

USP V and XP24000 Unofficial Manual



<http://blogs.rupturedmonkey.com>

Version 0.003

**!EARLY
DRAFT!**



WARNING! Very early draft version. Intended only to see if there is interest for such a document and to tout potential collaborating authors

DISCLAIMER: This manual is written as a community effort drawing on volunteers from the HDS and HP storage communities. As such it is not an official Hitachi, HDS or HP document and should not be considered authoritative. Technical accuracy cannot always be guaranteed although every effort will be made to ensure accuracy of information.

Document ownership

At the moment the document is owned by nigel@rupturedmonkey.com but the content ~~has been~~ will be provided by many people from the HDS and HP storage communities. For a list of contributing authors see Appendix A. At initial version I will own the document and all changes should go through me. However, future versions may be different such as each section having a principal author/owner.....

Want to contribute?

If you would like to contribute content, feel free to contact me at nigel@rupturedmonkey.com and let me know your experience as well as which areas you would like to contribute to. **Please do not provide content that is Hitachi, HDS or HP confidential or protected by NDA.**

Contributions are welcome for all areas of the document but especially areas marked with –

<Please contact me if you would like to write or contribute to this section>

Also if there are any budding artists out there, the document needs good quality diagrams.

Notes about this version

This is a **very** early draft version. It has not been proof read or had its technical content reviewed. I know that it is generally bad form to release such an un-checked document for download. However, I have several reasons for doing so. One reason is to judge how much interest and potential demand there is for the document (before I/we invest a lot of time and effort). I hope this early release will peak the interest of others who may wish to contribute content as well be proof readers and give feedback.

Terminology

Throughout this document, HDS terms and technology names will be used. For a cross-reference between HDS and HP terms see Appendix C.

CONTENTS

- 0 Introduction and Objective**
- 1 Hitachi, HDS, HP and Sun**
- 2 USP V High Level Overview**

HARDWARE SECTION

- 3 Major Hardware Components**
 - 3.1 Hardware High Level Basics**
 - 3.2 Front End Director**
 - 3.3 Cache Switch**
 - 3.4 Cache**
 - 3.5 Shared Memory**
 - 3.6 Switched Back End**
 - 3.7 Back End Directors**
 - 3.8 Disks and Array Groups**
- 4 Logical Data Flow**
- 5 The LUN and how it is made**

SOFTWARE SECTION

- 6 Microcode**
- 7 HDP**
- 8 UVM**
- 9 ShadowImage**
- A HUR**
- B TrueCopy**
- C Copy-On-Write**
- D ??**
- E ??**
- F ??**
- 10 ??**
- 11 ??**

Appendix A: Contributing authors

Appendix B: RAID implementations

Appendix C: Cross reference between HDS and HP terminology

0. Introduction and Objective

The reason I set out to write this manual is threefold –

1. There are too many people out there who have responsibility for USP Vs and XPs who know nothing about how they work and struggle to find decent information.
2. I wish something like this had existed when I was learning my first HDS enterprise array.
3. I'm actually sad enough to enjoy writing something like this.

I'm quite passionate about storage and participate in both HDS and HP storage forums as well as talk with lots of users via email re USP Vs and XPs and help wherever I can. I find myself repeating the same things over and over again and figured writing it down in a single document might help. That way I can just point people to the doc and it will, in some cases, literally answer their prayers.

When I started out in HDS storage no such document existed, and in fact neither did the HDS storage forums. So I had to learn the hard way. Despite this I thought I'd share my hard earned knowledge because I'm a nice guy. Seriously.

This document makes no attempt to guide the reader through how to configure the USP V in a "click this button" type manner. Instead it is intended to provide an overview of the hardware and software that make up the USP V and give the reader sufficient knowledge to understand and hold their own with the array and to enable people to ask the right questions of their HDS and HP representatives.

Comments and feedback are welcome mandatory. I can be contacted at via either of the following methods –

<http://blogs.rupturedmonkey.com>

nigel@rupturedmonkey.com

Please feel free to contact me with any feedback, thoughts, suggestions and corrections.

1. Hitachi, HDS, HP and Sun

What have all four of the above companies got in common?

Well..... Hitachi Ltd is the, Tokyo Japan based, company that designs and builds the USP V. The designers and engineers etc (the “factory”) live and work for Hitachi Ltd. The factory name for the USP V is the RAID600. When all is said and done, it is a Hitachi box.

At the time of writing, HDS, HP and Sun Microsystems all sell the RAID600, only under their own marketing names so called re-badging. They are often referred to as “vendors”.

- HDS sells it as the USP V
- HP sells it as the HP StorageWorks XP24000
- Sun Microsystems sell it as the Sun StoreageTek 9990V

Aside from each of the above vendors having different front doors on the unit, the USP V, XP24000 and 9990V are the same thing – the Hitachi RAID6000.

At a very high level the agreements between the different companies is as follows –

HDS (Hitachi Data Systems) is a wholly owned subsidiary of Hitachi Ltd and has very close ties with Hitachi Ltd. They are a storage centric company and act much like a sales and marketing department for the Hitachi storage business.

HP (Hewlett Packard) has an extensive OEM agreement with Hitachi Ltd.

Sun Microsystems is a reseller for HDS.

There may also be occasional differences in supported configurations between the vendors, for example one might support a wider range of disk drives than the others one may announce support for SSD before the others. Any more detail is outside of the scope of this document.



USP V

XP24000

The product names used throughout this document are trademarks of Hitachi, HDS, HP and Sun Microsystems.

2. USP V High Level Overview

The USP V is what is termed an External RAID Array. This “external” implies that it exists physically outside of the server, i.e. the disks are not physically located within on the end of a serial SCSI cable inside of a server. The “RAID Array” implies that it is an advanced and so-called intelligent disk subsystem and not just a bunch of disks (JBOD) hung together with no associated intelligence. And of course RAID implies that it supports many levels of RAID protection to both protect the data it stores as well as affect the performance thereof.

Because it is “external” it can have multiple servers attach to it concurrently. The USP V supports modern servers and operating systems attached to it including Mainframe systems. In fact it has its roots in Mainframe and has only more recently been extended to Open Systems hosts. However, this manual deals only with the Open Systems side of the array.

The basic concept of the USP V is to provide servers and applications with persistent storage. The USP V itself can hold more than 1,000 disks (1,152 to be exact) in a fully configured 5 frame system. These disks are hardware RAID protected and carved up into logical devices (LDEVs). These LDEVs can be any user configurable size within reason. Current limits are from as little as 40 or so MB up to 4TB for a single LDEV, but considerably larger using array based functions to concatenate multiple LDEVs that appear to hosts as a single volume. These LDEVs are then presented out to servers as SCSI Logical Units, often referred to as LUs or LUNs. To the attached servers, these LUNs look just like internal disks and can be used as such. For example these LUNs can be mounted as standard Unix and Windows mount points and have user data stored to them exactly like normal internal disk drives.

Servers connect to the USP V via Fibre Channel Ports often referred to simply as front end ports. Servers can connect directly to the front end ports on the USP V or via SAN switches.

The USP V has a large global DRAM cache and is also often referred to as “cache centric”. The global cache is the real heart of the USP V and is where the good stuff happens. At the time of writing, the USP V can be configured with 512GB of mirrored cache. This allows to USP V to operate at near the speeds of the DRAM cache rather than the disks on the back end. The USP V is extremely fast.

Behind the cache are disk adapters, referred to commonly as Back End Directors or BEDs. These BEDs control, among other things, the physical disks. They also provide hardware RAID protection.

Behind the BEDs are the disks. May be “Drives” is a more appropriate term since the recent introduction of SSD Enterprise Flash Drives. This is the ultimate destination of data written to the USP.

