

Hitachi Unified Storage VM System

– Architecture and Concepts

A White Paper

By Alan Benway (Performance Measurement Group, Technical Operations)

Hitachi Data Systems Partner and Internal Use Only

March 28, 2013



Notices and Disclaimer

Copyright © 2013 Hitachi Data Systems Corporation. All rights reserved.

The performance data contained herein was obtained in a controlled isolated environment. Actual results that may be obtained in other operating environments may vary significantly. While Hitachi Data Systems Corporation has reviewed each item for accuracy in a specific situation, there is no guarantee that the same or similar results can be obtained elsewhere.

All designs, specifications, statements, information and recommendations (collectively, "designs") in this manual are presented "AS IS," with all faults. Hitachi Data Systems Corporation and its suppliers disclaim all warranties, including without limitation, the warranty of merchantability, fitness for a particular purpose and non-infringement or arising from a course of dealing, usage or trade practice. In no event shall Hitachi Data Systems Corporation or its suppliers be liable for any indirect, special, consequential or incidental damages, including without limitation, lost profit or loss or damage to data arising out of the use or inability to use the designs, even if Hitachi Data Systems Corporation or its suppliers have been advised of the possibility of such damages.

Hitachi Unified Storage VM[®], Hitachi Unified Storage 100[®], and Virtual Storage Platform[®] are registered trademarks of Hitachi Data Systems, Inc. in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

This document has been reviewed for accuracy as of the date of initial publication. Hitachi Data Systems Corporation may make improvements and/or changes in product and/or programs at any time without notice.

No part of this document may be reproduced or transmitted without written approval from Hitachi Data Systems Corporation.

WARNING: This document can only be used as HDS internal documentation for informational purposes only. This documentation is not meant to be disclosed to customers or discussed without a proper non-disclosure agreement (NDA).



Document Revision Level

<i>Revision</i>	<i>Date</i>	<i>Description</i>
1.0	October 2012	Initial Release
1.1	March 13 2013	GA02 enhancements, IO flow updates
1.2	March 28 2013	FMD details

References

Hitachi HUS 100 Architecture & Concepts Guide, August 2012

Hitachi VSP Architecture & Concepts Guide, March 2013

Contributors

The author would like to recognize and thank the following contributors or reviewers of the 1.0 version of this document:

Dave Campbell, Technical Product Director HUS VM

Kenneth Daniely, Americas Geo PM for HUS VM

Alan Davey, HUS VM Platform Lead

Wendy Roberts, Technical Director, Enterprise Storage, APAC

Ian Vogelesang, TechOps, Master Performance Consultant

Rob Whalley, Product Manager, Enterprise Storage Systems, EMEA

Table of Contents

Introduction	7
System Highlights.....	7
Hitachi Virtualization (UVM: Universal Volume Manager)	9
Hitachi Dynamic Provisioning (HDP)	9
Hitachi Dynamic Tiering (HDT).....	10
Glossary	10
Design Overview.....	14
Basic Theory of Operation.....	15
Architecture Details.....	15
Array Limits.....	16
System Hardware Description	16
DBs: Disk Boxes.....	18
Back End Layout.....	18
Blade and Module Details	19
Overview.....	19
I/O Process Concepts.....	20
Read Hits (from cache)	20
Read Miss (from disk)	20
Writes (to disk)	21
MAIN Blade (MAIN)	22
Overview	22
Cache Memory.....	22
Flash Drive for Cache Backup.....	23
HM ASIC	23
MP Blade (MPB).....	25
Overview	25
LDEV Ownership by MPU.....	27
Front-end Module: 4 x FC 8Gbps Ports.....	28
Overview	28
Tachyon Chip.....	28
Back-end Module: 8x 6Gbps SAS links.....	29
Overview	29

SAS Processor Chip (SPC).....	30
Cache Memory and Local Memory.....	31
Cache Memory Overview	31
CM Cache Regions	32
Cache Region Organization	33
Cache Logical Partitions (CLPR).....	34
MPU Partitions Within CLPRs	34
Creating Additional CLPRs.....	35
Drives and Drive Boxes	36
DB: 12-Disk LFF Tray (DBL)	37
DB: 24-Disk SFF Tray (DBS).....	38
DB: 48-Disk LFF Drawer (DBX).....	38
DB: 12-Disk FMD Tray (DBF).....	39
Drive Details	40
FMD Drives	41
SSD Drives (Ian Vogelesang)	41
SAS Drives (Ian Vogelesang).....	41
Fibre Channel Port Operations.....	42
I/O Request Limits and Queue Depths	42
External Storage Mode I/O	43
Cache Mode Settings	43
Other External Mode Effects	43
General Storage Concepts	44
Understand Your Customer's Environment	44
RAID Levels and Write Penalties	44
Parity Groups and system Groups	45
RAID Chunks and Stripes.....	46
LUNS (host volumes)	46
Number of LUNs per Parity Group	47
Port I/O Request Limits, LUN Queue Depths, and Transfer sizes	47
Port I/O Request Limits	47
Port I/O Request Maximum Transfer Size.....	47
LUN Queue Depth.....	47
Mixing Data on the Physical drives.....	48



Selecting the Proper Disks	48
Mixing I/O Profiles on the Physical drives	48
Front end Port Performance and Usage Considerations.....	48
Host Fan-in and Fan-out	49
Mixing I/O Profiles on a Port.....	49
Appendix 1. Open-V Volume RAID Mechanisms	50
Appendix 2. HUS VM: LDEVs, LUNs, VDEVs and More.....	52
Internal VDEV	56
External VDEV	57
Layout of internal VDEVs on the parity group	57
Appendix 3. Concatenated Parity Groups.....	61
Advantages of concatenated Parity Groups.....	63
Disadvantages of concatenated Parity Groups:	64
Using LDEVs on concatenated Parity Groups as Dynamic Provisioning pool volumes	64
Summary of the characteristics of concatenated Parity Groups (VDEV interleave):	65
Appendix 4. PCI Express Details.....	66
Appendix 5. T10 DIF Details	68

Hitachi Unified Storage VM System

– Architecture and Concepts

A White Paper

By Alan Benway (Performance Measurement Group, Technical Operations)

Introduction

This document covers the hardware architecture and concepts of operations for the **Hitachi Unified Storage System VM (HUS VM)**. This document is not intended to cover any aspects of program products, customer application software, customer specific environments, or features available in future releases.

This document will familiarize Hitachi Data Systems' sales personnel, technical support staff, approved customers, and value-added resellers with the features and concepts of the HUS VM. Users who will benefit the most from this document are those who already possess an in-depth knowledge of the enterprise **Hitachi Virtual Storage Platform (VSP)** architecture.

This document will receive future updates to refine or expand on some discussion as the internals of the design are better understood or as upgrades are released. **This version includes references to 200GB SSDs and DBF/FMD which will not be available until June 2013.**

System Highlights

The all new **Hitachi Unified Storage System VM** system fits in between the midrange HUS 100 family and the high-end VSP model. The HUS VM employs a HiStar based controller chassis and it uses the modular 12 and 24-slot disk boxes and 48-disk drawers, SAS cables, DDR3 cache DIMMs, and the Front-end and Back-end connectivity modules from the HUS 100 family. It also uses the DBF 12-slot FMD drive boxes (flash modules) common with the VSP.

The controller design is a "compacted logical" implementation of the HiStar design as used in the VSP, but it uses completely different hardware with some faster, newer generation parts. The HUS VM also reuses most of the system software from the VSP. So in general it could be seen as a smaller VSP that provides a lot of I/O power as well as most of the features of a VSP.

The HUS VM is currently marketed to fit in that market segment between high-end midrange and enterprise systems. The HUS VM offers an entirely new type of scalable high performance storage, capable of supporting demanding workloads while providing great flexibility. The HUS VM offers much higher performance, better scalability, higher reliability, and greater flexibility than any competitive offering today.

These are the distinguishing features of the HUS VM:

- The HUS VM has a single **5U control chassis** which contains the HiStar based controller and one or more 19" racks. Up to **1152 disks** (including up to 128 Solid State Drives or 96 Flash Modules) may be installed using up to 48 disk boxes in these racks.

- Each controller utilizes two types of blades, these being a **MAIN blade** and an **MP blade**. The MAIN blade provides the cache and all functionality other than running the system software, that being provided by the Intel Xeon processor on the MP blade.
- **MAIN Blades**
 - Each MAIN blade has their own plug-in modules that provide the FC connectivity to hosts (or to external storage or remote copy links) as well as the SAS connectivity to the disks. They also contain half of the system cache and a local flash memory for backing up the contents of cache. There is a custom **HM ASIC** at the heart of each MAIN blade, this being an integration of many distributed elements within the VSP design.
 - The connectivity within each MAIN blade, as well as the external connections between the two MAIN blades and to each MP blade, is provided by PCI Express 3.0 links. No user data passes between the MAIN and MP blades; these links are used for command transfers and shared (control) memory writes. See Appendix 4 for details on PCI Express concepts.
 - There are small *Front-end connectivity modules (FE modules)* and *Back-end drive controller modules (BE modules)* – both parts soon to be shared with the HUS 100 family - that plug into sockets on the MAIN blades. The FE modules are very simple, with a single Tachyon QE8 chip providing four 8Gbps Fibre Channel external ports. The BE modules each contain a single SAS Processor Chip (SPC) that provides 2 disk tray cable ports, each port having 4 x 6Gbps SAS links.
 - Like the VSP, the HUS VM uses a cache-based **Control Memory (SM)** system, often labeled as *Control Memory* on the VSP and earlier enterprise systems. The master SM copy for the entire system resides in the lower range of cache on the two MAIN blades. Additionally, each MP blade keeps the majority of metadata and control tables for those volumes (LDEVs) it manages in tables in Local Memory (high speed RAM on the MP blade). The majority (perhaps 80%) of all Control Memory accesses are simply reads to this local RAM copy. Updates are written to both the Local Memory and the master SM copy located in cache.
- **MP Blades**
 - The MP Blades each contain a 2.1GHz Intel Xeon **8-core processor**, **Local Memory** (8GB of RAM), and a pair of **flash modules** for holding the system software. The single 8-core processor is operated as a pair of logical 4-core processors (**MPUs**).
 - Each MPU controls all I/O operations for a discrete group of LDEVs (LUNs when they are mapped to a host port) in the same manner that they are managed by single VSD processor boards in the VSP array. When first created, each new LDEV is automatically assigned in a round-robin fashion to one of the four MPUs.
 - Each MPU executes the **system software** for the following processes for those volumes (LDEVs) that it manages:
 - Target mode (Open Systems hosts)
 - External mode (Virtualization of other storage)
 - BE mode
 - Replication Initiator – TrueCopy Synch or HUR Initiator (MCU) mode
 - Replication Target (RCU) mode.

- The HUS VM uses modular (from the HUS 100 family) 12-slot LFF disk boxes, 24-slot SFF disk boxes and 48-slot LFF high density box (or drawer). It also supports the 12-slot DBF drive box (FMD drives). Up to 48 of the DBS or DBL boxes (including up to 8 of the DBF boxes), or 24 of the LFF drawers may be used, or some combination supporting up to 1152 drives. The ENC's (SAS switches) in the boxes (2 ENC's) or drawers (4 ENC's) are mechanically the same but use different ENC firmware for the HUS VM.
- Current (through June 2013) drive choices include:
 - o **SFF drives:** 200GB MLC SSD, 400GB MLC SSD, 300GB 15K, 600GB 10K, and 900GB 10k SAS drives
 - o **LFF drives:** 3TB 7.2K SAS drives
 - o **FMD drives:** 1.6TB drives

Hitachi Virtualization (UVM: Universal Volume Manager)

The HUS VM provides the same **Hitachi Virtualization** mechanism as found on the VSP. Other customer storage systems (often being repurposed upon replacement by newer systems) may be attached to some of the HUS VM's FC ports. These ports are now operated in *External Initiator mode* rather than the default host *Target mode*. The LUNs that are supplied by these external systems are accessed and managed by hosts that are attached to other HUS VM FC ports (operated in Target mode). As far as any host is concerned, all of the visible virtualized LUNs passed through the HUS VM external ports to the host target ports simply appear to be normal internal LUNs in the HUS VM. The HUS VM's FC ports that attach to the external storage system's front-end ports are operated in "external" or "SCSI initiator" mode (attached to storage), rather than the usual "SCSI target" mode (attached to hosts).

Virtualized (external) storage should only be deployed as Tier-2 and Tier-3. Tier-0 and Tier-1 use should be limited to the internal HUS VM LDEVs which come from Parity Groups based on SSD or SAS drives. One benefit of virtualization is greatly simplified management of LUNs and hosts. Another major advantage of virtualization is the ability to dynamically (and transparently) move LDEVs from Tier-0 or Tier-1 Parity Groups (known as RAID Groups on modular systems) down to Tier-2 or Tier-3 external storage using a different RAID level or drive type. These LDEV migrations are able to proceed while the original source LDEV remains on line to the hosts, and the HUS VM will seamlessly switch over the mapping from the original LDEV to the new lower tier LDEV when completed. No changes on the host mount points are required.

Hitachi Dynamic Provisioning (HDP)

Hitachi Dynamic Provisioning (HDP) provides a logical mechanism for grouping LDEVs from multiple RAID Groups into a single Pool (with many such independent Pools possible) that presents an automatically managed, wide striped block device to one or more hosts. A Dynamic Provisioning Pool is defined with two or more LDEVs (**Pool Volumes**) from Parity Groups with one RAID level and one drive type. Use of 32 to 128 drives per Pool would be more typical. The Pool mechanism creates a structure of 42MB Pool pages from each LDEV within the Pool when initially created. This is similar to the use of a host-based logical volume manager (LVM) and its wide striping mechanism across all member LUNS in its "pool" with use of a (typically) large volume chunk size (usually 1MB or greater).

Dynamic Provisioning Volumes (**DPVOLs** or virtual volumes) are then created with a user specified *logical* size (up to 60TB) and then connected to a single Pool. The host accesses the DPVOL (or many of them – even hundreds) as if it were a normal volume (LUN) over one or more host ports. A major difference is that disk space is not physically allocated to a DPVOL from the Pool until the host has written to different parts of that DPVOL's Logical Block Address (**LBA**) space. The entire logical size specified when creating that DPVOL could eventually become fully mapped to physical space using **42MB Pool pages** from every LDEV in the Pool. If new LDEVs from new Parity Groups are added to the Pool later on, a *rebalance operation* (restripping) of the currently allocated Pool pages onto these additional RAID Groups can be initiated.

Hitachi Dynamic Tiering (HDT)

Hitachi Dynamic Tiering (**HDT**) is a feature that enhances the operation of an HDP Pool. Previously, each HDP Pool had to be created using one RAID level and one drive type. So a *high performance* Tier-1 Pool could be set up using 15K RPM SAS drives in a RAID-10 (2D+2D) configuration, a *standard performance* Tier-2 Pool could use 10K RPM SAS drives and RAID-5 (such as 3D+1P), and an *archive performance* Tier-3 Pool could be set up with 7200 RPM SAS drives in RAID-5 or RAID-6 for near-line storage. Tier-2 and Tier-3 Pools could also be created using LUNs from virtualized storage (typically with SAS drives for Tier-2 and SATA drives for Tier-3).

The new HDT feature allows a *single* Pool to contain RAID-5 and/or RAID-6 types of RAID Groups (Pool Volumes) and any type of drive. At this time, the use of RAID-10 Pool Volumes or external LUNs is not supported. HDT manages the various tiers within a Pool automatically. Management includes the dynamic relocation of a page based on frequency of back end disk I/O to that page. Therefore, the location of a Pool page (42MB) is managed by HDT according to host usage of that part of an individual DPVOL's LBA space. This feature can eliminate most user management of storage tiers within a subsystem and can maintain peak performance under dynamic conditions without user intervention. This mechanism functions effectively without visibility into any file system residing on the volume. The top tier is kept full at all times, with pages being moved down a tier to make room for new high-activity pages. This is a long term space management mechanism, not a real-time relocation service.

Glossary

At this point some definitions of the various terminology used is necessary in order to make all of the following discussions easier to follow. Throughout this paper the terminology used by Hitachi Data Systems (not Hitachi LTD.) will normally be used. As a lot of storage terminology is used differently in Hitachi documentation or by the users in the field, here are the definitions as used in this paper:

- **Array Group** (installable, drive feature) – The term used to describe a set of four physical drives installed into any disk trays (in any “roaming” order on HUS VM). When a set of one or two such Array Groups (four or eight drives) is formatted using a RAID level, the resulting *RAID formatted* entity is called a **Parity Group**. Although technically the term Array Group refers to a group of bare physical drives, and the term Parity Group refers to something that has been formatted as a RAID level and therefore actually has initial parity data (here we consider a RAID-10 mirror copy as parity data), be aware that this technical distinction is often lost. You will see the terms Parity Group and Array Group used interchangeably in the field.

