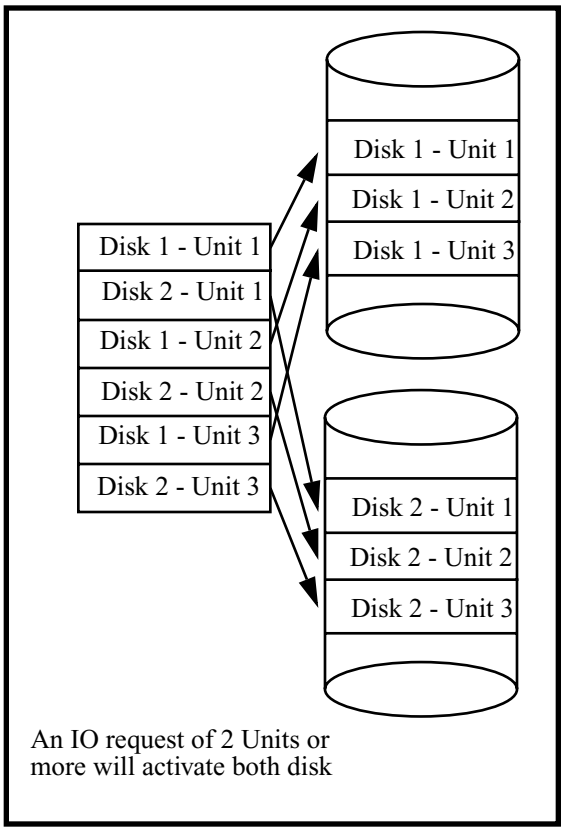



***Increased Aggregate Bandwidth***

The bandwidth of a device is restricted by the physical characteristics of that device e.g. a SCSI disk may provide 10 Megabytes per second. XVM provides the capability to activate multiple devices at the same time from a single IO request,, effectively increasing the aggregate bandwidth by the number of phycial devices activated. If an IO request spans across 5 SCSI devices with 10 Mb/sec each, the theoretical bandwidth can be 5 x 10 = 50 Mb/sec of aggregate bandwidth. This capability in XVM is provided by striping these devices together. More information regarding striping is provided in the later part of this document.

The diagram below shows a 2 disk stripe device, with capability of an aggregate bandwidth of (2 x Single Disk's).

Diagram 4 - Stripe Device.




***User Defined Stripe Factor***

In XVM, the stripe factor(also known as stripe unit) can be defined by the user. If the stripe factor is not specified, the default stripe factor is used.

***Stripe Width***

XVM supports stripe devices with 65K members. Although we do not envision such a wide stripe width will ever be used, it is an extension to XLVs suuport of only 100 members.

***Redundant Storage***

XVM provides the capability to automatically replicate data at the mainframe level. This capability is called Mirroring. This capability provides redundant storage access and recovery in the event of a catastrophic broken path to the device or physical media failure. XVM supports up to 65K members in any mirror device.

***Dynamic Mirror Insertion***

XVM supports dynamic mirror insertion. A mirror device can be inserted into any position of the hierarchy. This effectively converts the original device into a member of the mirror.

***Stackable Devices***

Concatenated, Mirrors and Stripe devices can be stacked in any order and at any depth. Users can build devices that consist of any hierarchy e.g. a mirror device of stripe devices of concatenated devices etc.

***Error Recovery***

IRIX physical disk device drivers does not automatically failover to the alternate path in the event of a path failure. XVM provides path failover logic for all XVM devices. XVM users are assured that for multipathed devices, XVM will perform automatic path failover.

***Extended XFS/CXFS Support***

Special support are provided to the IRIX XFS file system. XVM provides 4 special subvolume types: data, log, real-time and secondary partition. These special subvolume types will be discuss in more details in later part of this document.