<simple diagram of host, SAN Switch, FED, cache and disk needed>

Layered on top of all of that are many additional services provided by the USP V. These can be broken down in to the following three broad categories –

- Copy Services – Disk-to-disk backups, and Copy-On-Write snapshots....
- Replication Services – Synchronous and asynchronous distance replication products...
- Value-Add services – Storage Virtualisation, Dynamic Provisioning, Thin Provisioning...

In a nutshell, the USP V is a top of the range enterprise class external RAID array that is considered by many to be best of breed.

HARDWARE SECTION

DRAFT

3. Major Hardware Components

3.1 Hardware High Level Basics

<diagram needed identifying major components>

As data travels from the host to the USP V, the first component of the USP V it meets is a Front End Port., sometimes referred to as a channel port (Mainframe roots). The front end ports on the USP V can have long wave or short wave SFP transceivers, with short wave being shipped as standard.

From the front end ports, the data then travels the over the Hi-Star network and in to cache memory. Cache memory is mirrored for redundancy, with one half in Cluster 1 and the other half in Cluster 2 (Cluster 1 and 2 are essentially separate power domains but are explained in more detail later). If the data is write data, the operation is acknowledged back to the host as completed once the data is committed in cache.

All write data is duplexed to cache meaning that it is mirrored to cache in cluster A and Cluster B with a single write operation.

From cache the data is lazily destaged to disk according to the USP Vs internal caching algorithms. As data leaves cache and travels towards the disks it again travels over the Hi-Star network to the Back end Directors.

The Hi-Star network is a non-blocking crossbar switch.

The Back End Directors then send the data on its way to the disks via the switched FC-AL back-end loops.

Once data is committed to disk, the copy in cache is moved from the write cache into an area of read cache so that if it is referenced again it can be quickly retrieved from cache. Eventually it expires from cache and must be fetched from disk when requested.

3.2.1 Front End Directors

Front End Directors (FEDs) are installed in pairs, with one board being installed to Cluster 1 and the other to Cluster 2. If a FED is described as being an 8 port *feature*, this means 4 ports per FED board and 8 ports across the pair.

The front end ports on the USP V are fibre channel ports equipped with either shortwave or longwave SFP transceivers and can operate at 1Gbps, 2Gbps or 4Gbps and can auto-negotiate. They can also operate in a variety of topologies including FC-AL and Fabric Point-to-Point. Fabric point-to-point being the standard topology for attachment to Fibre Channel SAN switches.

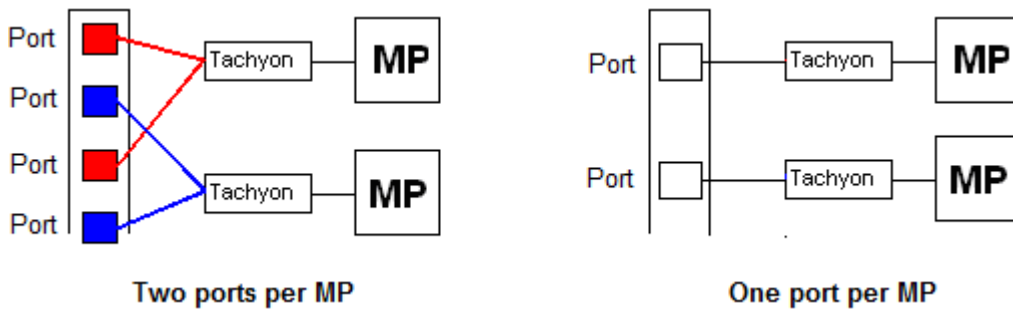
Individual transceivers (ports) as well as the FED boards themselves are hot pluggable (by a qualified engineer and NOT by a user or storage administrator).

Each front end port has its own WWPN that is based on the serial number of the array and the port identifier. Therefore replacing a port or even a FED does not change the associated WWPNs of the ports.

- The last two digits of the WWPN relate to the port number.
- The 4 digits prior to that are a hex representation of the serial number of the array.

Microprocessors

Behind each port is a microprocessor (MP). Depending on the model of FED installed, each front end port either has its own dedicated MP or shares an MP with one other port. Obviously the FEDs with a 1:1 ratio of ports to MPs are higher performing.



<this is an old drawing I put together in MS paint ages ago. If you have a better diagram please let me know>

For performance and redundancy, two ports sharing the same MP should be considered the same port. For example, the two red ports in the diagram above share the same MP. Therefore you would never put two paths from the same host on them as failure of the MP would render both ports "down".

The MPs on the FEDs run a portion of the microcode (DKCMain) and are responsible for all IO until it is completed to disk. They also run the code necessary for copy or replication programs such as HDP and TrueCopy. These functions all cause overhead for the MPs.

For example, a front end port with HDP LUNs that are being TrueCopied, ShadowImaged, Copy-On-Writes, LUSED and on Concatenated Array Groups will push its associated MP harder than a front end port that has non-HDP LUNs that are not replicated or copied etc.

Port Modes

Each port can be configured in any of the following modes -

- Target – Standard port that hosts connect to. Has LUNs presented on it.
- Initiator – Used for TrueCopy and HUR. Sends data to a remote USP V
- MCU Target – Used for TrueCopy and HUR. Receives data from a remote USP V
- External – Used to connect a USP V to an external storage subsystem (emulates a Windows initiator).

When using FEDs with two front end ports per MP, setting a port to one of the above modes forces its buddy port on to the same mode (buddy ports are ports that share the same MP). Basically the MP can only support one mode for both ports.

Access to Cache

Each FED and BED (Back End Director) is connected to Cache via multiple paths to the Cache Switches (CSW). These paths operate at 1064MB/sec. The CSWs themselves are fully non-blocking crossbar switches with multiple paths to the FEDs BEDs and Cache.

Each MP actually has two ASICs. One is responsible for passing data to the CSWs and the other is responsible for sending control data to Shared Memory via the dedicated point-to-point Shared Memory network (non switched). This architecture allows for separation of user data and control data making for better throughput and reducing bandwidth and other resource contention between user data and control data.

<diagram needed>

All MPs, on FEDs and BEDs can access both sides of cache and shared memory. So if cache A is lost, all FED and BED ports can still see cache B.

LUN Mapping

Sometimes people, or the documentation, may state that you can attach up to 2048 hosts per port. Although theoretically this is true, each host could only have a single LUN with a queue depth of 1 without risking overrunning the target queue depth. Basically each port has a supported queue depth of 2048. The same would also hold for any claims that you can present 2048 LUNs per port. Theoretically this may be true, however, each LUN could only have a single outstanding IO without overrunning the port.

2048 is the queue depth for the port. Although the port has a queue depth of 2048 each LUN has its own max queue depth of 32.

The FC/SCSI protocol specification only allows for a single byte for a LUN number you may expect a limitation of 256 LUNs per port. However, this is overcome by implementing virtual buses on each port.

When internal LDEVs are mapped to front end ports they are referred to as LUNs

Host Storage Domains

Each front end port can be divided into multiple secure virtual ports called Host Storage Domains (HSD). This allows each port to have multiple hosts simultaneously and securely connected via a SAN switch. It is not uncommon to see between 10 or 20 hosts connected to a single port, each with its own secure Host Storage Domain (performance must be taken in to account when deciding how many hosts access a single port).

Access to LUNs presented to an HSD is managed via WWPNs with only the hosts who's WWPNs are configured for the HSD being able to access the LUNs in the particular HSD.

Each secure HSD has its own LUN address space allowing each to have its own LUN 0.

<diagram of multiple HSDs per port needed>

Each HSD has a mode that is set according to the Operating System connecting to the HSD. For example there is a Host Mode for Windows, and another for HP-UX and another for AIX etc. These modes ensure that the correct SCSI code pages and OS specific requirements are adhered to. In addition to Host Modes, certain Host Mode Options can be configured via the SVP if specific non standard behaviour is required.

If Mainframe hosts are connected to the USP V, it is the CHA Adapters (FEDs) where the CKD<->FBA conversion takes place. This is required because all disks and cache slots in the USP V are SCSI FBA (Open Systems style fixed block architecture).