- **Back-end Module (BE module, installable, *DKB* feature)** - a SAS drive controller module that plugs into a socket on a MAIN blade and provides eight back-end 6Gbps SAS links via two SAS 4-Wide ports. There are four of these modules installed in an HUS VM unless it is purchased as a diskless system, which can then have four extra FE modules (16 more FC ports) instead of the four BE modules.
- **Cache Directory** – the 500MB region reserved on each Cache Memory Group (CMG, a row of four DIMMs on a MAIN blade) for use by the MPUs in managing the User Data region in that CMG. On a system with all 16 cache DIMMs installed, there will be 2GB of Cache Directory space in all.
- **CBX** (Controller chassis) - Hitachi's name for the controller box.
- **CHB** (Channel Adapter) - Hitachi's name for the **FE module**.
- **Command Transfer Circuit** – the 5 LRP processors within each HM ASIC that route communications between FE or BE modules and the MPUs.
- **Controller Chassis (CBX feature)** – the 5U box in the bottom of the controller rack (DKC) which holds the HiStar controller unit.
- **Concatenated Parity Group** - A configuration where the VDEVs corresponding to a pair of RAID-10 (2D+2D) or RAID-5 (7D+1P) Parity Groups, or four RAID-5 (7D+1P) Parity Groups, are interleaved on a RAID stripe level on a round robin basis. A logical RAID stripe row is created as a concatenation of the individual RAID stripe rows. This has the effect of dispersing I/O activity over twice or four times the number of drives, but it does not change the number, names, or size of VDEVs, and hence it doesn't make it possible to assign larger LDEVs to them. Note that we often refer to RAID-10 (4D+4D), but this is actually two RAID-10 (2D+2D) Parity Groups interleaved together. For a more comprehensive explanation see the *Concatenated Parity Group* section in the appendix.
- **Control Memory (CM)** - the region in each MP blade's Local Memory (RAM) and in the system cache (in Cache Memory Group 0) that is used to manage all volume metadata and system states. In general, Control Memory contains all of the metadata in a storage system that is used to describe the physical configuration, track the state of all LUN data, track the status of all components, and manage all control tables that are used for I/O operations (including those of Copy Products). The overall footprint of Control Memory in cache can grow up to **40GB** (20GB + 20GB mirror).
- **Controller Rack (DKC)** – the base rack that contains the controller chassis and up to 16 SFF disk trays (or 5 LFF disk drawers).
- **DB** – (Disk Box) – Hitachi's name for the disk trays, as well as **RKA** (SFF tray) and **DENSE** (LFF drawer).
- **DBF** – the 12-slot drive box for FMD drives, organized as four rows of 3 modules.
- **DIMM** – (*Dual Inline Memory Module*) – a “stick” of RAM in cache or local memory
- **Disk Enclosure names** –
 - o DBL – 2U LFF 12-slot box
 - o DBS – 2U SFF 24-slot box
 - o DBX – 4U LFF 48-slot dense drawer
 - o DBF – 2U FMD 12-slot box

- **DKC** – Hitachi’s name for the controller rack.
- **DKB** – Hitachi’s name for the Back-end disk controller module (BE module).
- **DPVOL** – (configurable, **Dynamic Provisioning VOLUME**) - the Virtual Volume connected to an HDP Pool. Some documents also refer to this as a **V-VOL**. It is a member of a V-VOL Group, which is a kind of **VDEV**. Each DPVOL has a user specified size between 8GB and 60TB in increments of one block (512 byte sector) and is built upon a set of 42MB pages of physical storage (so each DPVOL should be specified as a multiple of 42MB).
- **Drive (disk)** – an SSD drive or a SAS disk. (No SATA disks are supported in the HUS VM.)
- **DRR - Data Recovery and Reconstruction** processors that manage RAID parity operations, data encryption, and drive formatting or rebuilds.
- **eLUN** (configurable, **external LUN**) – an external LUN is one which is located in another storage system and managed as though it were just another internal LDEV. The external storage system is attached via two or more **FC ePorts** on an HUS VM and accessed by the host through other HUS VM FC ports.
- **ePort** (configurable, **External Port**) – An FC port that is operated as an *external* port rather than a *target* (host) port, an HUR MCU port, or an HUR RCU port. An external system directly connects to an HUS VM via two or more Fibre Channel host ports instead of to a host. Those HUS VM FC ports used in this manner are changed from the default *host target mode* to an *initiator mode* (external Port) by use of the **Universal Volume Manager** (UVM) software product. The HUS VM controller will ‘discover’ any external storage systems LUNs exported on each ePort. The eLUN is used within the HUS VM as a **VDEV**, a logical container from which LDEVs can be carved. Individual external LDEVs may be mapped to a portion of or to the entirety of the eLUN. Usually a single external LDEV is mapped to the exact LBA range of the eLUN, and thus the eLUN can be “passed through” the HUS VM to the host.
- **FC Port** – any of the four 8Gbps fibre channel ports on one of the FE modules.
- **Feature** (package) – an installable hardware option (such as an FE module, a BE module, cache DIMMS, or drives) that includes two modules (one located in each controller), or four DIMMs, or four drives as a **package**.
- **FMD** – the flash module drive (currently 1.6TB) that installs in the DBF Drive Box.
- **Front-end Module** – (**FE module**, installable, **CHB** feature) – the FC connectivity interface module that plugs into a socket on a MAIN blade. This Open Fibre Channel module is used to attach hosts (servers or HNAS) to the system. Each module provides four 8Gbps FC ports. These FC ports may also be used to attach to external storage or to remote systems when using the Hitachi Universal Replicator product.
- **LDEV** (configurable, **Logical DEVICE**) – A logical volume internal to the system that can be used to contain customer data. LDEVs are uniquely identified within the system using a six hex digit identifier in the form LDKC:CU:LDEV. LDEVs are carved from a VDEV (see VDEV), and there are three types of LDEVs - internal LDEVs, external LDEVs, and DPVOLs. LDEVs may be mapped to a host as a LUN, either as a single LDEV, or as a set of up to 36 LDEVs combined in the form of a LUSE. Note: what is called an LDEV in all Hitachi enterprise systems and the HUS VM is called an LU or LUN in HDS modular systems like the Hitachi Unified Storage 100 (HUS 100) or Adaptable Modular Storage (AMS) families

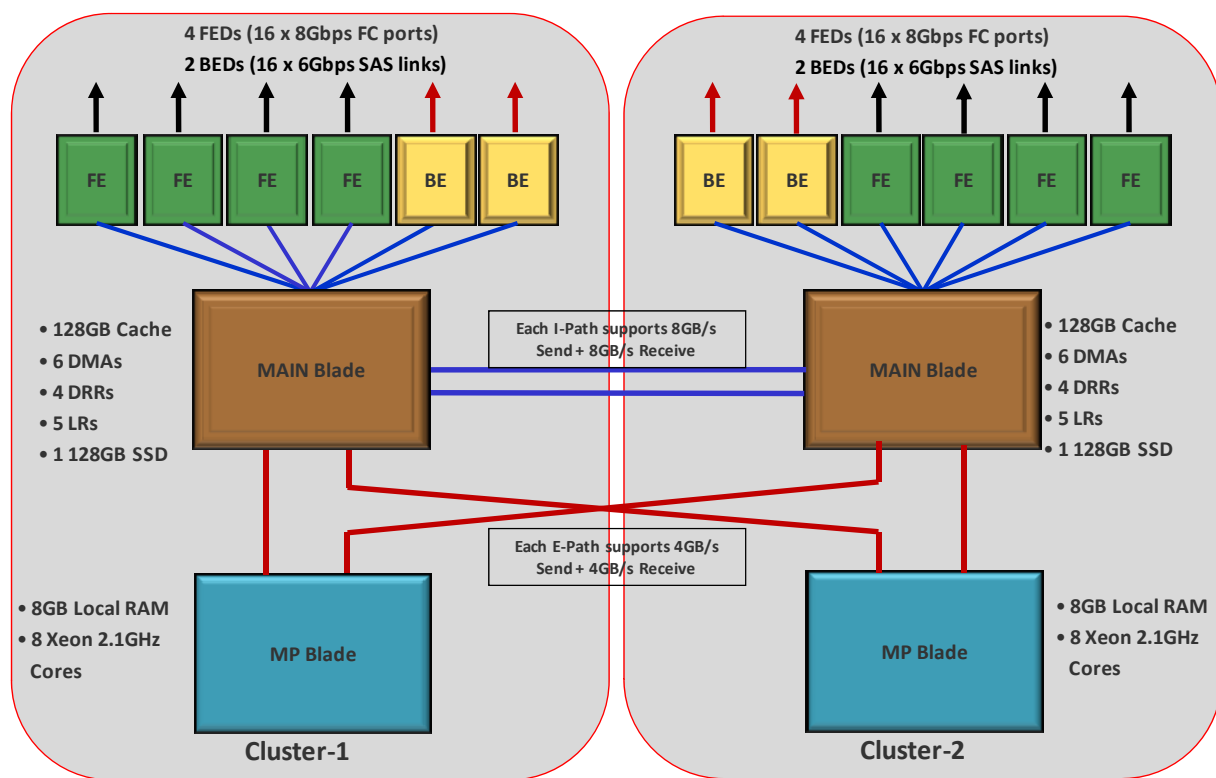
- **LRP (Local Router Processor, or Command Transfer Circuit)** – the 5 interface processors located inside each HM ASIC for managing control conversations between FE or BE modules and the MPUs.
- **LUN (configurable, Logical Unit Number)** – the host-visible identifier assigned by the administrator to an existing LDEV to make it usable on a host port. An *internal* LUN has no actual queue depth limit (but 32 is a good rule of thumb) while an *external* (virtualized) eLUN has a Queue Depth limit of 2-128 (adjustable) per eLUN per external port path to that eLUN. In Fibre Channel, the host HBA Fibre Channel port is the *initiator*, and the system's virtual Fibre Channel port (or Host Storage Domain) is the *target*. Thus the Logical Unit Number is the *number* of the logical volume within a target.
- **MP (microprocessor)** – the Intel Xeon eight-core 2.1GHz CPU used on the MP blade.
- **MPU (4-core logical processor)** - the logical 4-core device superimposed on the physical MP.
- **Open-V** – the name of the RAID mechanism on HUS VM (and VSP for Open hosts). See Appendix 1 for details.
- **Parity Group (configurable, RAID Group)** – a set of one or two system Groups (a set of 4 or 8 drives) formatted as a single RAID level, either as RAID-10 (often referred to as RAID-1+ in HDS documentation), RAID-5, or RAID 6. The VSP's Parity Group types are RAID-10 (2D+2D), RAID-5 (3D+1P), RAID-5 (7D+1P) and RAID-6 (6D+2P or 14D+2P). The Open-V RAID chunk (or stripe) size is fixed at 512KB. Internal LDEVs are carved from the VDEV(s) corresponding to the formatted space in a Parity Group, and thus the maximum size of an internal LDEV is determined by the size of the VDEV that it is carved from. The maximum size of an *internal* VDEV is approximately 2.99 binary TB. If the formatted space in a Parity Group is bigger than 2.99 TB, then multiple VDEVs must be created on that Parity Group. Note that there actually is no discrete 4+4 Parity Group type - see *Concatenated Parity Group*.
- **PCIe (PCI Express)** – a multi-channel serial bus connection technology that supports x1, x4, and x8 lane configurations. The resulting connection is called a “PCIe link”. The HUS VM uses the x8 type in most cases. The x8 link is capable of 8GB/s send plus 8GB/s receive in full duplex mode (i.e. concurrently driven in each direction). See Appendix 4 for details.
- **PDEV (Physical DEVICE)** – a physical internal drive.
- **RAID-1 or RAID-1+** – used in some documents to describe what is usually called “RAID-10”, a *stripe of mirrored pairs*. Thus when we say “RAID-1” in the context of a Hitachi enterprise VSP-family system, we mean the same thing as when we say “RAID-10” in the context of an AMS modular system. Note that alternative RAID-0+1 used by some vendors is quite different, as it is the very vulnerable *mirror of two RAID-0 stripes*, where if one disk fails, all protection is lost. In a mirror of stripes, it's not that you lose the data on a single drive failure, because after all, it's still a mirror, but in a mirror of stripes if one drive fails, the entire stripe goes down, and you are very vulnerable to a 2nd drive failure in the other stripe. In Hitachi's RAID-1 (RAID-10) where we have a stripe of mirrors, if two drives fail in different mirror pairs, then each mirror pair is still “alive” within the stripe and thus no data are lost.
- **Shared Memory (SM)** - the alternate term used for Control Memory.
- **SVP (installable, Service Processor feature)** - a blade PC running Windows that is installed in a slot in the controller chassis and used for system configuration, system software updates, runtime dumps, and various performance or status reports.

- **VDEV** (configurable) – The logical storage container from which LDEVs are carved. There are two types of VDEVs on the HUS VM:
 - **Internal** VDEV (2.99 binary TB max): maps to the formatted space within a parity group that is available to store user data. LDEVs carved from a parity group VDEV are called internal LDEVs.
 - **External** storage VDEV (4 binary TB max): maps to a LUN on an external (virtualized) system. LDEVs carved from external VDEVs are called external LDEVs.

Design Overview

Unlike most storage systems from other vendors, the HUS VM is a purpose-built system designed from the ground up to be a high performance, heavily featured storage system. In fact, all Hitachi midrange and enterprise storage systems are purpose-built. In the HUS VM (like all Hitachi designs), there are separate intelligent components from which the system is created. These components are operated in parallel to achieve high performance, scalability, and reliability.

Figure 1. Block Diagram Overview of the HUS VM Controller



An HUS VM system will have one controller rack plus one or more optional drive expansion racks. The controller rack (**DKC**) has the Controller Chassis (**CBX**) which contains the HiStar based controller. There are four types of Disk Boxes (**DBs**), with a **DBL** (12-slot LFF), a **DBS** (24-slot SFF) a **DBX** (DENSE, 48 3.5" drive slots in a drawer), and a **DBF** (12-slot FMD). The DBL, DBS, and DBX disk boxes are carried over from the HUS 100 modular line, but they operate with different internal firmware.

The HUS VM uses four types of blades and blade modules in the CBX:

- MAIN blades (**MAIN**) with the HM ASIC, system cache and sockets for their FE and BE modules
- MP blades (**MPB**) with the Intel Xeon processor and 8GB of Local Memory
- Front-end Connectivity modules (**FE**) with 4 x 8Gbps FC ports
- Back-end Drive Controller modules (**BE**) with 8 x 6Gbps SAS links in two SAS ports (each with 4 links)

Basic Theory of Operation

The two **MAIN blades** are the center of the design, with their custom **HM ASIC** doing much of the routine work with the 19 specialized processors embedded within each of them. The FE connectivity and BE drive controller modules are inserted into sockets on the MAIN blade which controls them. The MAIN blade controls them via its local **LRP processors** (also known as **Command Transfer Circuits**). The system cache is also located on the two MAIN blades. The MAIN blades cross-connect with a pair of external links (**I-paths**) over which DMA transfers between physical cache regions are managed. There are two other dedicated links (**E-paths**) to the MP blades over which all communications with them are managed. No data crosses these MP blade links, only FE/BE module commands and Control Memory information (usually only writes).

The two **MP blades** do all of the “thinking”, executing the system software and managing all I/O. They communicate with the FE and BE modules via those modules’ associated MAIN blade LRP processors. As each MP blade is directly connected to both MAIN blades, every MAIN blade FE and BE module can be controlled by either MP blade. The FE and BE modules touch user data, and all data movement in or out of cache is performed using the DMA channels within the FE and BE modules. All transfers within cache are performed by the HM ASIC DMA channels. The necessary *User Data* cache addresses to use for host data are provided by the owning MPUs. An owning MPU is that MPU which manages all I/O to a particular host LUN (or LDEV, to use the internal name). The MPUs each control a private region of the cache space spanning the two MAIN blades and manage their LDEVs in that region.

The **system cache** is managed in much the same way as it is on the VSP. While the MPUs each manage a set of initially allocated 64KB cache segments (with a **Global Free List** managing any unallocated 64KB segments as a system reserve for temporary heavy demands), the MPUs themselves do not have any read or write access to these same data regions. Similarly, only the MPUs have access to the reserved Control Memory regions in cache. The system software prevents the MPUs from accessing the data regions of cache, and the FE and BE modules are similarly prevented from accessing the Control Memory region. The FE and BE modules also have private reserved regions within cache (DXBF) which they use for temporary working space.

Architecture Details

The HUS VM system can be configured as a single rack with the controller chassis (CBX) and up to 16 x 2U disk boxes (DBL, DBS, and DBF) or 8 x 4U LFF Dense drawers (DBX), or some mixture. If more trays or drawers are required then from one or more additional racks may be added to the system. The system limit is 48 logical disk boxes in all. Note that each DBX drawer is two logical trays (each half – with 24 drives – functions as an independent unit). Up to 1152 drives are supported, including up to 96 FMDs or 128 SSDs.

Array Limits

Table 1 below shows the major system limits for the HUS VM and the VSP (both single and dual chassis).