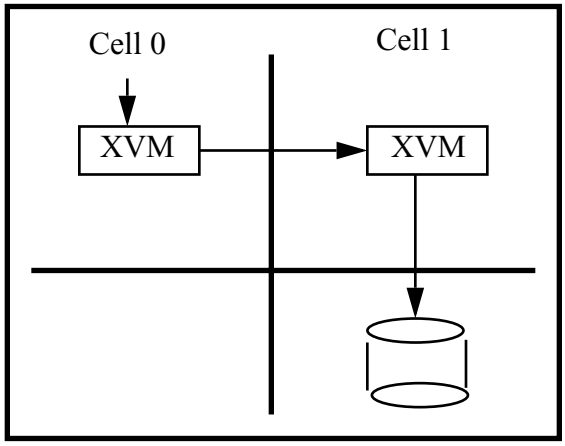
***Distributed Cellular IRIX Support***

In Cell IRIX, XVM provides transparent device distribution to it's applications. All XVM accesses are performed locally. If the underlying physical device is on another host, XVM will automatically redirect the IO appropriately. IO redirection is supported only in Cell IRIX - HPC. All disk devices in Cell IRIX - Cluster must be directly connected to all members.

XVM replicates it's configuration on all coorporating hosts.

The diagram below shows IO forwarding by XVM in Cell IRIX - HPC. An IO request from Cell 0 for a disk that is only connected to Cell 1.

Diagram 5 - XVM IO Forwarding in Cell IRIX.




In Cell IRIX - Cluster, all disks must be directly connected to all cells. XVM does not perform IO forwarding in Cell IRIX - Cluster environment.

***Hot Plug***

After the system has been booted, an XVM disk can be attached to the system and it's configuration will be automatically read, added and brought online.

***Hot Swap***

A disk experiencing media errors can have it's data restored onto another disk(mirroring). The bad disk can then be removed from the system.

***Root File System***

Root file systems using XVM devices can have multiple partitions. This feature will allow the file system to grow and span across multiple disks.

***Swap***

Swap devices using XVM can also have multiple partitions.

## XVM Software Architecture

***Volume Elements***

All XVM objects are called Volume Elements, except for physical disks. Physical disks objects are called Physical Volumes.

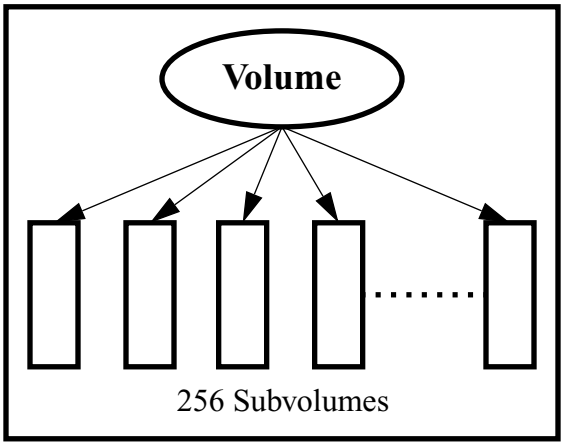
Volume Elements are composed of a hierarchy of logical storage objects; the highest Volume Element being a Volume and the lowest Volume Element being a Physical Disk Slice. Volumes, subvolumes, concatenated objects, stripe objects, mirror objects, physical slices are all Volume Elements.

***Volume***

The Volume is the top level object. The Volume object has the following capabilities:

- The Volume has a User defined name.
- Group a related set of subvolumes.
- Can consist upto 256 subvolumes

Diagram 6 - XVM Volume.




The Volume does not have an address space. It's primary function is to group related set of subvolumes for the XFS/CXFS file system.

The XFS/CXFS file system can have up to 4 related subvolumes in a single file system:

1. Data Subvolume.
2. Log Subvolume.
3. Real-Time Subvolume.
4. Secondary Partition Subvolume.

This association is neccessary because the data, real-time and secondary partition subvolume shares the same file system name space. Given the Volume name, XVM can inform the application the set of associated subvolumes.