3.2.2 Cache Switch (CSW)

<Please contact me if you would like to write this section>

3.2.3 Cache

It is fair to say that the cache subsystem (Cache Memory and Shared Memory) is the real beating heart of the USP V and is where the magic happens. Not only does it allow the USP V to operate extremely fast, it is also where all of the functionality such as HDP, ShadowImage..... Parity Generation and the likes happens.

None-volatile, Duplexed and Global

All cache memory in the USP V is DRAM and is non-volatile as it is protected by the USP Vs battery backup system. For this reason, the cache is sometimes referred to as NVRAM. In a vanilla install all cache is global, meaning that it is available to all LUNs.

Cache memory is installed in parallel in to the USP V in to both Cluster 1 and Cluster 2. Cache A is installed to Cluster 1 and Cache B is installed to Cluster 2. All data written to cache is duplexed to Cache A and Cache B. Cache A must be equal to Cache B. Cluster 1 and Cluster 2 are on separate power subsystems so that losing power to one Cluster will only render one half of the mirrored cache “down” (no data will be lost).

Under normal operating circumstances (not operating in CFW-Inhibited mode), all writes to the USP V are treated as Cache Fast Writes. This means that once the data is duplexed to both sides of cache, an acknowledgement is issued to the host, and so far as the host is concerned, the operation is considered complete.

All writes to cache are duplexed for redundancy. This means that each IO is written to Cache A and Cache B simultaneously with a single write operation. This behaviour is different to many mirrored cache implementations – where data is written to one side of cache and then a separate write/copy operation is required to mirror the data to the other side of cache, often over a dedicated gigabit Ethernet connection. The USP V implementation of write *duplexing* gives better performance.

In a USP V with no cache partitioning configured (covered later), all cache memory is available to all LUNs and is dynamically managed by the microcode.

Cache cannot be avoided

Cache Memory (CM) is an integral part of the I/O path on the USP V. This means that read and write data is always written to cache. There is no way to bypass cache!

All data transferred from the FEDs to the BEDs must go through cache. Again, cache cannot be avoided. Even ShadowImage jobs (controlled by the BEDs) that copy data from one LDEV to another LDEV over the same loops controlled by the same BEDs on the same Array Groups must go through cache (not that you would ever put a P-VOL and S-VOL on the same Array Group....).

<diagram needed>

Read Cache and Write Cache

Theoretically 100% of cache is available for read operations, although this will only ever be the case if there is absolutely zero write activity. Not likely.

Read data is not duplexed into both sides of cache due to the fact that the original data already exists on disk and can be safely retrieved from disk if one side of cache is lost. This effectively increases the size of the read cache. The USP V uses a simple calculation to determine which side of cache the Read data will be staged to, this hopefully balances the use of the Read cache over both Cache A and Cache B.

All writes to the USP V are duplexed to both sides of cache.

For write operations, the maximum amount of available cache is as follows – Assume your USP V has 512GB cache, this is 256GB in Cache A and 256GB in Cache B. However, the amount of cache available for writes is 70%, giving ~358GB for writes. This 358GB is then duplexed (halved) giving ~179GB of cache for write data.

The above percentages are not user configurable.

Cache Fast Writes and Destaging

An acknowledgement is issued to the host once data is duplexed to cache. The USP V keeps the data in cache for as long as possible hoping that the data will still be in cache the next time it is referenced. If data that a host is requesting is already in cache the USP V can respond to the host far quicker than if the data has to be fetched from disk.

There are several cache watermarks that determine the rate at which data is destaged from cache to disk. These watermarks are not user configurable.

Destaging of data from cache to disk usually starts when cache gets to around 25-30% occupancy. This is not aggressive destaging.

If cache occupancy reaches 40% then the USP V will speed up the de-staging process. Some applications may notice a slight performance hit when this happens.

At 70%¹ occupancy the USP V goes in to Emergency Destage mode and returns a BUSY to any host trying to write while there are no cache slots available. At this point your applications are guaranteed to notice a performance drop. Under Emergency Destage mode the host is still acknowledged with an end of operation when the data is written to cache. However, writes may be delayed while cache slots are freed up.

If one side of cache is lost, or duplexing is threatened for any reason, the other copy is immediately destaged to disk and the subsystem will operate degraded in Cache Fast Write Inhibited mode (CFW-inhibited) mode until the problem is resolved and cache memory is once again duplexed. CFW-Inhibited mode is where the end of operation acknowledgement is not sent to the host until the IO is committed to disk. This is very slow in comparison to acknowledging when data reaches cache, but is required to ensure that user data is not lost.

Under normal operating circumstances, once data is destaged to disk it is not immediately removed from cache, instead it is switched from a Physical Dirty queue to a clean queue in an area of Read cache. This increases the chance of write hits in future.

¹ This is actually 100% of write cache as 30% is always maintained for read data.

Pre-fetching

The USP V employs read-ahead techniques (pre-fetch) to increase read throughput and also increase the likelihood of cache hits for future requests for data local to the originally requested blocks.

When a cache miss condition occurs, the BEDs are instructed to fetch the required data from disk. The requested data is read from disk, as well as an entire track immediately following the target data. This is done because often a host will request more data immediately before or after the data just referenced. If the USP V has already read the remainder of the track to cache it will already have that data in cache when the host requests it. This is known as referential locality.

In order to determine if the requested data is in cache (hit) or not (miss), the cache directory is searched. The cache directory exists in Shared Memory and is searched by the FEDs each time a read or write operation happens. The cache directory is organised into a hierarchical set of tables to decrease the amount of time required to determine a hit or miss condition.

Cache Partitioning

It is not possible to manually tune or alter the cache slot or segment size on the USP V. The only way to manage cache is by partitioning the cache (Cache Partitioning Manager) or by pinning a LUN in cache (Dynamic Cache Residency Manager).

Cache Partitioning allows you to divide the cache up into somewhat independent areas referred to Cache Logical Partitions (CLPRs). CLPR is pronounced "clipper". This way you can guarantee an amount of cache dedicated to an application or set of applications.

However, even when cache is partitioned, one partition of cache being overrun can still have an adverse effect other partitions. For example, you have two cache partitions CLPR1 and CLP2. CLPR1 is overrun and at 70% occupancy and in Emergency Destage mode. When this condition occurs, aggressive destaging begins in **all** CLPRs. However, only hosts connected to LUNs in the affected CLPR will receive BUSY signals. The net result is that this is in some ways better than not having cache partitioning but not perfect.

Cache internals

Cache is divided in to slots and segments for efficient staging and destaging of data. These slots and segments are mapped in the cache directory in Shared Memory. The cache directory maps between what data is in cache, and its associated address in cache memory.

Cache slot and segment size on the USP V is fixed but based on LDEV geometry.

For OPEN-V volumes, cache slot size is 256K. This exactly matches OPEN-V track size.

For OPEN-V volumes each slot is made up of 4 segments, with each segment being 64K.

The minimum working cache unit size is the segment, and data is normally destaged in segments.

For random writes, the USP V will assign cache segments, but for sequential writes it will assign cache slots.

The above allows for efficient use of cache and optimum performance.

Several queues exist in cache memory and all data resident in cache resides in one queue or another. Anything more than this is more than likely Hitachi proprietary information.

The USP V attempts to coalesce write data before destaging to disk. This helps reduce the RAID5 penalty for random writes as well as allowing the USP V to batch up writes and arrange them to make best use of disk characteristics when writing to disk. (See section on RAID5 performance for details on write coalescing and the RAID5 random write penalty)

When destaging from cache to disk, the USP V writes two tracks per spindle before switching IO to the next spindle in the RAID set., This then gives us an effective chunk size, per spindle, of 512K (each track is 256K and two tracks per spindle = 512K “chunk”).