Table 1. Summary of Maximum Limits, Virtual Storage Platform and Unified Storage Platform VM

Table of Limits	VSP		HUS VM
	Single Chassis Controller	Dual Chassis Controller	Maximums
Data Cache (GB)	512	1024	256
Raw Cache Bandwidth	64GB/s	128GB/s	85GB/sec
Control/Shared Memory (with mirror, GB)	28+28		20+20
Cache Directory (GB)	4	8	2
SSD Drives	128	256	128
2.5" Disks (SAS)	1024	2048	1152
3.5" Disks (SAS)	640	1280	1152
Logical Volumes	65280		16384
Max Internal Volume Size	2.99TB		2.99TB
Max Thin Image Volume Size	4TB		-
Max External Volume Size	4TB		4TB
IO Request Limit per Port (rule-of-thumb, no real limit)	2048		2048
Queue Depth per LUN (rule-of-thumb, no real limit)	32		32
Number of Cache Partitions	32		29
Minimum Partition Size, Increment (GB)	4, 2		4, 2
HDP Pools	128		128
Max Pool Capacity	4PB		2PB
Max Capacity of All Pools	5PB		2PB
LDEVs per Pool (pool volumes)	1024		1024
Max Pool Volume size (internal/external)	2.99/4TB		2.99/4TB
DP Volumes per Pool	63,232		15,626
DP Volume Size Range (No SI/TC/UR)	46MB-60TB		46MB-60TB
DP Volume Size Range (with SI/TC/UR)	46MB - 4TB		46MB - 4TB

System Hardware Description

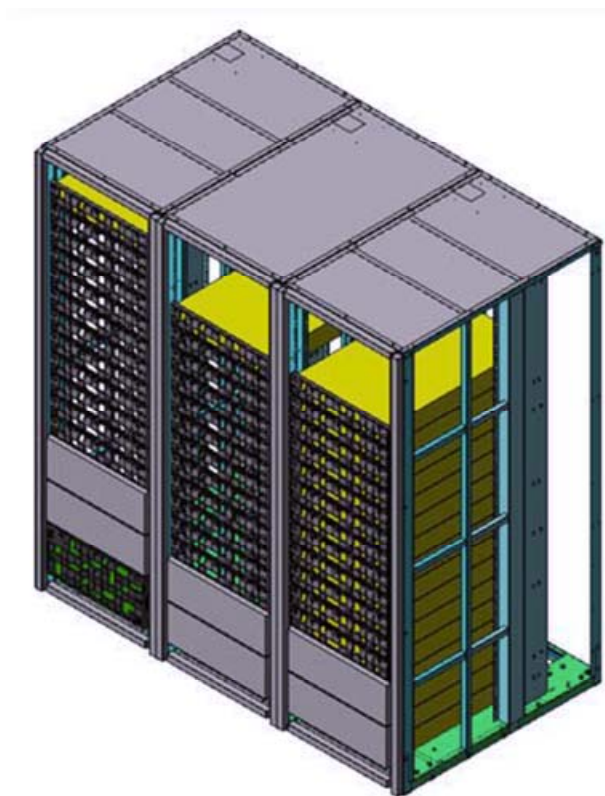
The figure below shows an HUS VM with 48 2U Drive Boxes installed in three racks. The controller chassis is at the bottom of the rack on the left. Notice how there is unused space (required) covered by gray panels in each rack. The use of the heavy DBX 48-slot drawers is limited to the 26U level unless special rack installation measures are followed to stabilize them.

A fully configured HUS VM system includes the following:

- 1 or more 19" standard racks
- A Controller Chassis (DKC) providing:
 - 2 Main blades, each with:
 - 128GB of cache (8 x 16GB DDR3 DIMMs)
 - 4 FE modules (4 x 8Gbps FC ports each, 16 ports total)

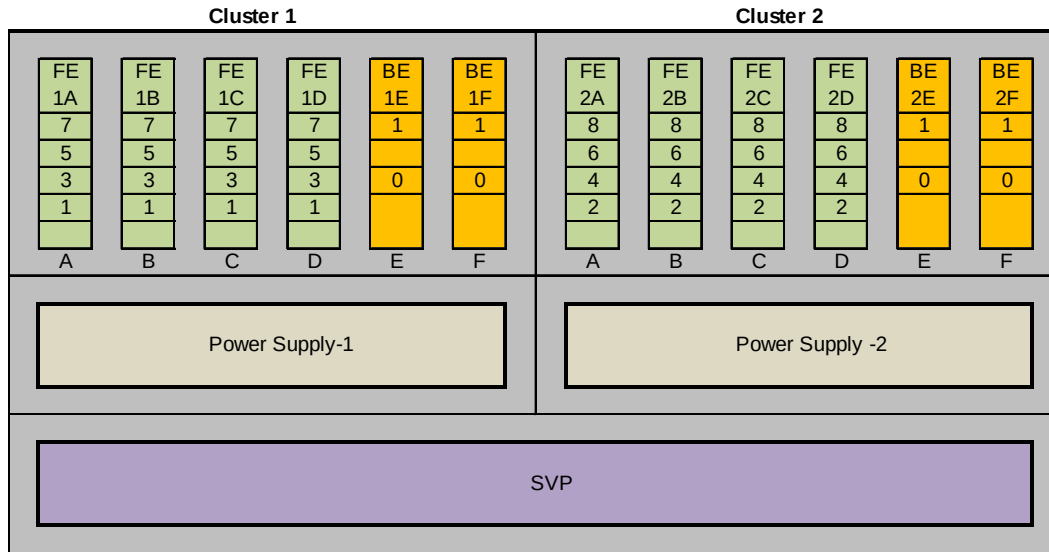
- 2 BE modules (8 x 6Gbps SAS links each, 16 links total)
 - 1 x 80GB or 1 x 128GB flash drive for cache backup
- o 2 **MP blades**, each with:
 - 1 x 8-core Intel Xeon 2.1GHz processor
 - 8GB of Local Memory (2 x 4GB DIMMs)
- Up to 1152 drives (SAS, SSD, and FMD), using:
 - o 2.5" drives (24-slot disk boxes)
 - o 3.5" drives (12-slot disk boxes and 48-slot high density drawers)
 - o Up to 128 SSDs are supported
 - o Up to 96 FMDs are supported in 8 DBF trays

Figure 2. HUS VM System with Three Racks and 48 2U Trays



The two MAIN blades (upper bays) and two MP blades (lower bays) are installed the four bays in the front of a controller chassis with two power boundaries called Cluster-1 (right side) and Cluster-2 (left side). Each blade has dual fans on the front of the assembly (what you see when viewing a chassis from the front). There is a MAIN and an MP blade in each such Cluster sharing a power supply. On the rear of the chassis are the slots for the FE and BE modules, the power supplies, and the Service Processor (SVP) unit. Figure 3 shows a rear view of the chassis.

Figure 3. Controller chassis (CBX) organization



The MAIN blade slots for the FE and BE modules are labeled “A-F”. The FE module port names are either “1, 3, 5, or 7” (Cluster-1) or “2, 4, 6, or 8” for Cluster-2. The BE module SAS port names are “0 or 1”. For example, the name of the first FE module in Cluster-1 is FE-1A. Port 7 would be FE 1A-7. For a diskless system dedicated to virtualization use, the four BE modules can be replaced by four more FE modules to provide a total of 48 x 8Gbps FC ports.

DBs: Disk Boxes

There are four types of disk boxes (known as trays and drawers on the HUS 100 products) supported on the HUS VM. These include:

- 2U disk box (**DBL**) with three rows of four Large Form Factor drive slots
- 2U disk box (**DBS**) with one row of 24 Small Form Factor (SFF) drive slots
- 4U dense disk drawer (**DBX** or **DENSE**) with 48 Large Form Factor (LFF) drive slots (in one drawer, with dual 24-slot sub-assemblies that are two independent disk boxes).
- 2U flash drive box (**DBF**) with four rows of three FMD slots

Each rack (including the DKC) can hold up to 16 2U drive boxes or 8 of the 4U DBX drawers. The overall back-end layout of BE module SAS ports and DBs (disk trays) is illustrated in Figure 4.

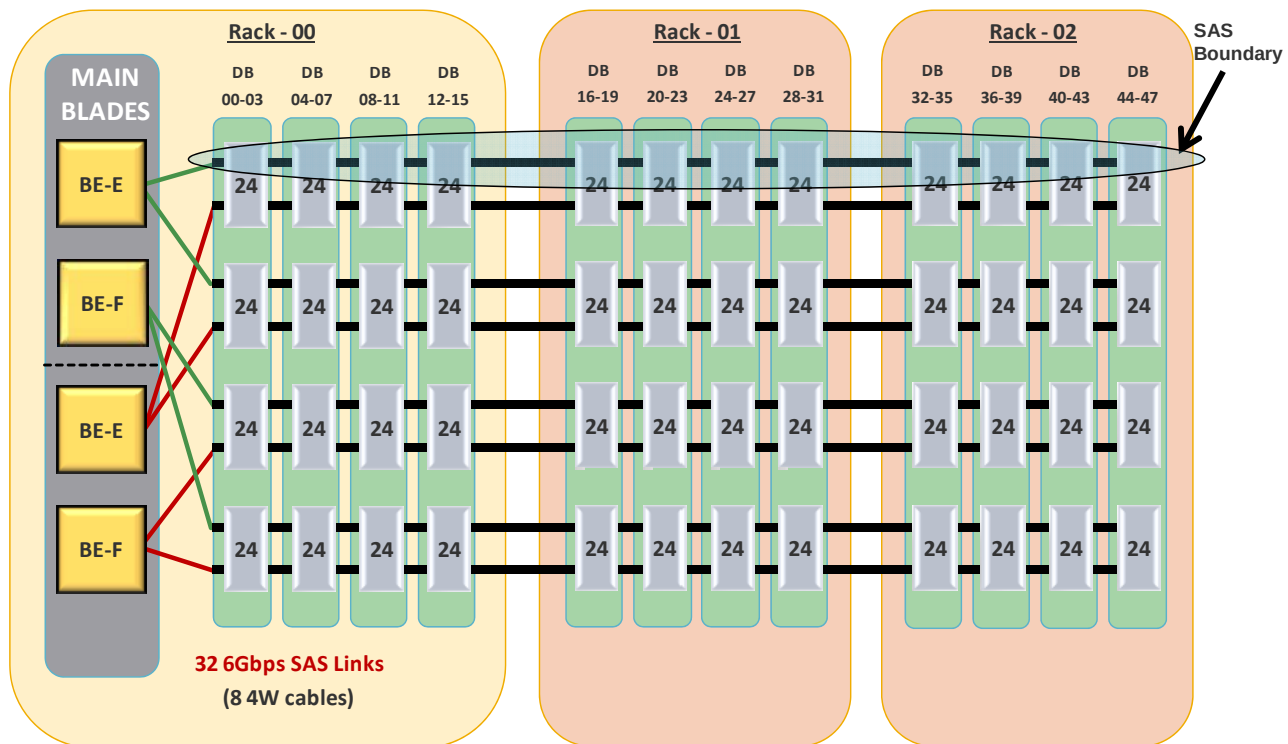
Back End Layout

The green shadings illustrate a (vertical column, in this figure) row of four 24-slot disk trays (or a row of 4 logical 24-slot trays from two DBX drawers). Each gray box within them is a disk box (or logical 24-slot box from a drawer). The first row of disk boxes would be named DB-00, DB-01, DB-02, and DB-03. The paths connecting the BE ports to each horizontal row (in this figure) or “stack” of disk boxes are SAS 4-Wide ports, with 4 independent 6Gbps SAS links within them. So each disk box is connected via 8 active

SAS links from two BE modules. The half of a stack of disk boxes that share the same SAS port can also be called a “SAS boundary”.

In this figure, the horizontal “stack” of boxes in the first column of boxes would include *DB-00*, *DB-04*, and *DB-08* and so forth to *DB-44* in Rack-02. Each stack can address up to 288 drives. These same drives (dual ported) are also accessed by another set of four SAS links (from a different BE module in the opposite cluster) as the next SAS boundary.

Figure 4. Overview of the BE ports to 48 DBS disk trays



Blade and Module Details

Overview

This section describes the MAIN and MP blades as well as the FE and BE modules in detail. The HUS VM uses four types of blades and installable modules in the controller chassis (CBX):

- MAIN blades (**MAIN**) with the HM ASIC, system cache and sockets for FE and BE modules
- MP blades (**MPB**) with the MPU logical processors and 8GB of Local Memory
- Front-end Connectivity modules (**FE**) with 4 x 8Gbps FC ports
- Back-end Drive Controller modules (**BE**) with 8 x 6Gbps SAS links in two SAS Wide ports

The two **MAIN blades** are the center of the design, with their custom HM ASICs doing much of the routine work with the 19 specialized processors embedded within each of them. The small FE and BE modules are plugged into sockets on a MAIN blade (as they are plugged into controller units on the HUS 100 systems). These modules are optional extensions to each MAIN blade, not an independent unit like

the autonomous FED and BED boards on the Grid on a VSP. The two **MP blades** do all of the “thinking”, executing the system software and managing all I/O. They indirectly control all of the FE and BE modules via the LRP processors within the associated MAIN blade HM ASIC.

The **FE modules** (plugged into slots on the MAIN blade that they belong to) are based on a single Tachyon QE8 chip, which is a powerful processor with a lot of independent functionality (described in detail lower down). Similarly, the **BE modules** (plugged into slots on the MAIN blade that they belong to) are based on a powerful dual-core SAS processor which also functions mostly on its own. The Tachyon and SAS processors communicate with their MAIN blade’s **LRP processors** (or *Command Transfer Circuit*, but they have far more power than this label implies). This is their command processing connection to the MPU logical processors located in the MP blades.

Each Tachyon or SAS processor also has several **DMA channels** built-in. These are used to directly access the **DXBF** and **User Data regions** of cache (described later in this paper). Access to the User Data regions first requires assignment of a cache address from the MPU that owns the LDEV in question.

I/O Process Concepts

This section is based on our current understanding of the manner in which the HUS VM processes internal commands. As we learn more (from ITPD revelations) or determine new details through our own forensic lab tests (where many of the low-level detail questions get sorted out), this section will be revised as needed.

The **FE module Tachyon chips** communicate with the attached MAIN blade’s **HM ASIC** (using the ASIC’s **LRP** processor associated with that MAIN blade module slot) to get the host I/O requests scheduled. The LRP processor creates and hands off a “job” to the owning **MPU** to execute. The LRPs have tables that contain maps of LDEV-to-MPU ownership. The MPU then responds to that LRP with a cache address (whether for a read hit, a read miss or a write buffer). If a read miss or write request, the MPU will spawn one or more additional jobs and dispatch these to the one or more BE module(s) **SPC Processors** that will then perform the requested physical disk read or write. For writes, the BE module(s) use one of the **DRR** processors within their MAIN blade’s HM ASIC to generate the parity blocks required and then write these blocks to the appropriate disks. There will be two disk updates required for RAID-10 or RAID-5, and three updates for RAID-6.

Read Hits (from cache)

For **read hits**, the FE Tachyon is instructed by the owning MPU - via that FE’s assigned MAIN LRP processor - to read a certain number of blocks already in cache (from a recent previous read or completed write) and to respond to the waiting host read request. One of the Tachyon chip’s DMA channels will directly read from that User Data cache address (from either MAIN blade, using the I-path link for MAIN-to-MAIN transfers if the required blocks are in the other MAIN blade’s cache) and transmit those blocks to the host.

Read Miss (from disk)

For a **read miss** (data not found in cache), the host request (via the LRP) to the owning MPU creates a second job which then schedules a read from disk by one of the BE modules to a target User Data region cache address specified by the MPU. The BE SPC can directly push that data directly to that cache address using one of its own DMA channels. When completed, the MPU responds to the waiting FE request with the cache address of the data, and the FE Tachyon then reads the blocks from the User Data region and completes the host I/O.

Writes (to disk)

For every host **write request**, the FE module Tachyon chip communicates the request to its assigned LRP. The Tachyon then moves the host data directly into a reserved region of cache (the DXBF region) using a DMA transfer by one of the Tachyon's four internal DMA channels. The Tachyon already has (or the LRP tells it) a memory address within its MAIN blade's local **FE DXBF region** to use as a temporary transfer buffer. That FE's assigned MAIN blade LRP then creates a task (job) for the owning MPU (for that LDEV) on the proper MP blade. That MPU then creates a job that sends commands (via the LRP processors on the MAIN blades that are attached to them) to the two or three BE modules that will perform the necessary read-write operations.

Next, the job request that is created by the LRP processor for that write request gets a pair of **target User Data cache addresses** (one from the User Data cache on each MAIN blade) from the owning MPU. The MPU then instructs one of that MAIN blade's DMA channels (in the ASIC) to move the blocks from that MAIN blade's local FE DXBF into usable addresses in User Data cache on both MAIN blades (duplexed - mirrored - for redundancy). One of these MAIN DMA transfers will use the I-path to reach the mirror location on the other MAIN blade's cache. Once the blocks have been written to both regions of cache, the original write blocks in the FE "scratchpad" DXBF are then deleted and the MPU tells the managing LRP to send a write completion message to the Tachyon, whereby the Tachyon then notifies the host.

The owning MPU will next create a back-end job for that LDEV which it routes through two or three MAIN LRPs to those BE modules that will perform the writes (on either MAIN blade, depending on where the BE modules are located). The BE modules SAS SPCs will then receive the MPU commands via their local MAIN's LRP to operate on this user data, and the SPCs can directly access the data in cache via their DMA channels.

For RAID-10 LDEVs there will be two write operations, perhaps on two BE modules, read directly from the User Data region of cache. For RAID-5 LDEVs, there will be two reads (old data, old parity) and two writes (new data and new parity), probably on two BE modules. For RAID-6 LDEVs there will be three reads (old data, old P, old Q) and three writes (new data, new P, new Q) using two or more BE modules. [Note: RAID-6 has one parity chunk (P) and one redundancy chunk (Q) that is calculated with a more complicated algorithm.]