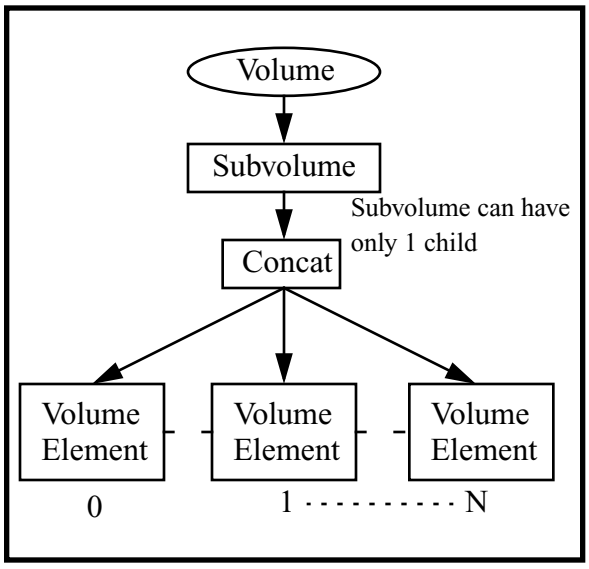
***Subvolume***

The subvolume is the second highest level object. Subvolumes have the following attributes:

- The Subvolume has a user defined name.
- Each subvolume has it's own address space( 0 to subvolume size - 1).
- Subvolume is the XVM character and raw device. These devices can be accessed using the cdevsw() and bdevsw() interfaces.
- Subvolumes is the only IO interface into XVM.
- A subvolume can have only 1 child. Any of the following Volume Elemnt can be a child of the subvolume: concatenate, mirror, stripe and physical slice. However, the Concatenated Volume Element is normally used as the child of the Subvolume. This allows the subvolume to grow/shrink by adding/deleting child volume elements to/from the concatenated volume element.

The diagram below shows a subvolume with a concatenated volume element as it's child and the children of the concatenated volume element.

Diagram 7 - XVM Subvolume.



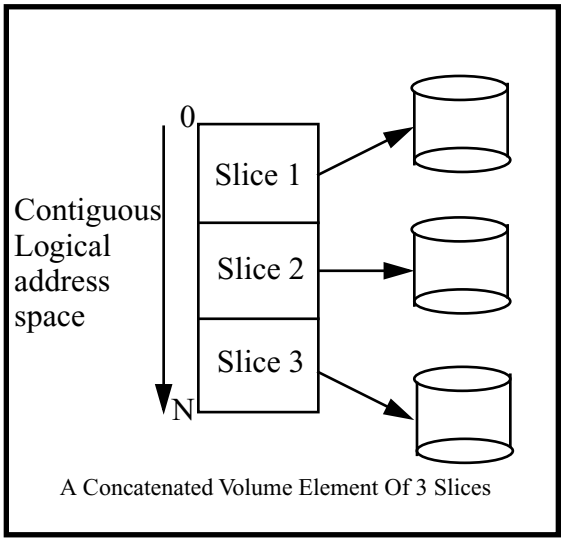

***Concatenation***

The concatenated volume element can be at any level after the subvolume(XVM does not have a tree dept restriction). This volume element is normally used as the only child of the subvolume. Concatenated volume element has the following attributes:

- User defined names.
- Can have other concatenated volume element, stripe, mirrors of physical slices as it's children.
- Provides the glueing capability in XVM.

The following diagram shows how a concatenated volume element glues together 3 physical slices, each from a different disk.

Diagram 8 - Concatenated Device.




## Mirrors

Mirror Volume Element provides the capability to replicate data on two or more separate members. These members can be any XVM Volume Elements. In the event of a catastrophic failure on one of it's members, data can be retrieved from any member of the mirror.

XVM provides the capability to support up to 65K mirror members.

## Stripes

Stripe Volume Element provides the capability to increase the aggregate bandwidth. Without stripe Volume Elements, the maximum bandwidth of a logical device is restricted to the bandwidth of a single disk. A stripe Volume Element of 5 disk devices can provide an aggregate bandwidth of 5 times the bandwidth of a single disk device.

XVM supports a maximum stripe membership of 65K.

Please refer to Diagram 4 - Strpe Device, for a pictorial represenation of a stripe device.

## Physical Slices

Physical slices are the lowest level Volume Element in XVM. A Physical Slice Volume Element describes a particular portion of a disk.

## XVM Configuration

Each XVM disk has an area called the XVM Label File. A XVM disk is a disk that has been explicitly given to XVM for management.