The formula for working out stripe size for all RAID configurations on the USP V is -

$$(256 * x) * y$$

Where x is always 2 (because it always writes 2 tracks at a time) and y is the number of data spindles in the RAID set.

Although the USP V supports Open Systems as well as Mainframe hosts, all data in cache and on disk is stored in SCSI Fixed Block Architecture (FBA) format. Conversion from CKD to FBA is performed by the Mainframe Channel ports (FED).

3.2.4 Shared Memory

Shared Memory is also DRAM volatile memory backed by the USP V battery subsystem, making it non-volatile. However, Shared Memory is entirely separate from normal cache memory and is used to store metadata rather than actual user data.

The USP V can be configured with 24GB of Shared Memory which in a fully configured systems is accessed over 256 paths (128 per cluster) all operating at 150MB/sec giving a peak bandwidth for control data of 38.4GB/sec (19.2GB/sec per cluster). Paths to Shared Memory are not shared with paths to Cache Memory over the Cache Switches, they are dedicated point-to-point paths. This helps ensure that USP V related control data etc does not contend for bandwidth etc with host/application data.

<diagram needed>

How Much Shared Memory Do I Need

The short and best answer to this is to consult your HDS or HP representative.

However, as a very loose and general rule you should install 20MB of Shared Memory for every 1GB of Cache Memory. However, adding additional Shared Memory to the USP V does not necessarily increase performance. In fact certain areas of SM are dedicated for certain program products such as COW and HDP and cannot be used to increase the number of DIFF tables for ShadowImage etc.

Some factors that influence the amount of Shared Memory required include, but are not limited to, the following –

- The number of Control Units and LDEVs
- Number of ShadowImage pairs
- Number of TrueCopy pairs

Metadata

Shared memory is absolutely vital and all internal USP V operations and communications (FEDs talking to BEDs, SIMs etc) must go through shared memory.

SM stores the following configuration data (and more) –

- Cache directory
- Configuration info CU info, LDEV info, available MPs..
- Device mapping (LUN → CU:LDEV → VDEV)
- LUN maps, status and configuration
- Are tracks replicated etc
- RAID information
- HDP Dynamic Mapping Tables etc

Because Shared Memory holds the Cache Directory, the front end MPs must query Shared Memory to determine if data is in cache (hit) or has to be fetched from disk (miss). It is therefore extremely important that the cache directory can be searched quickly and efficiently so that delays are not incurred due to “getting lost in cache” (i.e taking a long time to determine that the data is not in cache).

SM has 100% read and write duplication whereas CM only duplicates writes. If SM A fails then cache A is also unusable just like if the cache had also failed.

If cache memory is the heart of the USP V then Shared Memory is the brains.

3.2.5 Switched loop back end

All FC disks in the USP V are FC-AL. However, the USP V now employs a switched loop architecture on the back end. The protocol is still FC-AL (arbitrated loop) but the switches provide intelligent port bypass etc and bring many of the benefits of a switched network to the FC-AL world.

The fully switched FC-AL back end allows for far superior fault diagnosis and isolation. It also allows for faster access to disks – e.g. to reach the last disk in the loop you do not have to pass the circuitry of every other disk in the loop first, the switch will make a switched connection directly to the desired disk. As such, this reduces the amount of arbitration involved.

Although the switched backend improves access speeds the main gain is increased accessibility to disks and fault tolerance.

The backend now operates at 4Gbps but can negotiate to either 4Gbps or 2Gbps.

<diagram needed>

All disks on the same loop must operate at the same port speed which is based on the speed of the first disk in the loop– 2Gbps or 4Gbps.

3.2.6 Back End Directors

The Back End Directors are the hardware components that control access to the disks as well as staging data to and from the disks to cache. The back end, including BEDs, is now a fully switched 4Gbps FC-AL giving better access to disks at the end of the loop and better fault isolation.

Like FEDs, BEDs are installed in pairs, with each BED pair having 8 ports. Ports on the BEDs can operate at either 2Gbps or 4Gbps and all BEDs have a port to microprocessor (MP) ratio of 1:1.

Each BED port is an FC-AL port and connects to the FC-AL switched loop back end and ultimately in to the disk drives.

Each USP V can have a maximum of 8 BED pairs giving a total of 64 BED ports.

Each BED contains a parity generation co-processor referred to as a DRR chip (Data Recovery and Reconstruct) which performs all parity calculations.

Anything that falls under the control of a single BED pair is sometimes referred to as an Array Domain. So all disks that hang of a single BED pair are said to be in the same Array Domain.

<diagram needed>

The USP V allows you to intermix RAID levels, spindle size and rotational speed within the same Array Domain. However, all drives in the same Array Domain must operate at the same transfer rate on the port (2Gbps or 4Gbps). If you mix interface speeds they will all run at 2Gbps.

You can also intermix emulation types within an Array Domain although within a single Array Group, all LDEVs must have the same track geometry and format (i.e. all 3390 or all OPEN-x).

While the FEDs control TrueCopy, UVM, HDP and other software functionality, they do not control ShadowImage jobs. These are controlled by the BEDs.

The BEDs control access to and from disk, they also monitor loop, Array Group and processor utilisation.

In the USP V it takes two BED pairs to support every RAID configuration. This is because the new half sized BEDs only have 4 loops per PCB and therefore require two BED packages to support RAID 5 (7+1). RAID1 (4+4) can run off a single BED pair (single Array Domain) or across two.

<diagram needed>

3.2.7 Disks and Array Groups

The USP V can support a maximum of 1,152 x 3.5inch form factor drives installed internally in a fully configured 5 frame system.

All data stored on disks in the USP V is stored in Fixed Block Architecture format.

The USP V supports intermixing multiple disk types in the same USP V including Fibre Channel, SATA II and Solid State Drives/Enterprise Flash Drives.

There can be a maximum of 40 spare drives and minimum of 1. However, even if only a single spare is installed, 4 slots (Named Parity Group) are dedicated to spare drives and cannot take normal data drives. Optionally another 36 slots can be for either data or spares.

Although when installing spare drives, they must be installed to named Parity Group slots, installation does not have to conform to normal Parity Group restrictions. For example, you can install different size and RPM drives in a single named Parity Group location, so long as they are spares.

The USP V always has spare disks spinning (hot) and performs diagnostics against them to assist in keeping them healthy. Spares are global and can back up any drive that is the same speed (RPM) and the same or smaller size.

Disks are installed in four-disk packs referred to as Parity Groups. It is possible to configure 2 Parity Groups as an Array Group (8 disks single stripe). Each Parity Group/Array Group (hereafter referred to as Array Group) is RAID formatted to a single RAID level. The USP V supports the following hardware RAID levels –

- RAID1 – as 2+2 or 4+4
- RAID5 – as 3+1 or 7+1
- RAID6 – as 6+2

Each Array Group is carved in to LDEVs of a user configurable size. Each LDEV is comprised of adjacent stripes within a single Array Group. All LDEVs carved from a single Array Group will have a common RAID level (that of the underlying Array Group) and must also have a common emulation/geometry – either OPEN=x or 3390-x.

Command Tag Queueing

The USP V supports SCSI Command Tag Queuing (CTQ) allowing the subsystem to optimize head seek time etc. This is done by allowing the host to issue multiple commands without having to serialise them. The subsystem does not have to acknowledge each I/O sequentially, instead it is allowed to process requests in the order that is most efficient for spindle usage and head movement.

Track alignment

Track alignment is not considered essential on the USP V although it may still be a good idea to align partitions on a track boundary to help avoid disk and track overlaps as well as cache segment overlaps. The more sectors per track, the less of an improvement you will see, as less IOs will fall on a track boundary. For OPEN-V volumes the USP V has 512 sectors per track (256K).

Concatenated Array Groups

The USP V also supports the ill-named Concatenated Array Groups. This is available for two or four RAID5 7D+1P Array Groups allowing you to configure essentially RAID5 14+2 or RAID5 28+4 – this gives what is often referred to as “wider striping” where each stripe will touch more back end disks. If your Array Groups are considered the bottleneck this is a good option and is almost standard practice on most USP V installations where RAID5 is deployed.