The BE modules use the User Data region of cache for handling the parity blocks, with the aid of that MAIN blade's internal DRR processors (HM ASIC). The old data and parity (or two "parities" for RAID-6) are read by BE SPCs into the assigned cache region (for that LDEV). One or more DRR processors on that same MAIN blade will be given the source addresses of the new host data blocks in User Data cache, the cache addresses of the just-read blocks of old data and old parity (or two parities for RAID-6), and the cache addresses to use for the output of the XOR operations. The DRRs then perform the XOR operations using these different source and target addresses. Once they are done, the BE modules then perform the actual writes to disk from the User data region (new data blocks and new parity blocks). These cache segments are released upon completion of the writes, as is one of the duplexed write segments in the User Data region (the other write segments are now available for subsequent read hits).

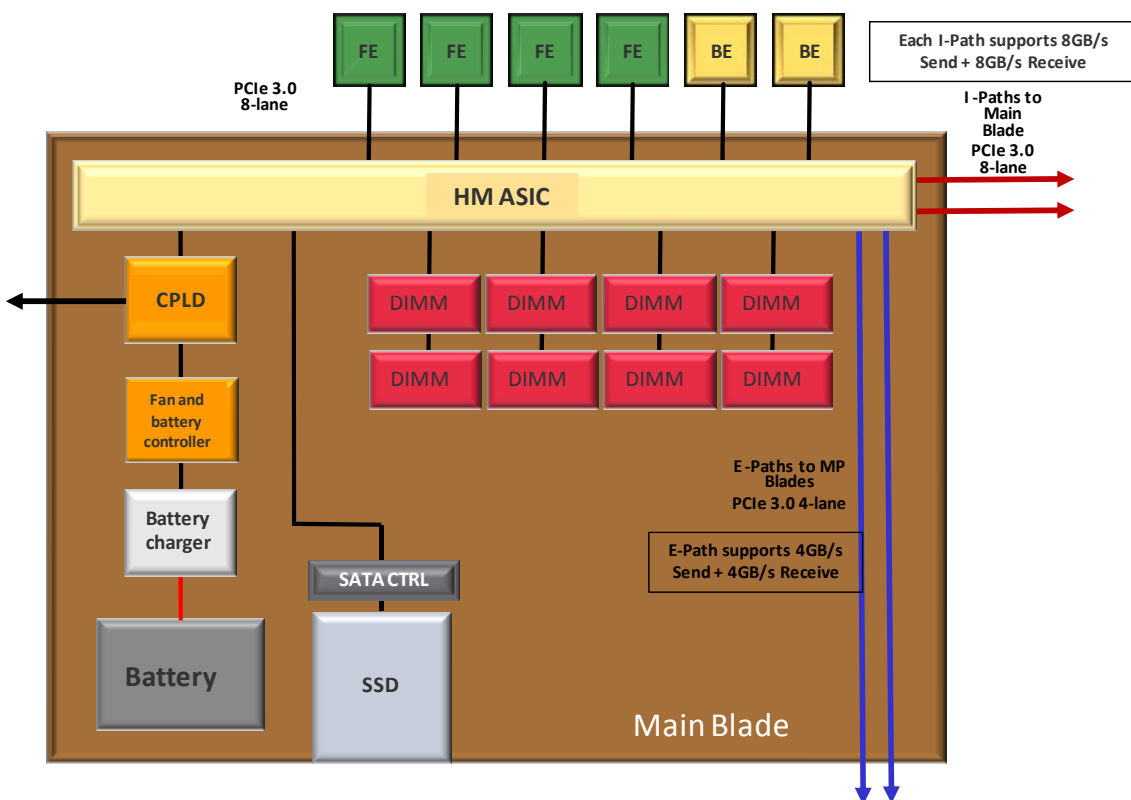
MAIN Blade (MAIN)

Overview

The two **MAIN blades** are the center of the HUS VM design, with their custom **HM ASIC** doing much of the routine work using the 19 specialized processors embedded within each of them. FE and BE modules plug into a MAIN blade, which then interacts with them via the local **LRP processors** (also known as **Command Transfer Circuits**) over dedicated PCIe links.

Each MAIN blade takes on the duties of these corresponding hardware elements from the VSP design: Grid Switch boards, DCA cache boards, and FED and BED board local DXBF scratchpad memory and Data Accelerator ASICs (with their internal LRP, DRR, and DMA channels).

Figure 5. MAIN blade Details



The system cache is located on the two MAIN blades, using up to 16 DIMMs (8 per blade). The MAIN blades have a pair of PCIe x8 external links (I-paths) over which DMA transfers between the User Data regions on the two MAIN blades are directed. There are two other dedicated PCIe x4 links (E-paths) to each MP blade over which all communications with them are managed. No data crosses these MP blade E-paths, only FE/BE module commands and Control Memory writes (updates to the global tables).

Should a MAIN blade and/or an MP blade fail, the system will continue to run (at reduced performance) using the surviving blades.

Cache Memory

Each MAIN blade has eight **DDR3-1300 DIMM** slots, organized with four memory controllers within the HM ASIC, each with 2 DIMM slots (thus 8 independent banks of DIMMs per system). There is a choice of

either using 8GB or 16GB DIMMs, or a mixture per MAIN blade. Each row of DIMMs (across the memory controllers) must be of the same type. Each MAIN blade supports the following cache sizes:

- 32GB (4 x8GB DIMMs)
- 64GB (8 x8GB DIMMs or 4 x16GB DIMMs)
- 96GB (4 x8GB and 4 x16GB DIMMs)
- 128GB (8 x 16GB DIMMs)

Each DDR3 memory controller (four per ASIC) has a peak theoretical transfer rate of 10.62GB/s, so the entire system cache has a peak rating of about 85GB/s. [Note: The HUS 150 with 32GB of cache is rated at 42.2GB/s. A single chassis VSP with 512GB of cache is rated at 32GB/s for reads and 32GB/s for writes, as limited by its 32 x PCIe 1.0 Grid Switch paths to the 8 VSD boards.]

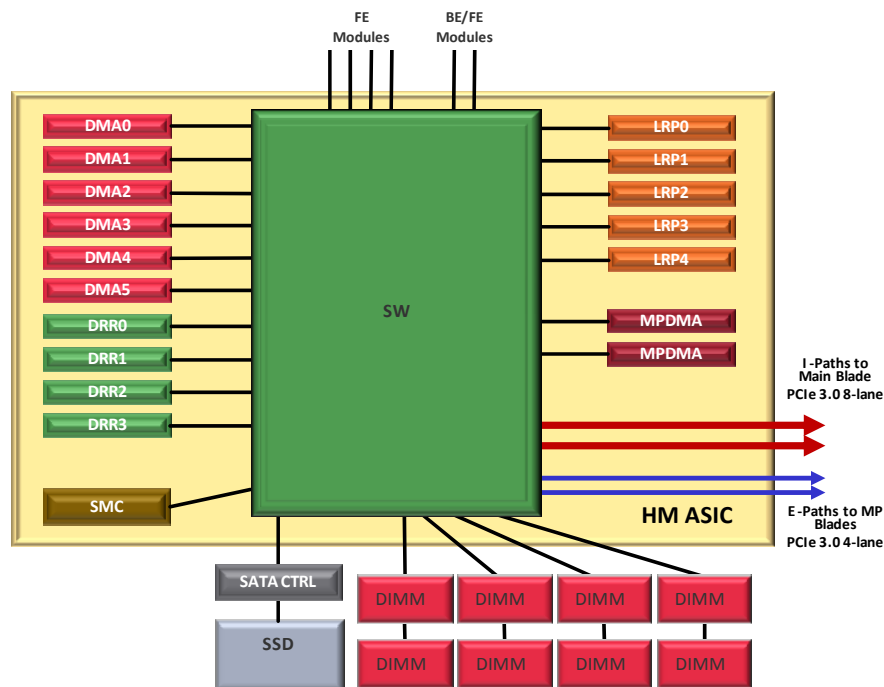
Flash Drive for Cache Backup

Each MAIN blade has one local 80GB or 128GB flash drive installed for use in backing up its entire cache space in the event of a system shutdown. If there is a power outage, this is the backup target for a MAIN blade's entire cache space. In the case of a planned system shutdown, it is the backup target for just the Control Memory region. During a power outage, the on-blade battery power keeps the DIMMs, the flash drive and the MAIN blade functioning while the HM ASIC destages cache to the flash drive. There is enough battery power to support two or three such outages back-to-back without recharging.

HM ASIC

The custom Hitachi HM ASIC manages everything attached to or on the MAIN blade. The following figure illustrates the major functions within the ASIC.

Figure 6. MAIN Blade HM ASIC Details



The major functions within each HM ASIC include:

- **Grid Switch** paths:
 - six PCIe x8 links for I/O modules (up to 4 FE and 2 BE modules or 6 FE modules)
 - two PCIe x8 links (I-paths) to cross connect to the other MAIN blade
 - two PCIe x4 links (E-paths) that connect to the MP Blades (one path each)
 - each link to the internal processors (LRP, DMA, DRR, SMC, MPDMA)
- Four **memory controllers** per MAIN blade HM ASIC, each controller having two DDR3-1300 DIMM slots with 10.6GBps of bandwidth (so 42.4GBps per MAIN blade of raw cache throughput).
- Six **DMA channels** used to move data within cache on the Main blades. These are shared by the LRP processors in a round robin fashion.
- Four 4 **DRR (Data Recovery and Reconstruction) processors**, used to:
 - generate RAID parity masks and blocks
 - format disks
 - control data encryption
 - manage disk rebuilds (Data Recovery and Reconstruction)
 - These are shared by all 4 MPUs, but the preference is to limit the DRR use to the same MAIN blade where the target User Data cache and DXBF addresses are located.
- Five **LRP processors** are used to route commands among the FE and BE modules and the MP Blades. These chips know which MPU owns each LUN. The fixed association between LRPs and MAIN modules are as follows:
 - LRP0 -> FE-A, FE-C
 - LRP1-> FE-B, FE-D
 - LRP2-> BE-E or FE-E
 - LRP3-> BE-F, FE-F
 - LRP4-> failure processing
- An **SMC processor** used for managing the local Control Memory region in cache.
- Two **MPDMA channels** manage direct cache access (Control Memory, Cache Directories) by the two MP blades.
- **Battery controller** - for the MAIN blade battery.
- **Fan controller** - for the dual fans on that cluster.
- **CPLD Processor** – for monitoring and reporting status to the SVP.
- A **SATA controller** that operates the local 80GB SSD used to back up all of the contents of cache during a power failure. This SSD is mounted in a removable tray on the front of the MAIN blade behind the fan.

[Note: The **Data Accelerator ASICs** on the VSP systems include the LRP, DMA, and DRR functions, and these are located on each FED and BED board.]

Table 2. Names of HM ASIC Element Names by Cluster

	ASIC	CL1	CL2
DMA	DMA0	DMA-10	DMA-20
	DMA1	DMA-11	DMA-21
	DMA2	DMA-12	DMA-22
	DMA3	DMA-13	DMA-23
	DMA4	DMA-14	DMA-24
	DMA5	DMA-15	DMA-25
DRR	DRR0	DRR-10	DRR-20
	DRR1	DRR-11	DRR-21
	DRR2	DRR-12	DRR-22
	DRR3	DRR-13	DRR-23
LRP	LRP0	LRP-10	LRP-20
	LRP1	LRP-11	LRP-21
	LRP2	LRP-12	LRP-22
	LRP3	LRP-13	LRP-23
	LRP4	LRP-14	LRP-24

MP Blade (MPB)

Overview

The two **MP blades** execute the system software, do all of the “thinking” for processing I/O, and send operational commands to FE and BE modules via the LRP processors within each HM ASIC. Basically, the MPUs tell the FEs and BEs what they have to do and where in the User Data cache to operate.

As each MP blade is directly connected to both MAIN blades, every FE and BE module can be controlled by the MPUs on either MP blade. The FE and BE modules touch user data, and all data movement in or out of cache is performed using the DMA channels within the FE and BE modules. The ASIC DMA channels control movement of data over the I-paths between the cache spaces on the two MAIN blades. Target cache addresses are provided by the owning MPUs. An owning MPU is that MPU which manages all I/O to a particular host LUN (or LDEV, to use the internal name). The MPUs each control a private region of the User Data cache regions on the two MAIN blades and manage their LDEVs within them.

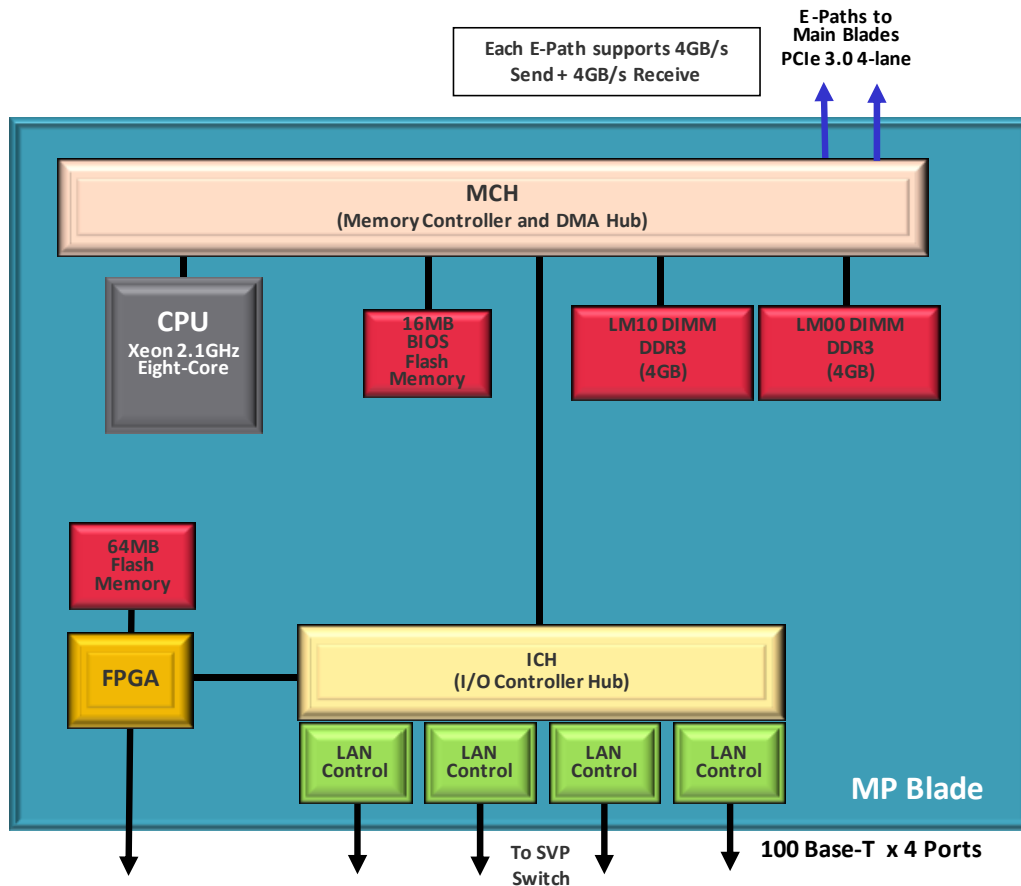
Each MP blade contains a **Xeon 8-core processor** and 8GB of **Local Memory** (RAM). The Intel processor is operated as a logical pair of 4-core processors (MPUs) in order to match up with the system software from the VSP that is used on the logically similar but physically very different HUS VM.

These MPUs execute all of the internal I/O jobs dispatched to them from the MAIN blade LRP processors. It is important to note that user data is never passed through an MP blade. The MPUs only execute I/O requests (identified by up to 64k “tags” per MPU) and software such as HDP or copy products, as well as manage the Control Memory tables.

There are two **Flash Memory modules** on each MP blade. The smaller 16MB flash is the MP blade's "BIOS" that it executes from at power-on time (like a motherboard). The larger 64MB flash is used to hold the updateable system software and current system configuration.

The **FPGA** (Field Programmable Gate Array) processor controls a variety of functions which include: the 'hotline' interface, power supply monitoring functions, the failure interface, and flash memory controller.

Figure 7. MP Blade Details



Each MPU will schedule an I/O process that depends upon the nature of the job it is executing. A process will be of one of the following types: **Target**, **External (virtualization)**, **BED**, **Replication Initiator**, **Replication Target**. In addition there will be system software such as HDP, HDT and housekeeping utilities executed by the MPUs.

- **Target (FE)** process – manages host requests to and from an FE module for a particular LDEV.
- **External (virtualization)** process – manages requests to or from an FE port used in virtualization mode (to external storage). Here the FE port is operated as if it were a host port in order to interoperate with the external system.
- **Disk (BE)** process – manages the staging or destaging of data or parity blocks between cache (User Data regions and the BE DXBF regions) to internal drives via a BE module. The DRR processors within the ASICs process parity for writes.

- **Replication Initiator (MCU)** process – manages the “respond” side of a Hitachi Universal Replicator or TrueCopy Synch connection on an FC port.
- **Replication Target (RCU)** process – manages the “pull” side of a remote copy connection on an FC port.