Each disk's XVM Label File describes the configuration of all the slices of that disk. XVM configuration, unlike UNICOS or UNICOS/mk, are not kept in one place. XVM configuration is build as each disk is brought online, and trees that are complete are available for IO e.g. mounted as file systems. The XVM Label File is initialized when a disk is given to XVM. The XVM Label File is updated as configuration changes are made for that disk.

The complete XVM configuration is derived from all the XVM Label Files on all the XVM disks.

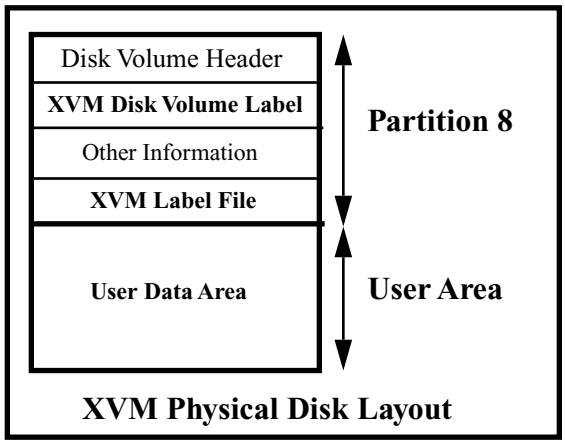
# XVM Disk

XVM disks are disks that are managed(partitioned) by XVM. When a disk is given to XVM, it is initialized with:

- XVM Disk Volume Label
- XVM Label File

The following diagram shows an initialized XVM disk.

Diagram 9 - XVM Disk Layout.




An XVM disk is self identifying. The iopath to the disk is initialized when the disk is discovered. XVM disk can be removed and inserted anywhere in the system without any configuration changes. This feature is not supported in UNICOS or UNICOSmk.

## XVM Disk Volume Label

The XVM Disk Volume Label contains attributes that describe the disk These attributes includes:

- Name of the disk.
- Size of the XVM Label File.
- Start and length of the User Area.

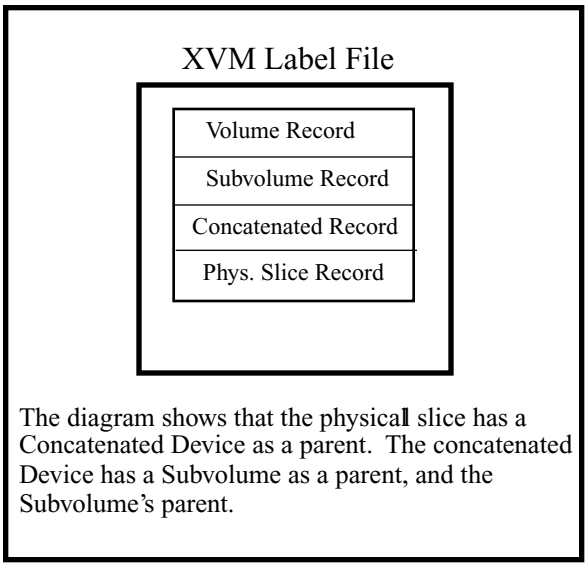
## XVM Label File

The XVM Label file contains record entries that describes:

- Physical slices of the disk.
- Hierarchy parents of these physical slices.

The XVM Label File contains the complete hierarchy of a configured physical slice. It does not contain who the peers are e.g. a 2 slice concatenated device, each slice from each disk. The following diagram shows all the configuration record entries for 1 of the slice on the disk.

Diagram 10 - XVM Configuration Records.




Notice that the configuration of the other slice is not given on this disk. Therefore, configuration for this volume is not complete by just reading the configuration from this disk. XVM will have to read the second slice configuration from the other disk to complete the configuration.

## Conclusion

Functionalities and features supported in XLV, UNICOS and UNICOSmk are also supported in XVM. XVM provides extended features like: stackable logical devices, mirror, concat and stripe Volume Elements can have up to 65K members, dynamic mirror insertion, virtually unlimited number of physical slices per disks, Cell IRIX support with a fully replicated configuration on all hosts etc..