<diagram needed>

These Concatenated Array Groups are still built from standard RAID5 (7+1) Array Groups. This means that each underlying RAID5 (7+1) Array Group has its own dedicated parity stripe and can therefore only suffer a single failed spindle. The concatenated Array Group can tolerate 4 failed spindles, so long as they occur in discrete Array Groups.

Quick Format

All LDEVs cut on a USP V are RAID formatted (they have zeros written them – often referred to as being zeroed out). This can be a time consuming operation as the writing of zeros is performed as a low priority background operation so as to not impact host IO. Creating a lot of new LDEVs on a USP V can often take many hours.

The Quick Format operation allows volumes (LDEVs) to be made available to hosts immediately after the Quick Format operation starts even though the volumes are not yet fully formatted (the format operation is running in the background). This feature requires the configuration of a “System Disk” within the USP V to store the control table for the Quick Format operation in case power is lost.

Systems disks need a minimum of 46MB free space to support Quick format.

If a host writes to an area of the volume being quick formatted and that area has not yet been formatted, performance may degrade (if it is not absorbed by cache).

4 Logical data flow

<diagram of data entering the USP V, hitting FED, CSW, CM, CSW, DKA, FSW LOOPS, DISKS>

Brief numbered walkthrough of the above diagram

<Please contact me if you would like to write this section>

5. The LUN and how it is made up

Basic idea..... show how a classical (non HDP) LUN is created – Physical disks installed as a Parity Group/Array Group with RAID level and VDEV(s) then cut into LDEVs and mapped in to CU:LDEV space. Then added to front end port and HSD as LUN.....

<Please contact me if you would like to write this section>

DRAFT

SOFTWARE SECTION

6 Microcode

<Need Major minor version nubers>

The main microprogram is DKCMain.

A portion of DKCMain runs on the FED ports a portion runs on the BED ports.....

<Please contact me if you would like to write this section>

7 Hitachi Dynamic Provisioning

Hitachi Dynamic Provisioning is a new and much improved way to manage capacity, LUNs and provisioning. It is based on the concept of pooling LDEVs and Array Groups together into constructs referred to as Pools. From these Pools virtual volumes are then created and presented to hosts as standard LUNs. However, these virtual volumes do not consume any space until the host writes to them. Even then, space is consumed relative to how much data the host writes.

At a high level, HDP provides the following 5 major benefits –

1. Simplified LUN management (quicker and less thought intensive provisioning)
2. Thin Provisioning (allocate on demand)
3. Wider back end striping
4. Zero Page Reclaim
5. Dynamically grow LDEVs

HDP Theory

An HDP **Pool** is created from multiple LDEVs referred to as **Pool-VOLs**.

Once a Pool is created, its space is divided into 42MB extents referred to as **pages**. This **page** is the basic unit of allocation, and is fixed at 42MB. For more detail regarding the 42MB Page size Appendix D: Theb42MB Page.

When an HDP Pool is first created, all of its pages are **free**, as they are not allocated (loaned) to any DP-VOLs.

DP-VOLs are virtual volumes created and **associated** with a single HDP Pool but mapped into the CU:LDEV address space and then to front end ports just like normal LDEVs. When a DP-VOL is initially created it is allocated (loaned) no pages. Therefore it consumes no capacity.

Pages are allocated to DP-VOLs on demand as hosts write to them.

The **Free Page List** is an HDP metadata related table that is referenced when choosing which page to allocate to a DP-VOL. When an HDP Pool is first created, all Pool-VOLs used to create the Pool have their pages grouped together in a concatenation type configuration as illustrated below –

Free Page List - Pre Initialise



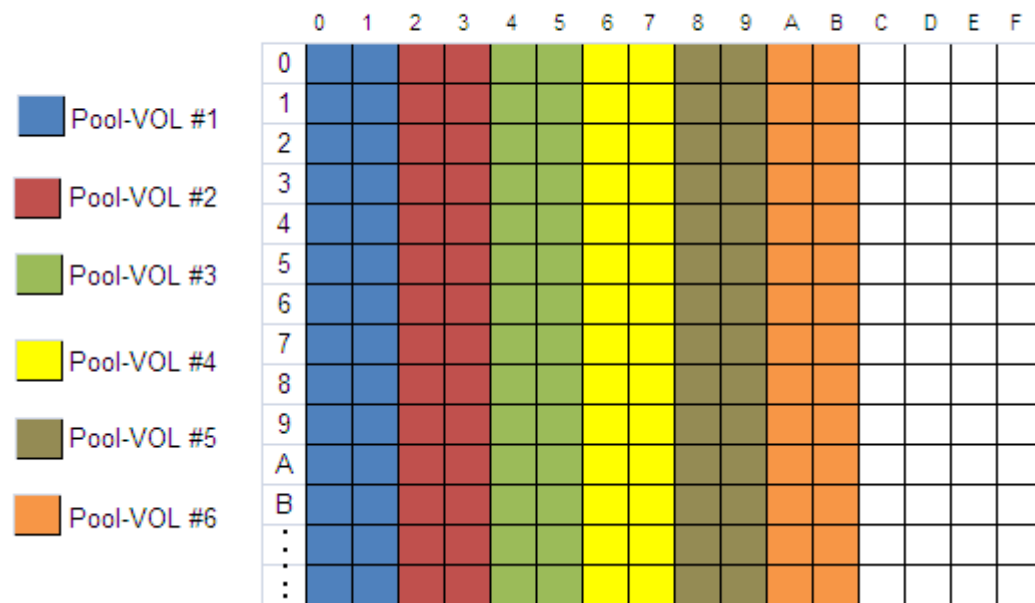
Pages are allocated from the Free Page list in series on a first-come-first-served basis starting at location 0:0 and then moving on to 0:1, 0:2, 0:3..... 1:0, 1:1, 1:2.....

With the Free Page List in its pre-initialised status, all pages Pool-VOL #1 will be allocated to DP-VOLs prior to any pages on Pool-VOL #2 being allocated etc.

Prior to allocating pages from a Pool (making the Pool production ready) it is highly recommended to **Initialise** the Pool by clicking the **Initialize** button.

Performing the Initialise operation re-arranges the pages in the Free Page List in more of a striping formation –

Free Page List - Post Initialise



After the Free Page List is initialised, pages will be allocated more evenly over the Pool-VOLs in the Pool, similar to a stripe. Assuming the example above, two consecutive pages from Pool-VOL #1 will be allocated before switching to Pool-VOL #2.....

NOTE: *The Free Page List is an HDP construct that is referenced in HDS documentation. However, the author is not familiar with its actual structure. The diagrams above are for illustrative purposes only and should not be taken as actual or authoritative. Knowledge of the internal structure of the Free Page List may assist in Pool construction decisions, but the author is not in possession of such information.*

Zero Page Reclaim Theory

As of microcode 60-04-04 and with the associated upgraded version of Storage Navigator, the USP V supports Zero Page Reclaim operations on a per DP-VOL basis. The Storage Navigator GUI refers to this as a Discard Zero Data operation.

When you select to perform a Zero Page Reclaim operation on a DP-VOL, the USP V will search the Pages allocated (loaned) to that DP-VOL, looking for Pages that have no data. Or to be more correct, it searches for pages that contain only zeros.

Sidestep: Most storage arrays will zero out a volume when it is RAID formatted, basically writing zeros to all blocks comprising the volume. This helps the XOR parity calculation.