Each MPU sends commands through the HM ASIC LRP processors to reach the FE and BE modules. The MPUs do not directly communicate with hosts, internal or external drives, or access the user data in cache. One very simplistic manner of visualizing this separation of responsibilities is that the MPU is a “server” running threads, the FE module is a host port card, and the BE module is a drive controller card, and the FEs and BEs perform DMA transfers of data to server memory.

Each MP blade has 8GB of local RAM that is partitioned into nine workspaces:

- private workspaces for processor core-1 through core-8
- a common Control Memory region for lookups and updates for the currently managed LDEVs on the two MPUs

All metadata associated with those LDEVs managed by a particular MPU is located in this partial copy of the global Control Memory region. All lookups occur in here at very high speed. All updates to the metadata for these LDEVs *write through* local memory over I-path links to the master (and mirror) copy of Control Memory located in cache (in the CMG0 region). Since only one MPU controls all I/O to an LDEV, there are no costly cross-MPU coherency locks.

A more detailed discussion of the Control Memory, DXBFs, User Data cache, CLPRs, and MPU partitions is found in the ***Cache Memory and Local Memory*** section below.

LDEV Ownership by MPU

Each of the four MPUs is assigned a specific set of LDEVs to manage. There is a 16,384 system limit on the number of LDEVs that may be established. [Note: see Appendix 2 for a discussion of VDEVs, LDEVs and related details.] When new LDEVs are created, they are round-robin assigned across the MPUs using a pattern of MPU-10, MPU-20, MPU-11, and then MPU-21. The two MPUs on Cluster-1 are named MPU-10 and MPU-11. Those on Cluster-2 are named MPU-20 and MPU-21.

The individual LDEV associations to MPU can be looked up and manually changed either by Storage Navigator or by a script that uses CLI commands from the ***raidcom*** utility. There is no automatic load balancing mechanism to move “hot” LDEVs around among the MPUs in order to even out the processing loads. It is no longer necessary to keep every LDEV from the same Parity Group assigned to the same MPU. [This is also true now for the VSP with system software 4.0 and later.]

An MPU will accept all I/O requests for an LDEV it owns without regard for which FE port received that host request, or which BE modules will perform the physical disk operations. Each MPU interacts with all installed FE and BE modules in the system (but through the proper LRPs on their associated MAIN blade ASIC).

Each MAIN HM ASIC maintains a local copy of the LDEV-to-MPU mapping tables in order to know which MPU owns which LDEV. No other MPUs will ever be involved in these I/O operations unless that MPU fails. Should an MP blade fail, all of its assigned LDEVs are temporarily reassigned to the two MPUs on the other MP blade. Those MPUs will first have to read all of the necessary LDEV information from the master control tables maintained in the Control Memory region. Upon replacement of the failed MP blade, the reverse will occur, and the original LDEV management list will revert to the new MPUs. They will first have to read the appropriate sections of Control Memory before starting. As a rule of thumb, an

MPU should be kept below 70% busy to manage host latencies, and only 35% busy if providing for processor headroom to maintain host performance in case of a failure of the other MP blade. Note: higher peak utilization during batch operations is fine as long as the average utilization during the batch window is below 40%.

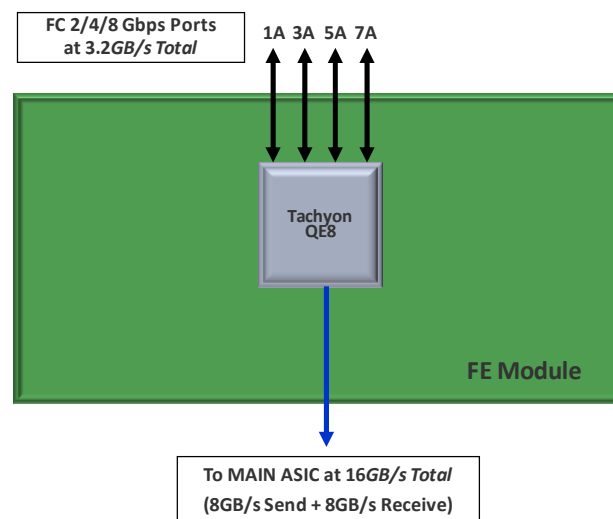
Front-end Module: 4 x FC 8Gbps Ports

Overview

The **Front-end connectivity modules** provide the fibre channel ports for connections to hosts, external storage or remote copy connections (Hitachi Universal Replicator, TrueCopy Sync). These modules are installed into module slots on the rear of the MAIN blade assemblies in the controller. There is one type of module, which has four 8Gbps FC ports that can auto-negotiate down to 4Gbps or 2Gbps rates depending on what the host port requires. This is the same module as used on the HUS 100 family of midrange arrays, except that the port SFP is exchangeable to allow for either short wave or long wave operation.

There can be four FE modules installed per MAIN blade (or six if no BE modules are installed). This provides up to 32 FC ports per system (48 ports for a diskless system). The FE module is fairly simple, primarily having a **Tachyon QE-8** processor on the board. The slot that the FE module plugs into on a MAIN blade provides power and a PCIe x8 link to the MAIN blade HM ASIC. All interaction with an FE module is via the statically assigned LRP processor in the attached HM ASIC. The Tachyon has four DMA channels for moving data blocks in or out cache (reads from the User Data regions or writes to the FE DXBF regions)

Figure 8. FE Module



Tachyon Chip

A Tachyon processor (single chip) is used to bridge the Fibre Channel host connection to a usable form for internal use by the storage controller. The FE modules use the QE8 PM8032 Tachyon processor. This high power Tachyon processor provides a variety of functions for the HUS VM, to include:

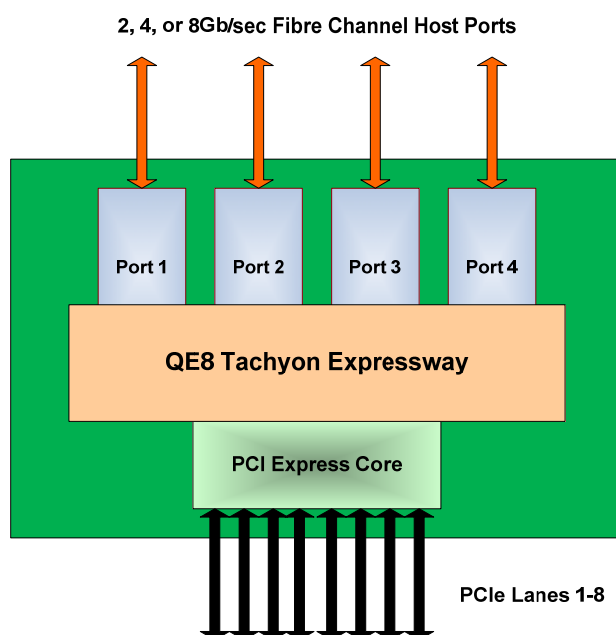
- A conversion of the Fibre Channel transport protocol to the PCIe x8 link for use by one to four controller processors:

- o SCSI initiator and target mode support
 - o Complete Fibre Channel protocol sequence segmentation or reassembly
 - o Conversion to the PCIe link protocol
- Provides simultaneous full duplex operations of each port
- Provides four DMA channels for directly writing blocks to system cache
- Error detection and reporting
- Packet CRC encode/decode offload engine
- Auto-synch to a 2Gbps, 4Gbps or 8Gbps port speed per path

The QE8 processors can provide very high levels of performance as they are directly connected by a high-performance, x8 PCIe link to the HM ASIC. The QE8 processor can drive all four of its 8Gbps ports at full sequential speed. However, for random small block loads, the QE8 controller cannot drive all four ports at full speed (limited by handshaking overhead).

The figure below provides a generalized internal view of the four-port QE8 processor from the PMC Sierra QE8 literature.

Figure 9. QE8 Tachyon Processor Internals



Back-end Module: 8x 6Gbps SAS links

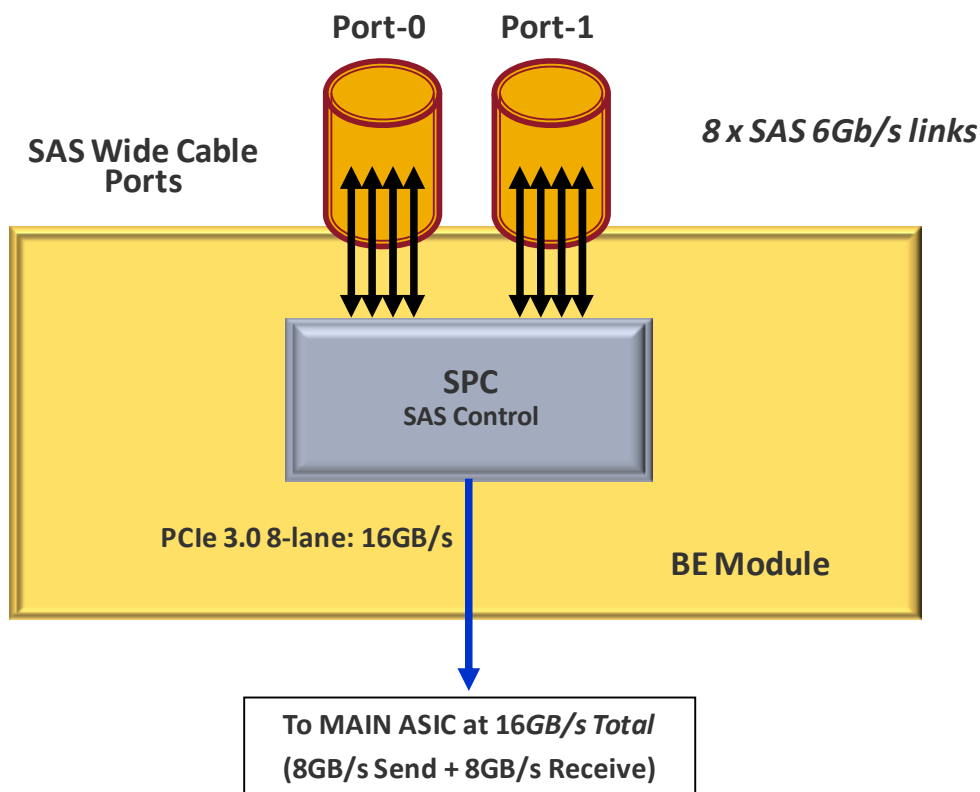
Overview

The **Back-end drive controller modules** provide the SAS links used to attach to the disks in a set of disk trays. These modules are installed into specific slots on the rear of the MAIN blade assemblies in the controller. There is one type of module, and it includes two SAS Wide ports on the rear panel by which a pair of SAS Wide cables to the first row of four disk trays (DB-00 to DB-03) are connected. There are four 6Gbps full duplex SAS links per port, with two ports per module. This is the same module and cable as used on the HUS 100 family of midrange arrays.

Except for a pure virtualization (diskless) configuration, there are two modules installed per MAIN blade, with four modules per system. This provides 32 x 6Gbps full duplex SAS links (over 8 SAS Wide ports) per system to control up to 1152 drives. The BE module is fairly simple, primarily having a powerful **SAS Processor chip (SPC)** on the board. The slot that the BE module plugs into on a MAIN blade provides power and an x8 PCIe link to the MAIN blade HM ASIC. All interaction with the HUS VM system from the BE module is via the predetermined LRP processor inside the HM ASIC.

The BE modules control all direct interaction to the drives (SAS and SSD). Each BE module has a powerful dual-core SPC that executes all I/O jobs received from the MPUs (via the LRP processor in that BE's MAIN blade HM ASIC), managing all reading or writing to the drives. Each BE module provides two SAS Wide ports on the rear panel for the cable attachment to two disk boxes. Each port is a bundle of four independent 6Gbps SAS links. Each port controls up to 288 drives in different "stack" of disk boxes (refer back to Figure 4 for details).

Figure 10. BE Module



SAS Processor Chip (SPC)

Each SPC is a high performance dual-core CPU that has the ability to directly control SAS or SSD drives (actually, an SSD is a SAS drive) over its eight full-duplex 6Gbps SAS links. The SPC has multiple DMA channels for moving data blocks in or out of cache (the User Data regions or BE DXBF temporary regions). The SPC works in conjunction with its MAIN blade's HM ASIC using certain LRP and DRR processors.

The back-end functions managed by the SPC include the following:

- control all reads or writes to the attached drives
- execute all jobs received from an MPU processor via the MAIN LRP
- use internal DMA channels to move data or parity blocks in or out of the BE DXBF or User Data regions
- tracking of each drive's status and alert an MPU if a problem occurs or is developing

The SPC creates a direct connection with one drive per link via a disk box's (DB) internal ENC SAS switch when the SPC needs to communicate with a drive. The speed at which a link is driven is 6Gbps as all drives available for the HUS VM are of the 6Gbps variety.

Should a link fail within a port, the SPC will no longer use it, failing over to the other 3 links in that port.

Cache Memory and Local Memory

The HUS VM has two types of **physical** memory systems, which include:

- **Cache Memory (CM)** – the primary cache space located on the MAIN blades of up to 256GB (if using 16 x 16GB DIMMs) which is used for several purposes. These 16 DIMMs are organized into two groups per MAIN blade, each having a row of 4 DIMMs. These groups of four DIMMs are called *Cache Memory Groups*, with the lower row (per blade) being **CMG0** and the upper being **CMG1**. Both an 8GB DIMM and a 16GB DIMM are supported at this time, each type being installed as a set of four per MAIN blade. Each CMG must use the same type of DIMM. This is shown in Figure 11 below.
- **Local Memory (LM)** on each MP blade. This was illustrated back in Figure 7.

The **CM system** holds all control information, directories, and user data in different cache regions within CM. The **LM system** provides a local copy of the Control Memory tables relating to those LDEVs managed by the two MPUs on the MP blade, as well as the workspace for each of the 8 cores. Some or all of CM will be backed up in the event of a system shutdown or power loss, but the LM regions are not as they are temporary space.

Cache Memory Overview

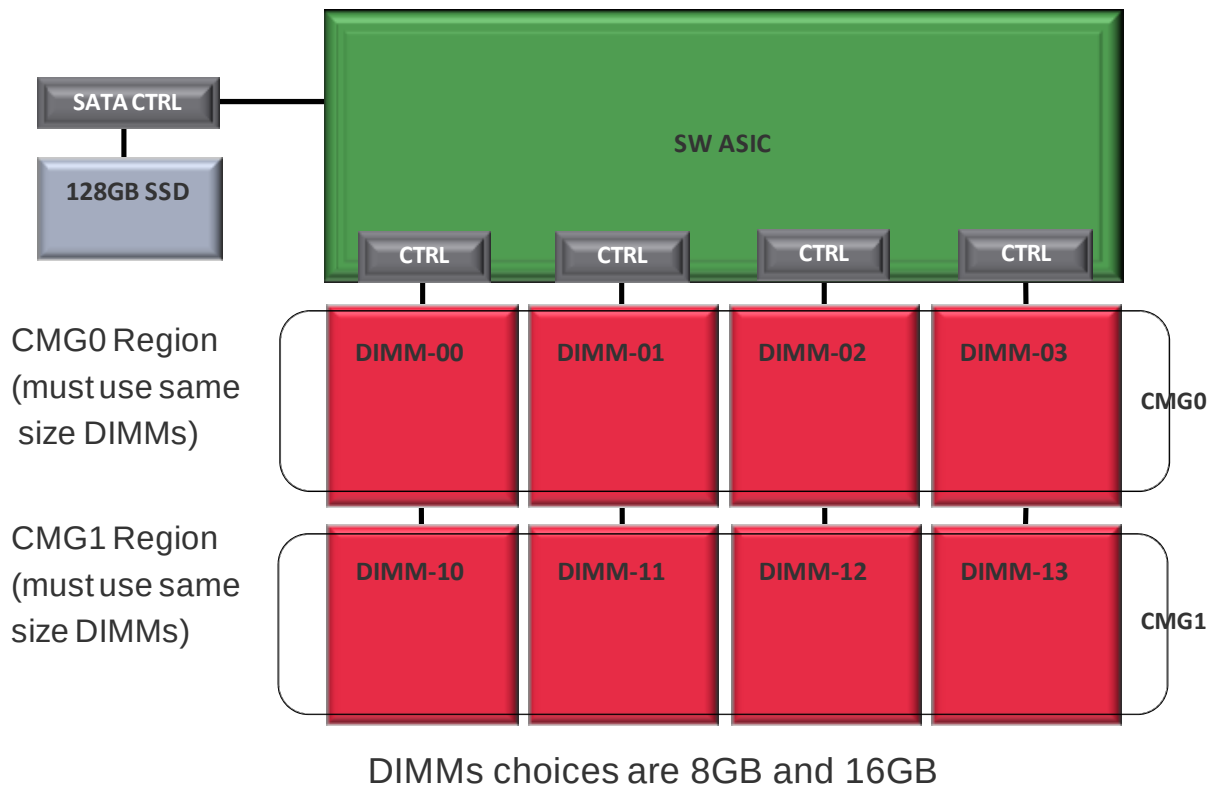
The HUS VM uses a dynamic, partitioned cache allocation system in order to provide for the User Data cache space for each MPU that they use for managing the data for their discrete set of LDEVs. Each MPU has a set of LDEVs for which it controls all I/O operations. This distribution of LDEV ownership is normally fairly uniform across the four MPUs in the system.