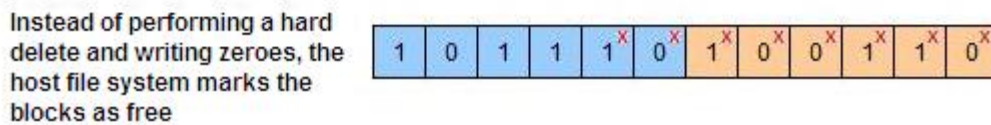
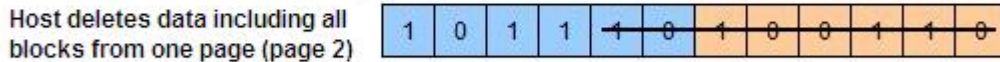
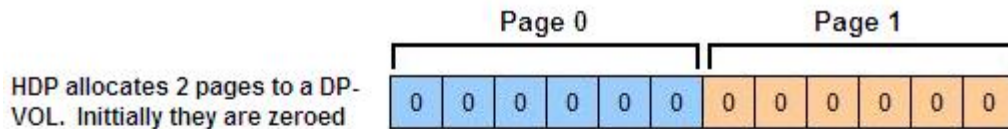
So any page (42MB worth of consecutive stripes in the underlying Pool-VOL) that the array finds comprised entirely of zeros, it assumes is unused and breaks the association between the page and the DP-VOL. This has the effect of placing such pages back into the Free Pool and freeing up capacity that can be used by other DP-VOLs. This has obvious capacity saving benefits and is a useful tool for any Thin Provisioning array to have up its sleeve.

How useful this functionality is relies on a few factors. Two of which include; page size, and file system behaviour

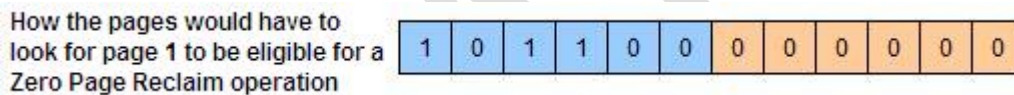
Page size. It would seem, at first glance, that the smaller the page size, the more opportunity for reclaiming space. The theory behind this being that there is more chance of finding pages comprised entirely of zeros if your page is relatively small. Conversely you are less likely to find multiple megabytes worth of consecutive zeros, as would be required if you have a larger extent size. However, this may have little effect in practice, especially after migrations from thick to thin.

File System Behaviour. One point in relation to file systems is that many file systems do not zero out space when a file is deleted. Instead they take the lazy approach and simply mark the blocks as free but leave the original data in place.

While this soft delete behaviour may prove useful should you need to recover a deleted file, it is not friendly to Zero Page Reclaim operations. Put in other words, freeing up 150GB of space on a volume does not normally (depends on your file system behaviour....) write zeros and make the space eligible for Zero Page reclaim. See diagram below -



The above diagram is an oversimplification but illustrates the point. For Extent 1 to be a candidate for Zero Page Reclaim, the file system would have to hard delete (zero out) all deleted data as shown in the diagram below -



With present day file systems, the best use case for Zero Page Reclaim may be after migrating volumes from classical thick volumes to new thin dynamic volumes. For example, this will work well with a 500GB traditional thick volume where only 200GB has been written to. When migrating this to a thin volume you will more than likely be able to reclaim the untouched 300GB at the end of the volume to the Free Pool.

It is also possible to use software tools to zero out deleted files. Examples of such tools are secure erase tools.

For HDP Best Practice information see <http://blogs.rupturedmonkey.com/?p=274> for the latest version of the HDP Unofficial Best Practice guide.

8 UVM

<Please contact me if you would like to write this section>

DRAFT

9 ShadowImage

<Please contact me if you would like to write this section>

DRAFT

A Hitachi Universal Replicator

<Please contact me if you would like to write this section>

DRAFT

B TrueCopy

<Please contact me if you would like to write this section>

DRAFT

C Copy-On-Write

<Please contact me if you would like to write this section>

DRAFT

D Additional suggested chapters

<Please contact me if you would like to write this section>

E Additional suggested chapters

<Please contact me if you would like to write this section>

Appendix A: Contributing Authors

Principal author: Nigel Poulton, <http://blogs.rupturedmonkey.com> ,
nigel@rupturedmonkey.com, <http://twitter.com/nigelpoulton>

Contributing author (RAID 6 content): Devang Panchigar - <http://storageneve.com>

Appendix B: RAID configurations

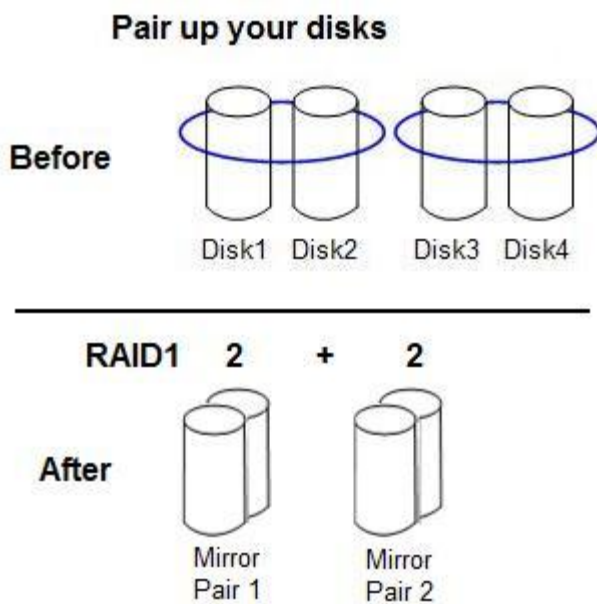
RAID1

The basic principle behind RAID1 is that of mirroring. In its most basic form, RAID1 takes two separate drives and bonds them together as a mirror. One drive is an exact mirror copy of the other. This enables a RAID1 configuration to tolerate the loss/failure of a single drive without losing data. However, it has a capacity overhead of 50%. This is because the entire capacity of one of the disks is required to implement redundancy.

USP V RAID1 implementation details

For RAID1 the USP V supports either RAID1 (2+2) or RAID1 (4+4)

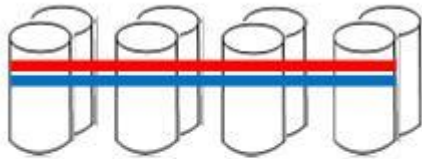
RAID1 (2+2) utilises two pairs of mirrored disks. Assume you have four disks; Disk1, Disk2, Disk3 and Disk4. You take Disk1 and Disk2 and configure them as a mirror. You then do the same with Disk3 and Disk4. This gives you two mirrored pairs. See diagram below -



Data is then striped across the two mirrored pairs.

RAID1 (4+4) is similar to RAID1 (2+2). It is a single stripe across 4 mirrored pairs of disks. Each full stripe write to disk touches all 4 mirrored pairs (all 8 physical spindles) disks. See diagram below -

RAID10 (4+4)



— Stripe 1. Touches all spindles in RAID set. Stripe size = 2048K

— Stripe 2. Touches all spindles in RAID set. Stripe size = 2048K

When the USP V does RAID1 it is actually doing RAID10, sometimes referred to as RAID1+0.

RAID10 is exactly what it says on the tin - RAID1 and RAID0

To break it down -

RAID 1 = Mirroring

RAID 0 = Striping

So RAID10 is a combination of mirroring and striping, in that order. First you create your mirrored pairs (that's the RAID1), and then you stripe across them (that's the RAID0). 1 + 0. Simple.

This is more resilient than 0+1, where you first create your stripes and then mirror them. If your RAID1 (4+4) was RAID0+1 you would create a stripe using your first 4 disks, and then create a second stripe on your remaining 4 disks. Once those two stripe sets were created you would configure one as a mirror of the other.

However, in a RAID0+1 configuration the failure of a single disk in either stripe will break the mirror! Not good! The USP V does RAID10 and does not allow for configuration of RAID0+1.

RAID5

<Please contact me if you would like to write this section>

RAID5 Concatenation

<Please contact me if you would like to write this section>

DRAFT

RAID6

The bulk of the below data on USP V RAID6 is based on the following article by Devang Pachigar (the article also contains links to other vendor implementations of RAID6) –

<http://storagenerve.com/2009/02/09/hitachis-hds-raid-6/>

The USP V allows Array Groups to be configured as RAID6 (6D+2P).

Below is an abstract about RAID 6

Technology: Striping Data with Double Parity. Two parity writes per row with parity distributed.

Performance: Medium

Capacity overhead: 25% (with additional drives you can bring down the capacity overhead, although the USP V does not allow for configurations other than 6D+2P).

Performance overhead: ~33%

Array Group configuration: 6 data and 2 parity stripes.

Risk of Data Loss: Low. Can survive double disk failures without losing data.

Advantages: Extremely low probability of data loss

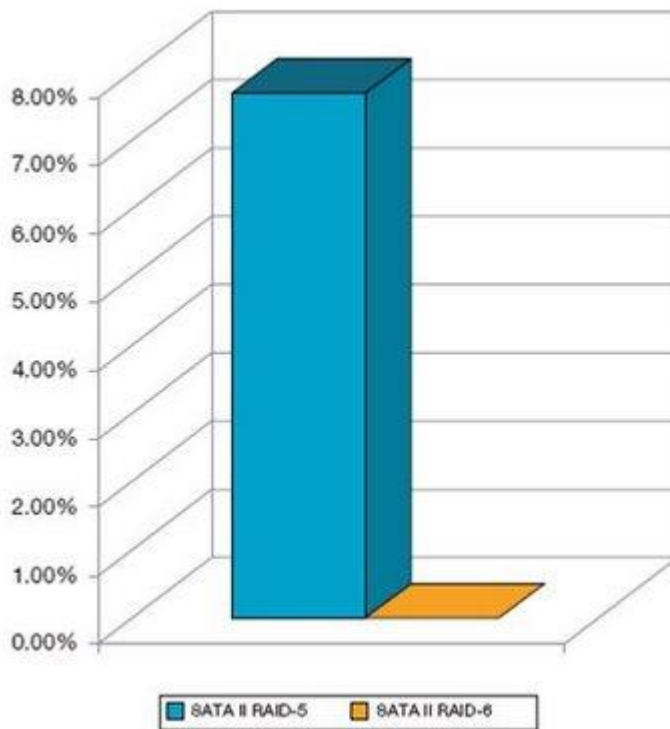
Disadvantages: Poor random write performance. Additional capacity overhead due to requiring N+2 drives to implement because of two-dimensional parity scheme.

RAID 6 is essentially an extension of RAID 5 which allows for additional fault tolerance by using a second independent distributed parity scheme (two-dimensional parity). Data is striped on a block level across a set of drives, just like in RAID 5, and a second parity stripe is calculated and written across all the drives.

RAID 6 provides for an extremely high data fault tolerance and can sustain multiple (up to 2) simultaneous drive failures which typically makes it a perfect solution for mission critical applications that cannot tolerate data loss but do not require high performance.

Hitachi does not recommend using RAID 6 for high performance applications where extreme random writes are being performed. In these cases, the use of RAID 1 (RAID 1+0) or Concatenated RAID5 (7+1) Array Groups is essential.

Probability of Data Loss with RAID 5 and RAID 6



As you see in the above graph, the probability or the percentage of exposure related to RAID 5 double disk failures is as much as 7.5% while the chance of triple failure in a RAID 6 configuration is ~0%. As drive sizes are increasing, the usage of RAID 6 will become more prominent.

USP V RAID6 implementation details



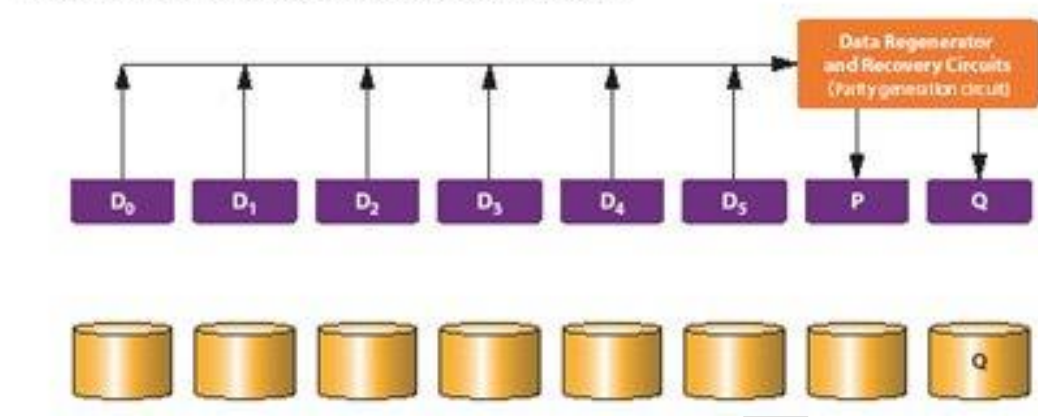
The above example shows an 8 disk Array Group in a USP V configured as RAID6 (6+2).

D1, D2, D3, D4, D5, D6 represent DATA BLOCKS and P1 and P2 the Dual Parity stripes.

The data blocks are followed by the parity and then the last parity drive is where the new data blocks starts to write again. With this sequential nature, the vast improvement is seen in the performance of this technology.

Lets introduce some mathematical formulas with implementation of RAID 6.

Figure 1. Formula for Generating Parity Data in RAID-6



In the above, D₀, D₁, D₂, D₃, D₄ and D₅ are the Data Blocks (Stripes) and P = Calibration data and Q = Secondary Parity

Using mathematical formula's with the Data Stripes (D₀, D₁, D₂, D₃, D₄ and D₅) and XOR (Exclusive OR), the P (Calibration data) is generated.

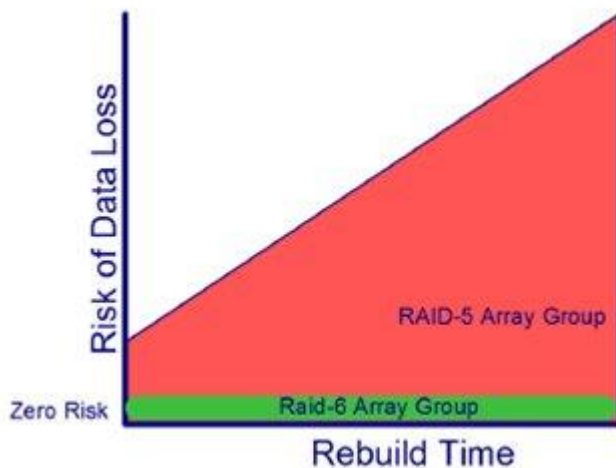
$$P = D_0 \text{ XOR } D_1 \text{ XOR } D_2 \text{ XOR } D_3 \text{ XOR } D_4 \text{ XOR } D_5$$

Q is the product of Coefficient and Data Stripes (D₀ through D₅) XOR

$$Q = A_0 * D_0 \text{ XOR } A_0 * D_1 \text{ XOR } A_0 * D_2 \text{ XOR } A_0 * D_3 \text{ XOR } A_0 * D_4 \text{ XOR } A_0 * D_5$$