Cache space is managed in the same manner as on the VSP, with 64KB segments mapped by an MPU for host blocks for a specific LDEV. Each MPU maintains its own **Write Pending** queue which operates on its current set of dirty write segments.

The top level management of the cache system is the **Global Free List** that manages all of the 64KB cache segments that make up the User Data cache region. Each MPU also operates a **Local Free List** (of

those 64KB segments allocated to it from this Global Free List) from which it privately draws and returns segments for individual I/O operations on those LDEVs it manages. If an MPU determines by its recent workloads that it holds excess 64KB segments, some of these 64KB segments in its Local Free List are released back to the system's Global Free List for to be available for another MPU to borrow.

Figure 11. Physical Cache Layout in a MAIN Blade

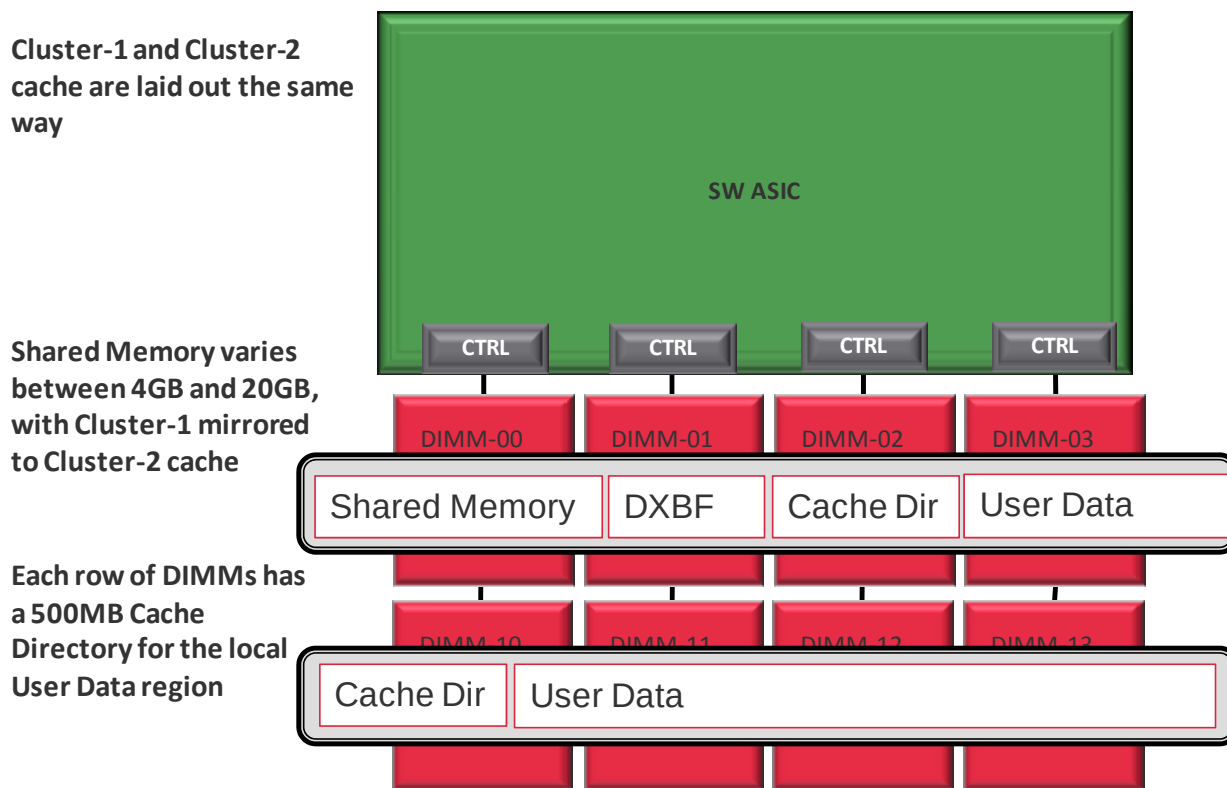


CM Cache Regions

The CM system has five distinct **cache regions**, which are logical structures superimposed over the physical space that are used to manage access to data, metadata, control tables and system software. These regions include:

- **Control Memory regions** for the master global copy of configuration data, metadata and control tables (one per MAIN, where one is a mirror). A reduced copy is kept in Local Memory on each MP blade.
- **User Data Cache Regions** for the MPUs for user data blocks, including the mirrored write blocks and Copy Product data blocks (one or two per MAIN).
- **FE Module DXBF regions** (a temporary FE workspace, one per MAIN)
- **BE Module DXBF regions** (a temporary BE workspace, one per MAIN)
- **Cache Directory regions** that point to 64KB segments in the local CMG's User Data area

Figure 12. Illustration of the Cache Regions



Cache Region Organization

The cache regions on each MAIN blade are organized as described below.

1. The **Control Memory region** occupies a 4-20GB space reserved from the CMG0 rank of DIMMs on each MAIN Blade. Control Memory writes to MAIN-1 are mirrored (over the E-paths using the HM ASIC DMA channels) to MAIN-2, and the reverse is also true. The size of the SM region increases as the system configuration grows and as software features like HDP, HDT, True Copy Sync and HUR are put to use. The Control Memory region contains the following types of information:
 - a. information about the physical system configuration details
 - b. all details regarding the Parity Groups, VDEVs and LDEVs
 - c. tables used to manage external I/O (virtualization) and Replication operations
 - d. Dynamic Provision and Dynamic Tiering control information
2. The **DXBF region** is located in CMG0 and provides the local write buffer region for FE modules and workspace area for the BE modules for that MAIN blade. There is one region used for FE modules and a separate one used for BE modules (and DRR processors) attached to that MAIN blade.
3. The **Cache Directory region** (500MB each) in both CMG0 and CMG1 which contains a mapping table between LDEVs and active cache segments (by MPU) in that CMG's User Data region.

4. The **User Data region** is all cache space not used for the above functions that is set aside for general use by the four MPUs for user data blocks. This entire space is used by the default CLPR-0 (Cache Logical Partition, a cache segment container). It is organized by a Global Free List (for that CLPR) as a collection of all of the 64KB cache segments that comprise it, these being the unit of cache allocation.

The local Control Memory and Cache Directory regions are backed up to a MAIN Blade's local 80GB or 128GB SSD on a planned shutdown. On power loss, all of the cache (which includes all Write Pending segments) per MAIN Blade is backed up to its local SSD.

Cache Logical Partitions (CLPR)

The HUS VM has a base or default CLPR-0 that includes all available User Data region space. Optional CLPRs may be established (by reducing the size of CLPR-0) to manage workloads on certain Parity Groups. [Note: all of the LDEVs from an individual Parity Group must be assigned to the same CLPR, but they could be managed by different MPUs.] All Parity Groups are assigned to CLPR-0 by default. CLPR-0 will use the entire User Data region as a pool of 64KB segments to be allocated to the four MPUs. The space within CLPR-0 (less a small reserve for borrowing) is divided up among the four MPUs.

Figure 13. Cache Sizing with the Default CLPR-0

User Data Region, Cluster-1		User Data Region, Cluster-2	
CMG0, MAIN-1	CMG1 MAIN-1	CMG0, MAIN-2	CMG1 MAIN-2
11.5GB	31.5GB	11.5GB	31.5GB
(CM=20GB)	(CD=.5GB)	(CM=20GB)	(CD=.5GB)
(CD=.5GB)		(CD=.5GB)	

CLPR-0 = 86GB	
(11.5 + 31.5 + 11.5 + 31.5)	

MPU Partitions Within CLPRs

The space within each CLPR (or only CLPR-0 if just using the default) is divided up into private dynamic (not fixed in size) partitions for each of the four MPUs. The maximum initial space per MPU Partition is roughly determined by dividing the total cache space for that CLPR by four. So if CLPR-0 has 85GB, each MPU would nominally have up to 20GB, with a reserve in the Global Free List of perhaps 6GB. (Specifics on this are unknown).

Figure 14. MPU Sub-partitions within CLPR-0

MPU Partitions, CLPR-0	
Global Free List	6GB reserve (?)
MPU-10	20GB
MPU-11	20GB
MPU-20	20GB
MPU-21	20GB

Creating Additional CLPRs

The **Virtual Partition Manager** (VPM) software feature can be used to reduce the size of CLPR-0 and then create one or more additional CLPRs, with up to 29 total CLPRs per HUS VM, using a minimum CLPR size of 4GB with an increment of 2GB per user established CLPR.

When creating additional CLPRs, the original CM User Data region is essentially being split into multiple regions of sizes specified by the user, where that space is only used for those Parity Groups mapped to that CLPR. So cache usage by application or by storage tier (and external LDEVs) may be controlled in this fashion.

Figure 15. Cache Example with Optional CLPR-1

User Data Region, Cluster-1		User Data Region, Cluster-2	
CMG0, MAIN-1 11.5GB (CM=20GB) (CD=.5GB)	CMG1 MAIN-1 31.5GB (CD=.5GB)	CMG0, MAIN-2 11.5GB (CM=20GB) (CD=.5GB)	CMG1 MAIN-2 31.5GB (CD=.5GB)

CLPR-0 = 56GB
CLPR-1 = 30GB

The space in each new CLPR created will always be shared by all four MPUs, with each starting off with a fixed share of that space whether or not they have LDEVs assigned there. There is a minimum footprint per CLPR per MPU that will always remain regardless of usage. The size of the CLPR determines the initially allocated space per MPU and the size of the local reserve space (a type of Global Free List). A 16GB CLPR-1 likely has an 8GB reserved space with 2GB assigned to each MPU (going by VSP CLPR tables). So if only 1 MPU has LDEVs assigned in CLPR-1, the 6GB of space for the other 3 MPUs is locked down and can never be used.

Each CLPR has its own space management and queue management mechanisms, such as a Free Queue, a Clean Queue, and a Dirty Queue. Each MPU maintains a private Write Pending Limit per CLPR.

Figure 16. MPU Sub-partitions within two CLPRs

MPU Partitions, CLPR-0 and CLPR-1			
Global Free List		4GB reserve (?)	
MPU-10	13GB	CLPR-0	
MPU-11	13GB		
MPU-20	13GB		
MPU-21	13GB		
Global Free List		2GB reserve (?)	
MPU-10	7GB	CLPR-1	
MPU-11	7GB		
MPU-20	7GB		
MPU-21	7GB		

Drives and Drive Boxes

There are four types of optional **disk boxes (DBs)**:

- 2U 12-slot FMD box (**DBF**)
- 2U 12-disk LFF SAS box (**DBL**)
- 2U 24-disk SFF SAS box (**DBS**)
- 4U high density 48-disk LFF SAS single drawer (**DBX**)

Figure 17. Map of the BE Module ports to DBs

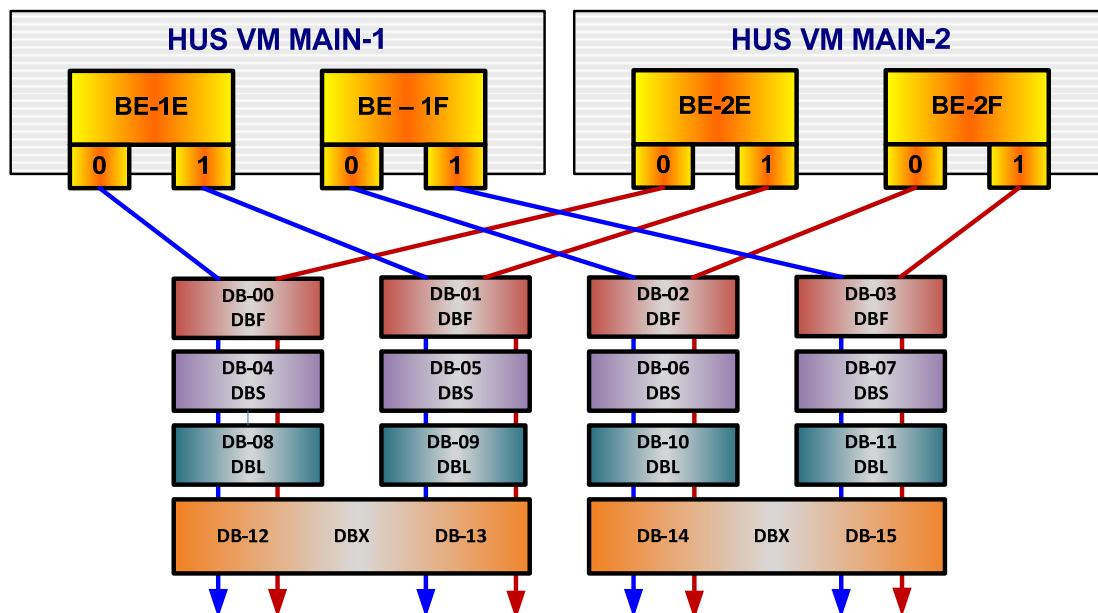
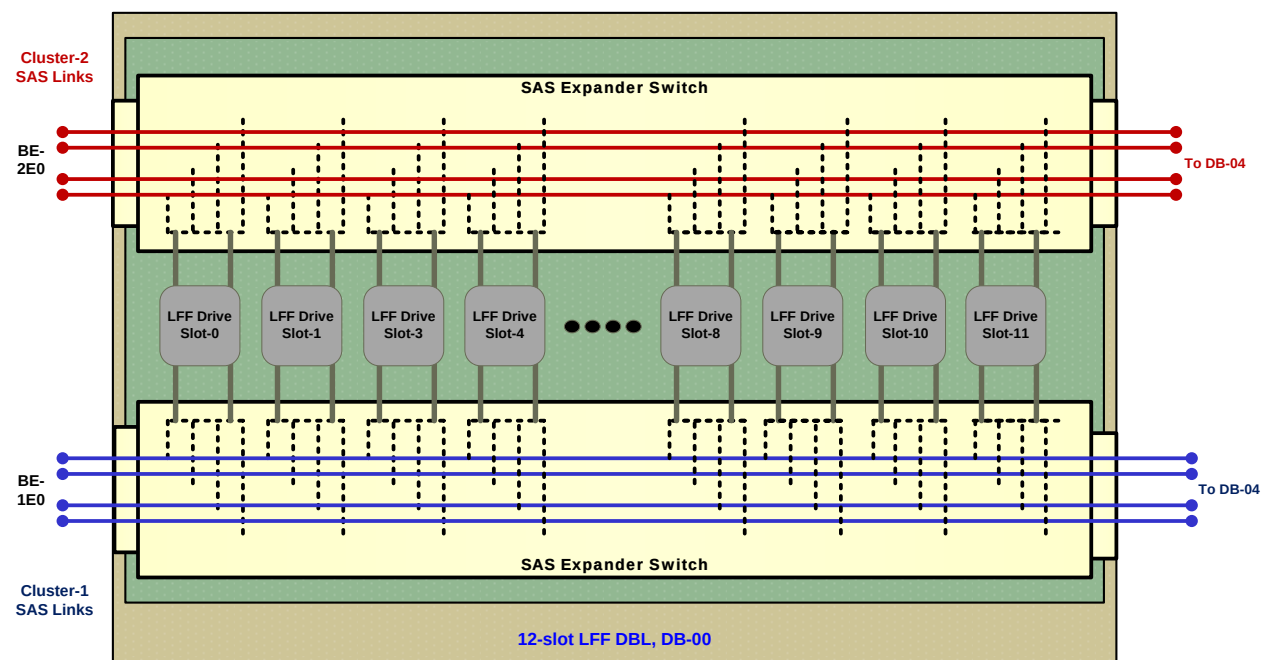


Figure 17 above illustrates the connection of disk boxes to the BE Module ports. Notice how the highest performing drive boxes have been placed closest to the BE Module ports. For the DBF, DBS, or DBL options it is best practice (but not required) that they be installed four at a time to balance the back end connections. Also, it is a best practice (but not required) to locate the drives specified as members of a Parity Group from among these four drive boxes in a row (vertical striping). It will likely be easier to track the configuration of the system (disks to Parity Groups), as well as maintain the same level of latency per disk. Each new row of disk boxes introduces an incremental amount of switch-to-switch latency.

DB: 12-Disk LFF Tray (DBL)

Figure 18 depicts a simplified view of the 2U 12-disk 3.5" SAS box details. Inside each 12-disk box there is a pair of SAS "expander" switches. These may be viewed as two 16 port SAS switches attached to the two SAS wide cables coming from the BE ports, or from the next drive box in the stack in the direction of the controller. The two expanders cross connect the dual-ported disks to all eight of the active 6Gbps SAS links that pass through each box. Any of the available 3.5" SAS drives may be intermixed in these boxes. The drives are arranged as three rows of 4 slots each for drive canisters. All drives within each box are dual attached, with one full duplex drive port going to each switch.