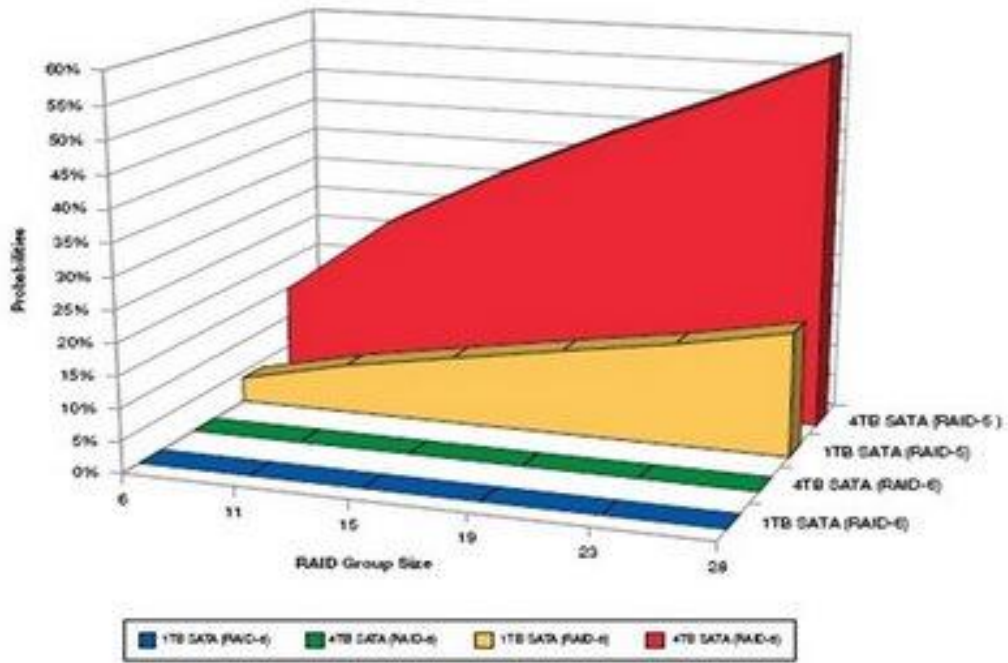
Typically with one drive failure the P (Calibration Data) is used to Generate or rebuild the new drive, with two drive failures, the P and Q data is used to rebuild the new drive.

The chart below shows the risk of data loss associated with RAID 5 and RAID 6



As times elapse with the drive failure on RAID 5 (with time to respond and rebuild times), the risk associated tends to increase. With RAID 6 and a drive failure the risk associated tends to stay the same and at ~0%.

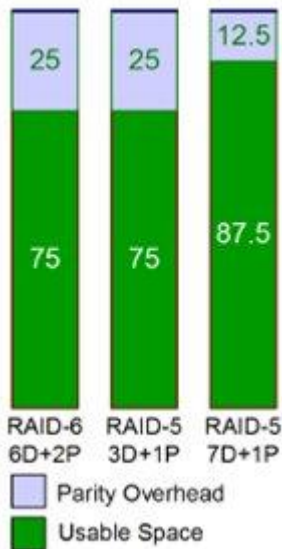
Based on different Raid Group Size, here is the risk of data loss with rebuild times.



As you can see in both of the above graphs, the risk associated with RAID 6 is pretty much zero percent.

Capacity Overhead

As discussed earlier, there is an additional capacity overhead with usage of RAID 6 vs usage of RAID 5. However, the risk associated with RAID 5 is much higher than the capacity overhead by RAID 6. Below is a graph that shows the capacity overhead associated with RAID6 (6+2) compared to the two USP V RAID5 implementations.



As can be seen in the above graph, the capacity overhead with 6 Data drives and 2 Parity drives is only 25%. If you were running Mirroring or some other variation of RAID 5, the overhead can be between 25% to 50%.

From a performance standpoint, the performance of RAID 5 and RAID 6 is very similar when comparing; Random Read, Sequential Read and Sequential Write workloads. However, there is added penalty with Random Write workloads. This is because of the two dimensional parity. The random write performance penalty associated with RAID5 is amplified with RAID6 because of the second parity write per row –

Because pre-update data and pre-update parity must be read and new parity calculated, this adds a significant performance penalty to RAID5 which equates to 4 I/Os per random write as shown below –

- 1 read of original data
- 1 read of original parity
- 1 write of new data
- 1 calculation and write of new parity

This becomes 6 I/Os for RAID6 because of the requirement for reading and writing of two sets of parity (P and Q) to each row, as shown below

- 1 read of original data
- 1 read of original parity P
- 1 read of original parity Q
- 1 write of new data
- 1 calculation and write of new parity P
- 1 calculation and write of new parity Q

RAID 6 on the USP V takes ~33% performance hit for Random Write workloads when compared to RAID1 2+2 and RAID5 7+1. However, this is only realised if the cache cannot absorb the performance hit and if it cannot coalesce writes.

RAID6 Summary

RAID6 is not recommended for high performance applications. It is the RAID of choice when mitigating failures is top priority and is most often deployed when large, especially SATA, disks are used. RAID6 can tolerate two simultaneous failures without losing data.

Appendix C: HDS and HP terminology cross-reference

<Please contact me if you would like to write this section>

DRAFT

Appendix D: The 42MB Page

The USP V supports the following hardware RAID levels –

- RAID10 2+2
- RAID10 4+4
- RAID5 3+1
- RAID5 7+1
- RAID6 6+2

In addition to the above, the USP V also supports the ill-named but quite impressive Concatenated Array Group (it is actually a stripe). Concatenated Array Groups allow you to join 2 or 4 x RAID5 7+1 Array Groups to get wider back-end striping.

So if we add these to the above list, we have an additional two RAID levels to consider -

- RAID5 14+2
- RAID5 28+4

As we can see, the USP-V gives us all three of the popular RAID levels (1, 5 and 6) as two options for Concatenated Array Groups, giving a total of 7 possible RAID configurations.

To make the following clear, we will use the SNIA definition of “stripe size” as a baseline –

“A parity RAID array's stripe size is the stripe depth multiplied by the number of member extents less the number of parity extents”

Stripe depth on the USP V for OPEN-V volumes is 512K (each track is 256K and the USP V writes two tracks per spindle before moving on).

Member extent is another way of saying member disks*.

Parity extent is another way of saying parity disks*

So to work out stripe size the calculation is -

$$\text{Stripe depth} \times (\text{member extents} - \text{parity extents}) = \text{stripe size}$$

As an example the calculation for RAID5 (7+1) will be as follows –

$$512 \times (7-1) = 3584\text{K}$$

Side note: The following diagram shows how to read member extents and parity extents from standard RAID notation –

Member extents
Parity extents
RAID5 (7+1)

Based on the above formula, the respective "stripe size" of each of the above RAID configurations is as follows -

- RAID10 2+2 = 1024K
- RAID10 4+4 = 2048K
- RAID5 3+1 = 1536K
- RAID5 7+1 = 3584K
- RAID5 6+2 = 3072K
- RAID5 14+2 = 7168K
- RAID5 28+4 = 14336K

Next consider that 42MB = 43008K.

42MB is the lowest number that all of the above stripe sizes divide perfectly into, without generating a remainder.

Then if we factor in other things such as cache slot size and segment size as well as pre-fetch (based on tracks (256K) and multiples thereof) they all also divide perfectly in to 42MB.

Also with the USP V being track centric/cache slot centric (both 256K), it tends to internally map and manage things, such as external storage, in multiples of this track/slot size. Again, divides perfectly in to 42MB.

Further to this internal mapping... Let's not forget that because HDP borrows a lot from the Hitachi implementation of Copy-On-Write technology (COW) I will refer to the operation of allocating a new 42MB page to a LUN as a Borrow-On-Write, or BOW operation.

Each time a BOW operation occurs there is overhead including –

1. Search the free page table for the next available free page (if there is now logic on top of this to more evenly spread the load then the overhead will be more)
2. Update the Dynamic Mapping Table (DMT) and the free page table
3. Map the page into the DP-VOLs allocated page table
4. Make the blocks available for access

So with the above in mind, the smaller the page size, the more often these BOW operations will be required when growing a volume. Each time incurring a small overhead, so the less frequently they happen the better. Of course these operations only occur when a new page is demanded.

Also, the HDP Dynamic Mapping Table (DMT) is another layer that must be traversed in order to map LBAs in a LUN back to blocks within a Pool for normal read and write requests that do not require allocation of a new page.

Also, if Page size was smaller, the DMT would of necessity be larger due to there being more pages per Pool. As a result it would take longer to search and update. And when you think that each pool can have millions of pages, the DMT could get very large.