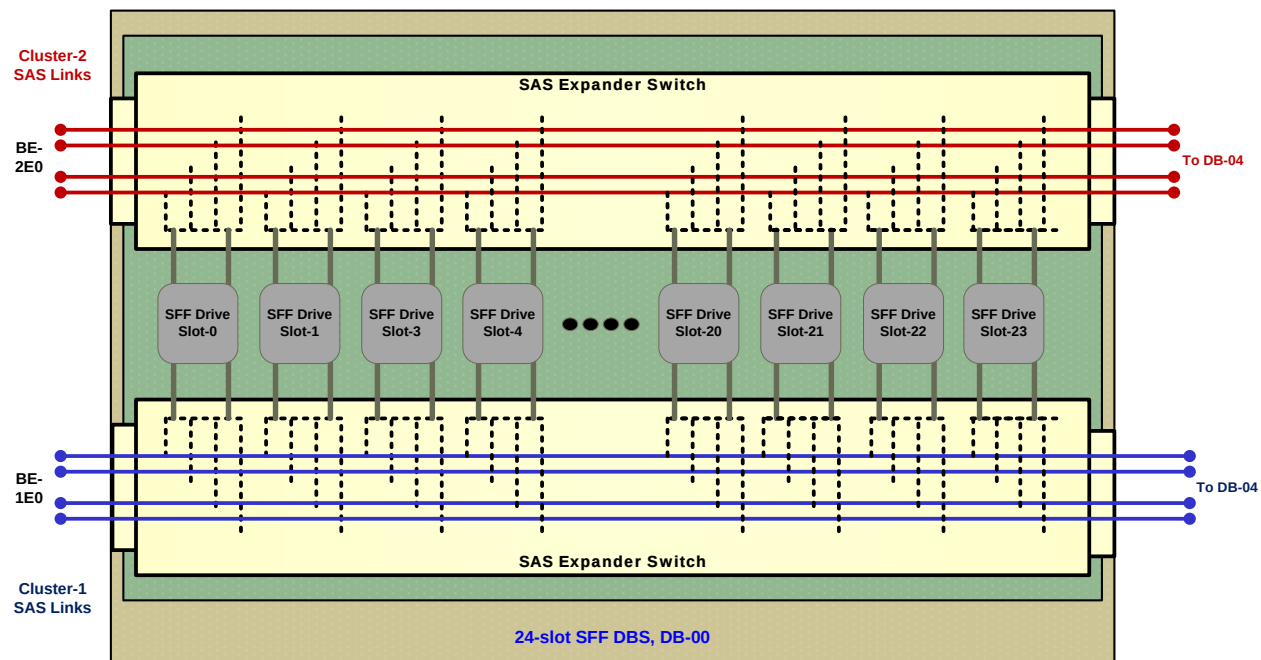
Figure 18. 12-disk DBL box



DB: 24-Disk SFF Tray (DBS)

Figure 19 depicts a simplified view of the 2U 24-disk 2.5" SAS tray details. Inside each 24-disk tray there is a pair of SAS "expander" switches. These may be viewed as two 32 port SAS switches attached to the two SAS wide cables coming from the BE ports, or from the next drive box in the stack in the direction of the controller. The two expanders cross connect the dual-ported disks to all eight of the active 6Gbps SAS links that pass through each box. Any of the 2.5" SAS or SSD drives may be intermixed in these trays. The drives are arranged as one row of 24 slots for drive canisters. All drives within each box are dual attached, with one full duplex drive port going to each switch.

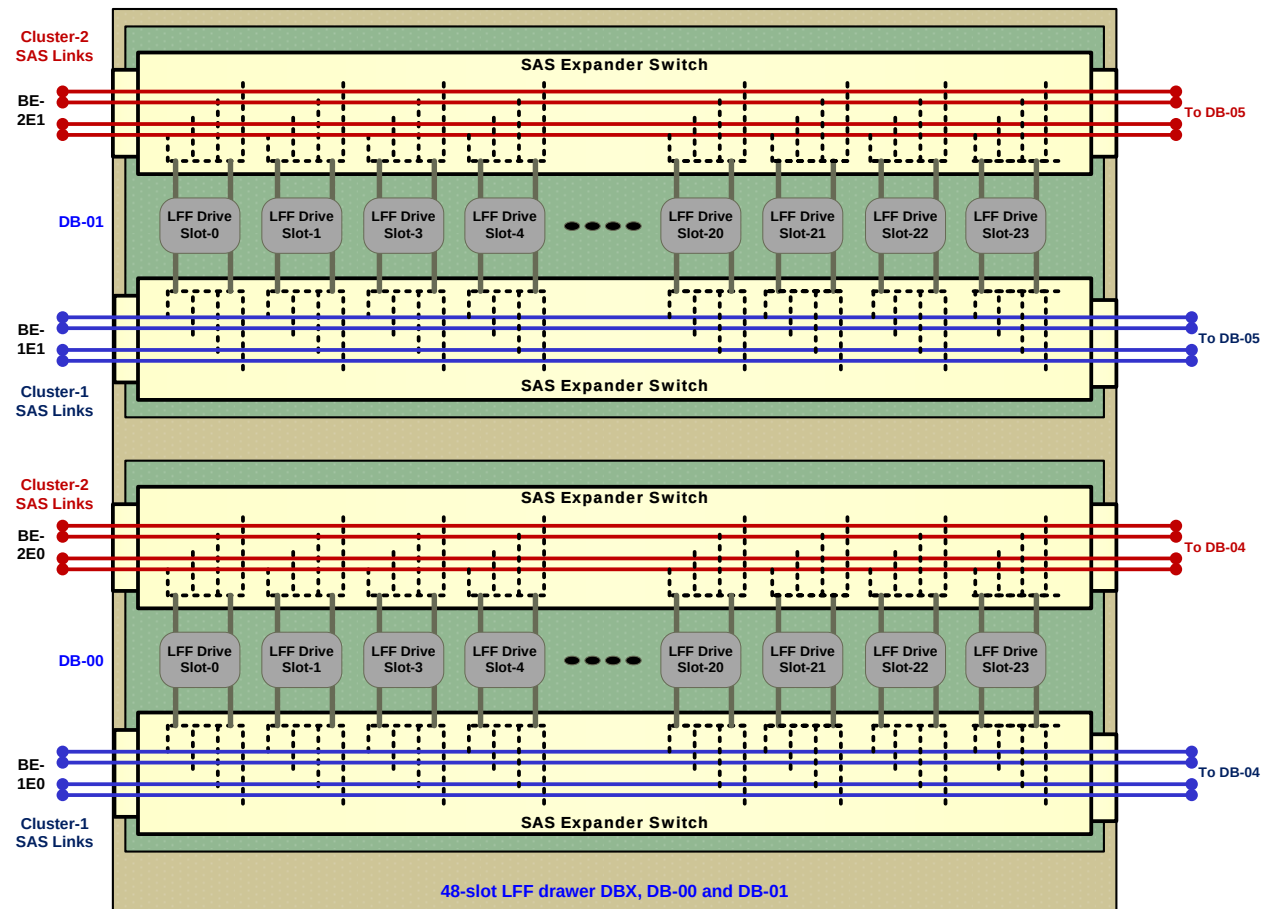
Figure 19. View of the SAS Links, Switches and Drives in an SFF DBS



DB: 48-Disk LFF Drawer (DBX)

The high density 4U LFF disk drawer shown in Figure 20 has 48 3.5" vertical disk slots. This drawer operates as two individual 24-disk boxes, with the disks arranged in four columns (as seen from above) of 12 disks each. The drawer includes four Wide SAS connectors (16 SAS links) for IN ports and four more (16 SAS links) as OUT ports. Each of the 24-disk sub-boxes has two 24-port SAS "expander" switches. There are four SAS links entering each expander, four links passing through to the next drive box, and 24 switched ports connecting to the disks. This drawer is used for the 3.5" 3TB SAS 7.2K RPM drives.

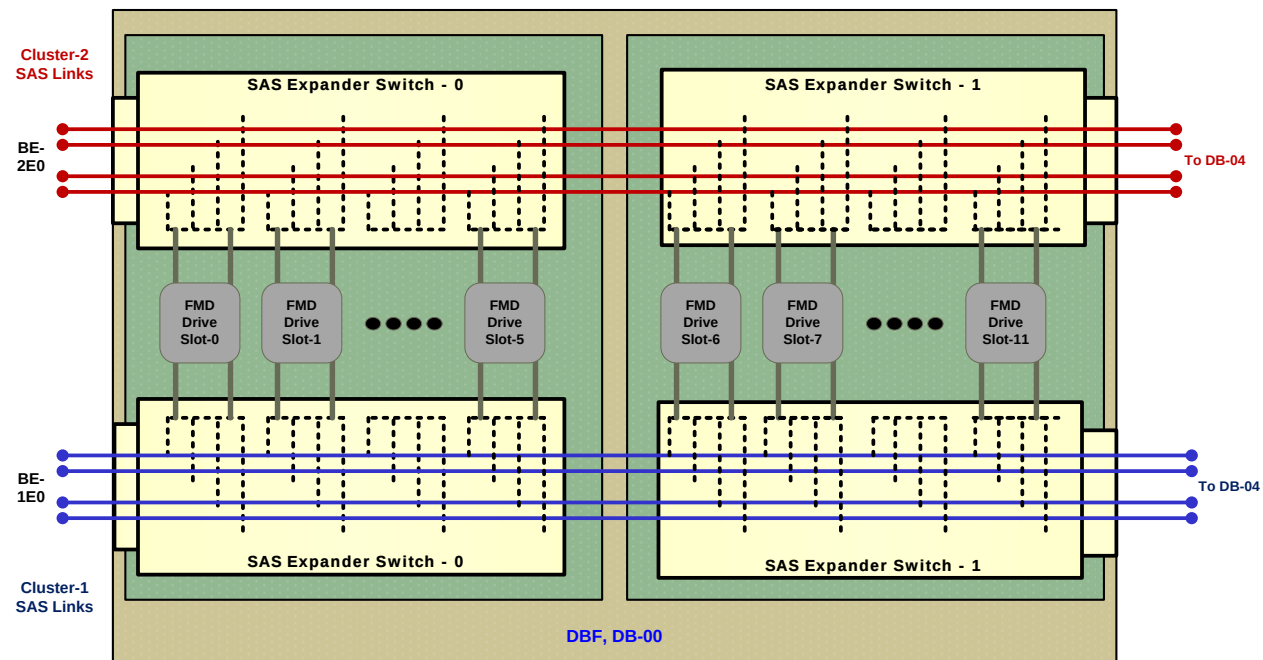
Figure 20. 3.5" 3TB SAS 7.2K RPM 48-Disk Drawer



DB: 12-Disk FMD Tray (DBF)

Figure 21 depicts a simplified view of the 2U 12-slot DBF box details. Inside each 12-FMD module box there are four SAS “expander” switches. These may be viewed as four 20-port SAS switches attached to the two SAS wide cables (8 SAS links) coming from the BE ports, or from the next drive box in the stack in the direction of the controller. The two expanders cross connect the flash modules to all eight of the active 6Gbps SAS links that pass through each box. The flash modules (FMDs) are arranged as three rows of 4 slots each. All FMDs within each box are quad-attached, with two full duplex drive ports going to each switch.

Figure 21. DBF details



Drive Details

The HUS VM has a limit of 1152 drives, including up to 128 SSD drives, 96 FMD drives and spares. Table 3 illustrates the advertised size (decimal) and the raw size (binary, like RAM) after Parity Group formatting of each type of available drive. Also shown is the “rule of thumb” average expected random IOPS rate for each type of drive. **Decimal** means that 1GB is shown as 1,000,000,000 bytes. **Binary** (the number system used in computers) would show 1GB as 1,073,741,824 bytes. On most computers, a 1GB drive would show up as less than 953.7MB just due to this difference in number systems. The 953.7MB value is what most operating systems would report as the size of a raw unformatted device. File system formatting usually cuts this by about 4-5%, so 953.7MB might be about 916MB usable to applications.

Table 3. Table of Supported Drive Types and Rule-of-Thumb IOPS Rates

HDD Type	RPM	Form Factor	Port Speed	Advertised Size (GB)	Raw Size (GB)	Nominal Random Read IOPS
3 TB SAS	7200	LFF	6Gbps	3000	2937	100
1.6 TB FMD	Hitachi MLC	FMD	6Gbps	1600	1759	5000+
900 GB SAS	10k	SFF	6Gbps	900	864	145
600 GB SAS	10k	SFF	6Gbps	600	576	145
400 GB SSD	HGST-MLC	SFF	6Gbps	400	393	5000+
300 GB SAS	15k	SFF	6Gbps	300	288	200
200 GB SSD	HGST-MLC	SFF	6Gbps	200	196	5000+

The type and quantity of drives selected for use in an HUS VM and the RAID levels chosen for those drives will vary according to an analysis of the customer workload mix, cost limits, application performance targets and the usable protected capacity requirements.

FMD Drives

Hitachi's custom FMD drives could be described as turbo-charged SSDs. Each FMD has much higher performance than an SSD due to much higher internal processing power and degree of parallelism. The first drives are 1.6TB, with 3.2TB drives planned for later release. Inside each FMD is a 4-core processor and ASIC (the Advanced Switch Flash Controller - **ASFC**) that controls the internal operations, the four external 6Gbps SAS ports, and the 128 flash memory chips. An SSD drive has one processor and 16 or 32 such flash memory chips. There is a considerable amount of logic in the ASFC that manages the space and how writes are managed. This greatly aids in substantially increasing the write rate for an FMD drive over a standard SSD drive.

SSD Drives (Ian Vogelesang)

SSD drives are the newest drive technology in storage systems. "Drive" is the proper term since these are not "disks" in any sense. SSD drives are more costly than disks in both the "\$ per GB" metric and the "\$ per IOPS" metric, but they do provide much lower read miss response times – the primary reason for using SSDs. The MLC SSD drives currently available for HUS VM are the 200GB and 400GB SFF models.

SSDs are often going to be used with RAID-5 (7D+1P) for read intensive workloads and RAID-10 (2D+2D) for write intensive workloads. For environments that must avoid dual drive failures at all costs, then RAID-6 must be used, as SSD drives do not have higher reliability than do regular disks. Although there are no seek times or rotational delays on SSDs, there is a substantial difference between the use of these three RAID levels with regards to the number of controller cache blocks in use (data and parity) and the extra back-end read and write drive operations. Although these can be sub-millisecond operations, compared to the access times of 5-7ms for SAS drives, the RAID operations still add to the overall BE module and HM ASIC DRR processor loads.

It is with small block random read workloads where SSDs justify their high costs. Keep in mind that the MP blade's microprocessor IOPS capability is the same for FMDs, SSDs and HDDs, and thus system IOPS capability doesn't increase with flash technology, even if that same IOPS can be obtained using far fewer SSDs.

SSDs are far less cost effective for large block sequential transfers compared to HDDs. Large block sequential transfers tend to be limited by host or internal system path bandwidths rather than MPU capability.

SAS Drives (Ian Vogelesang)

SAS drives are the primary drives used now in both enterprise and midrange systems from Hitachi. These are the workhorse drives, having both high performance and good capacities. They fall in between the random performance levels of SSD and SATA (much closer to the SATA end though in comparison to SSD rates). All three drive types will have about the same sequential read performance per matching RAID levels. SAS disks are the same as Fibre Channel disks but with a SAS interface. SAS drives used in the HUS VM come in 15K RPM, 10K RPM, and 7.2K RPM rotation speeds, and in sizes currently ranging from 300GB to 3TB. The SAS interface speeds on all Hitachi supported drives is 6Gbps.

SAS drives are designed for high performance, having dual host ports and large caches. The dual host ports allow four concurrent interactions with the attached BE modules. Both ports can send and receive data in full duplex at the same time, communicating with the DRAM buffer in the drive. However, only

one data transfer at a time can take place internally between the DRAM buffer in the drive and the serial read/write heads.

SAS drives may be low level formatted to native sector sizes of 512 bytes and 520 bytes. Hitachi uses the 520 byte format. All host I/Os to the VSP system are done using the usual 512-byte sectors. See Appendix 5 for a discussion of the new T10 Diff (Data Integrity Field) that replaces the previously used (VSP, USP V, USP) internal check byte. The HUS VM and VSP now pass through and use the host generated T10 field, streamlining the operation. Previously, this 8-byte field was discarded at the FC port and replaced with an internal proprietary version (more overhead per I/O).

It is beyond the scope of this document to talk about drive internals. (If you are interested, see my blog posting on this topic. Navigate to <http://techops.hds.com/>, and then click on Blogs and see the archive from July, 2010.) The drive internally has the same challenges and meets them with analogous mechanisms.

Fibre Channel Port Operations

I/O Request Limits and Queue Depths

Every Open Fibre path between the host and the storage system has a specific maximum instantaneous request capacity, known as the **Maximum I/O Request Limit**. This is the limit to the *aggregate* number of requests (host tags) being directed against the individual LDEVs mapped to a host port. For the HUS VM, the average port limit is about 2048 tags. However, when an FC port is placed into **external storage mode** (it becomes a SCSI *initiator*), the port I/O Request Limit drops to 256 tags in order to control the response times from the external system.

The actual physical port limit is affected by the current request stack for each MPU. Each MP blade has a tag limit of 64k entries for all of the LDEVs managed by its two MPUs from all host ports in use. So it is difficult to state what the actual individual per-port limit might be as this is a dynamic value across the four MPUs. We generally use 2048 tags per port as a rule of thumb to work with.

LUN Queue Depth is the maximum number of outstanding I/O requests (tags) **per LUN** on the **host** port within the server. This is distinctly separate from the system's port **Maximum I/O Request Limit**. A LUN queue depth is normally associated with random I/O workloads and high server thread counts since sequential workloads are usually operating with a very low thread count and large block size.

On the HUS VM, the per-LDEV rule-of-thumb queue depth is 32 per port but can be much higher (no real limit) in the absence of other workloads on that FC port in use for that LDEV. There isn't a queue depth limit per LDEV as such, but the usable limit will depend upon that LDEV's Parity Group's ability to keep up with demand.

In the case of external (virtualized) LUNs, the individual LDEV queue depth per external port (ePort) can be set to 2-to-128 in the **Universal Volume Manager** GUI. Increasing the default queue depth value for external LUNs from 8 up to 32 can often have a very positive effect (especially on response time) on OLTP-like workloads. The actual limit will depend on how many other concurrently active external LUNs there are on that port, as there is a hard limit of 256 tags per external port (ePort) shared among all LDEVs on that path.

External Storage Mode I/O

Each FC port on the FE module may be configured in one of four modes: a **target** mode (accepting host requests), an **external** mode (driving virtualization requests), as a **Replication Initiator (MCU)** mode (respond) or as a **Replication Target (RCU)** mode (pull).

The **external mode** is how other storage systems are attached to FE ports and virtualized. Front-end ports from the external (secondary) system are attached to some number of HUS VM FE ports as if the HUS VM was a Windows server. In essence, those FC ports are operated as though they were a type of back-end port. LUNs from attached systems that are visible on these external ports are then remapped by the HUS VM out through other specified FC ports that are attached to hosts. Each such external LUN will become an internal HUS VM LDEV that is managed by one of the four MPUs.

As I/O requests arrive over host-attached ports on the HUS VM for an external LUN, the normal routing operations within the FE module occur. The request is managed by the owner MPU in (mostly) the Local Memory working set of the global Control Memory tables, and the data blocks go into that managing MPU's data cache region. But the request is then rerouted to one or more other FC ports (not to BEs) that control the external paths where that LUN is located. The external system processes the request internally as though it were talking to a server instead of the HUS VM.

The two **Replication MCU** and **RCU** port modes are for use with the Hitachi Universal Replicator and True Copy Sync software that connects two Hitachi enterprise systems together for replication.

Cache Mode Settings

The external port (and all LUNs present on it) may be set to either "Cache Mode =ON" or "OFF" when it is configured. This controls the write behavior and the maximum transfer size. The "ON" setting has cache behavior identical to that of internal LDEVs, with immediate response to the server once the write data has been duplexed into cache, and uses a 256KB transfer size. The "OFF" setting directs the managing MPU to hold off on the write completion message to the server until the data is staged in cache in the *external* system. The "OFF" setting does not change any other cache behavior such as reads to the external LUNs, but it does increase the maximum transfer size to 512KB. However, this change makes a significant difference when writing to slower external storage that presents a risk for high write pending cases to develop. "Cache Mode = ON" should normally not be used when high write pending rates are expected to occur in the external system. In general, the rule of thumb is to use "Cache Mode = OFF".

Other External Mode Effects

Recall from above that the Queue Depth for each external LUN can be modified to be between 2 and 128 in the Universal Volume Manager configuration screens. Increasing the default value from 8 up to 32 (probably the best choice overall), 64, or perhaps even 128 can often have a very positive effect (especially Response Time) on OLTP-like workloads. Along with this, recall that the maximum overall I/O Request Limit for an external port is 256 (not the average of 2048). If cache mode is ON and the external subsystem cannot keep up with the host write rate, then data will start backing up in Write Pending in the HUS VM. Once this reaches the limit, and there is no more available space in WP, then host I/O will be held off until there is space in Write Pending.

The external port represents itself as a Windows server to the external system. Therefore, when configuring the *host type* on the ports on the virtualized storage system, it must be set to "Windows mode".

General Storage Concepts

Understand Your Customer's Environment

Before recommending a storage design for customers, it is important to know how the product will address the customer's specific business needs. The factors that must be taken into consideration include: **Capacity, Performance, Reliability, Features and Cost** with respect to the storage infrastructure component. The more data you have, the easier it is to architect a solution as opposed to just selling another storage unit. The types of storage configured, including the drive types, the number of RAID Groups and their RAID levels, as well as the number and type of host paths is important to the solution.

RAID Levels and Write Penalties

The HUS VM system currently supports RAID levels 5, 6, and 10. RAID-5 is the most space efficient of these three RAID levels.

RAID-5 is a group of drives (RAID Group or Parity Group) with the space of one drive used for the rotating parity chunk per RAID stripe (row of chunks across the set of drives). If using a 7D+1P configuration (7 data drives, 1 parity drive), then you get 87.5% capacity utilization for user data blocks out of that RAID Group.

RAID-6 is RAID-5 with a second parity drive for a second unique parity block. The second parity block includes all of the data chunks plus the first parity chunk for that row. This would be indicated as a 6D+2P construction (75% capacity utilization) if using 8 drives, or 14D+2P if using 16 drives (87.5% capacity utilization as with RAID-5 7D+1P). Note that in this case we use the term "parity" for the Q block in RAID-6, even though the calculation of the Q block is much more complicated mathematically.

RAID-10 is a mirroring and striping mechanism. First, individual pairs of drives are placed into a mirror state. Then two of these pairs are used in a simple RAID-0 stripe. As there are four drives in the RAID Group, this would be represented as RAID-10 (2D+2D) and have 50% capacity utilization.

The HUS VM, like the VSP and earlier Hitachi enterprise subsystems, has a RAID-10 Parity Group type called 4D+4D. Although we use the term 4+4, in the HUS VM, a 4+4 is actually a set of two 2+2 parity groups that are RAID stripe interleaved on the same 8 drives. Thus the maximum size LDEV that you can create on a 2+2 is the same as the maximum size LDEV that you can create on a 4+4, although on the 4+4 you can create two of them, one on each 2+2 interleave.

RAID-10 is not the same as RAID-0+1, although usage by many would lead one to think this is the case. RAID-0+1 is a RAID-0 stripe of N-disks mirrored to another RAID-0 stripe of N-disks. This would also be shown as 2D+2D for a 4-disk construction. However, if one drive fails, that RAID-0 stripe also fails, and the mirror then fails, leaving a user with a single unprotected RAID-0 group. In the case of real RAID-10, one drive of each mirror pair would have to fail before getting into this same unprotected state.

The factors in determining which RAID level to use are cost, reliability, and performance. Table 3 shows the major benefits and disadvantage of each RAID type. Each type provides its own unique set of benefits so a clear understanding of your customer's requirements is crucial in this decision.

Table 4. RAID Level Comparison

	RAID-10	RAID-5	RAID-6
Description	Data Striping and Mirroring	Data Striping with distributed parity	Data Striping with two distributed parities
Number of drives	4/8	4/8	8, 16
Benefit	Highest performance with data redundancy; higher write IOPS per Parity Group than with similar RAID-5.	The best balance of cost, reliability, and performance.	Balance of cost, with extreme emphasis on reliability
Disadvantages	Higher cost per number of physical drives	Performance penalty for high percentage of Random Writes	Performance penalty for all writes

Another characteristic of RAID is the idea of “write penalty”. Each type of RAID has a different back end physical drive I/O cost, determined by the mechanism of that RAID level. The table below illustrates the trade-offs between the various RAID levels for write operations. There are additional physical drive reads and writes for every application write due to the use of mirrors or XOR parity.

Note that larger drives are usually deployed with RAID-6 to protect against a second drive failure within the Parity Group during the lengthy drive rebuild of a failed drive.

Table 5. RAID Write Penalties

SAS / SSD	HDD IOPS per Host Read	HDD IOPS per Host Write
RAID-10	1	2
RAID-5	1	4
RAID-6	1	6

Table 6. Break Down of RAID Level Write Costs

	IOPS Consumed per host I/O request
RAID-10	1 Data Write, 1 mirrored Data Write
RAID-5	2 reads (1 data, 1 parity), 2 writes (1 data, 1 parity)
RAID-6	3 reads (1 data, 2 parity), 3 writes (1 data, 2 parity)

Parity Groups and system Groups

On Hitachi enterprise products the terms **Parity Group** and **Array Group** are used when configuring storage. The Array Group is the set of four associated drives, while the Parity Group is the RAID level applied to either 1 or 2 array groups. However, in general use in the field, the terms Array Group and Parity Group are often used interchangeably.

RAID Chunks and Stripes

The Parity Group is a logical mechanism that has two basic elements: a virtual block size from each drive (a **chunk**) and a row of chunks across the group (the RAID **stripe**). The cache slot size is 256KB on the VSP and HUS VM, but two such contiguous cache slots per drive comprise a **512KB** chunk. The stripe size is the sum of the chunk sizes across a RAID Group. This only counts the “data” chunks and not any mirror or parity space. Therefore, on a RAID-6 group created as 6D+2P (eight drives), the stripe size would be 3072MB (512KB x 6).

Note that some industry usage replaces *chunk* with “stripe size”, “stripe depth”, or “interleave factor”, and *stripe size* with “stripe width”, “row width” or “row size”.

Note that on all current RAID systems, the chunk is a primarily unit of protection and layout management: either the parity or mirror mechanism. I/O is not performed on a chunk basis as is commonly thought. On Open Systems, the entire space presented by a LUN is a contiguous span of 512 byte blocks, known as the Logical Block Address range (LBA). The host application makes I/O requests using some native request size (such as a file system block size), and this is passed down to the storage as a unique I/O request. The request has the starting address (of a 512 byte block) and a length (such as the file system 8KB block size). The storage system will locate that address within that LUN to a particular drive address, and read or write only that amount of data – not that entire chunk. Also note that this request could require two drives to satisfy if 2KB of the block lies on one chunk and 6KB on the next one in the stripe.

Because of the variations of file system formatting and such, there is no way to determine where a particular block may lie on the raw space presented by a volume. A file system will create a variety of metadata in a quantity and distribution pattern that is related to the size of that volume. Most file systems also typically scatter writes around within the LBA range – an outdated hold-over from long ago when file systems wanted to avoid a common problem of the appearance of bad sectors or tracks on drives. What this means is that attempts to align application block sizes with RAID chunk sizes is a pointless exercise.

The one alignment issue that should be noted is in the case of host-based Logical Volume Managers. These also have a native “stripe size” that is selectable when creating a logical volume from several physical storage LUNs. In this case, the LVM stripe size should be a multiple of the RAID chunk size due to various interactions between the LVM and the LUNs. One such example is the case of large block sequential I/O. If the LVM stripe size is equal to the RAID chunk size, then a series of requests will be issued to different LUNs for that same I/O, making the request appear to be several random I/O operations to the storage system. This can defeat the system’s sequential detect mechanisms, and turn off sequential prefetch, slowing down these types of operations.

LUNS (host volumes)

On a VSP, when space is carved out of a Parity Group and made into a volume, it is then known as a Logical Device (LDEV). Once the LDEV is mapped to a host port for use by a server, it is known as a LUN (Logical Unit Number) and is assigned a certain World Wide Name if using fibre channel interfaces on the system. On an iSCSI configuration, the LUN gets a name that is associated with an NFS mount point. Note that general usage has turned the term of “LU” into “LUN”.

Number of LUNs per Parity Group

When configuring an HUS VM, one or more LDEVs may be created per Parity Group, but the goal should be to clearly understand what percentage of that group's overall capacity will contain active data. In the case where multiple hosts attempt to simultaneously use LUNs that share the same physical drives in an attempt to fully utilize capacity, seek and rotational latency may be a performance limiting factor. In attempting to maximize utilization, RAID groups should contain both active and less frequently used LUNs. This is true of all physical drives regardless of size, RAID level, and physical characteristics.

Port I/O Request Limits, LUN Queue Depths, and Transfer sizes

There are three aspects of I/O request handling per storage path that need to be understood. At the port level, there are the mechanisms of an **I/O request limit** and a **maximum transfer size**. At the LUN level, there is the mechanism of the **maximum queue depth**.

Port I/O Request Limits

Each server and storage system has a particular **port I/O request limit**, this being the number of total outstanding I/O requests that may be issued to a system **port** (not a LUN) at any point in time. This limit will vary according to the server's Fibre Channel HBA and its driver, as well as the limit supported by the target storage port. As switches serve to aggregate many HBAs to the same storage system port, the operating limit will be the lower of these two points. In the case of a SAN switch being used to attach multiple server ports to a single storage port (fan-in), the limit will most likely be the limit supported by the storage port.

On the HUS VM, the rule of thumb average I/O request limit per FC port is 2048. This means that, at any one time, there can be up to 2048 total active host I/O commands (tags) queued up for the various LUNs visible on that FC port. The actual limit is determined by how many requests (tags) are currently in the queue on each MPU, or group of 4-cores. Each MPU has a limit of 64k tags to use in managing all host I/O requests.

Most environments don't require that all LUNs be active at the same time. As I/O requests are application driven, this must be taken into consideration. Understanding the customer's requirements for performance will dictate how many LUNs should be assigned to a single port.

Port I/O Request Maximum Transfer Size

There is also a host tunable parameter at the driver level that controls the **maximum transfer size** of a single I/O operation. This is usually a per-port setting, but it also might be a per-host setting. The defaults for this can range from fairly small (like 32KB) to something mid-sized (128KB). The maximum is usually 1MB-16MB. It is probably a best practice to set this to the largest setting that is supported on the host. The maximum transfer size that the HUS VM will accept on a host port is 16MB. This value controls how application I/O requests get processed. If one used a large application block size (say 256KB) and the port/LUN default was just 32KB, then each request would be broken down (fragmented) into 8 * 32KB requests. This creates additional overhead in managing the application request. The use of a large maximum transfer size such as 1024KB will often be readily apparent on the performance of the system.

LUN Queue Depth

There is either (or both) a *host per-port limit* or a *host per LUN limit* known as the **maximum queue depth**. This is the limit of how many outstanding I/O requests may be scheduled by a host on an HBA port or for each **LUN** on an HBA port. On a host HBA, these queue depth limits are usually configurable

as a driver tunable, usually on a per-port basis rather than a global setting. On Hitachi storage systems, there is no actual limit per LUN. The usable queue depth per LUN when there are multiple simultaneously active LUNs from the same RAID group limit will be determined by physical I/O capacity of that group of drives.

Mixing Data on the Physical drives

Physical placement of data by Parity Groups (not just the LDEVs from the same Parity Group) is extremely important when the data access patterns differ. Mixing highly “write intensive” data with high “read intensive” data will cause both to suffer performance degradation. This performance degradation will be much greater when using RAID-5 or RAID-6 due to the increase in back end drive operations required for writes. When using SAS drives, there are two physical I/Os required for each random write for RAID-10, four for RAID-5, and six for RAID-6.

Selecting the Proper Disks

In all cases, distributing a workload across a higher number of small-capacity, high-RPM drives will provide better performance in terms of full random access. Even better results can be achieved by distributing the workload over a higher number of small LUNs where the LUNs are the only active LUNs in the Parity Group. When cached data locality is low, multiple small-capacity high-RPM drives should be used.

One must also take into consideration the case where a system that is only partially populated with 15k RPM drives will be able to provide a much higher aggregate level of host IOPS if the same budget is applied to lower cost 10k RPM drives. For instance, if there is a 100% increase in the cost of the 15k drives, then one could install twice as many drives of the 10k variety. The individual I/O will see some increase in response time when using 10k drives, but the total IOPS available will be much higher.

Mixing I/O Profiles on the Physical drives

Mixing large-block sequential I/O with small-block random I/O on the same physical drives can result in poor performance. This applies to both read and write I/O patterns. This problem can occur at the LUN level or RAID Group level. In the first case, with a single large LUN, files with different I/O profiles will result in poor performance due to lack of sequential detect for the large block sequential I/O requests. In the second case, where multiple LUNs share a Parity Group, files having different I/O profiles will result in sequential I/O dominating the drive access time, due to pre-fetching, thereby creating high response times and low IOPS for the small-block random I/O. The resolution to these two issues is to use different LUNs and different Parity Groups for different data types and I/O profiles.

Front end Port Performance and Usage Considerations

The flexibility of the front end ports is such that several types of connections are possible. A couple of port usage points are considered:

- Port fan-in and fan-out
- I/O profile mixes on a port

Host Fan-in and Fan-out

With a SAN, the ability to share a port or ports among several host nodes is possible, as is configuring a single host node to multiple storage ports. Fan-in is one of the great promises of fiber channel based SAN technologies—the sharing of costly storage resources. Several host nodes are fanned *into* a single storage port, or a host node has fanned out to several storage ports.

Fan-in

Host Fan-in refers to the consolidation of many host nodes into one or just a few storage ports (many-to-one). Fan-in has the potential for performance issues by creating a bottleneck at the front end storage port. Having multiple hosts connected to the same storage port does work for environments that have minimal performance requirements. In designing this type of solution, it is important to understand the performance requirements of each host. If each host has either a high IOPS or throughput requirement, it is highly probable that a single 2-gigabit or 4-gigabit FC-AL port will not satisfy their aggregate performance requirements.

Fan-out

Fan out allows a host node to take advantage of several storage ports (and possibly additional port processors) from a single host port (one-to-many). Fan-out has a potential performance benefit for small block random I/O workloads. This allows multiple storage ports (and their queues) to service a smaller number of host ports. Fan-out typically does not benefit environments with high throughput (MB/sec) requirements due the transfer limits of the host bus adapters (HBAs).

Mixing I/O Profiles on a Port

Mixing large-block sequential I/O with small-block random I/O can result in poor performance. This applies to both read and write I/O patterns. This problem can occur at the LUN level or Parity Group level. In the first case, with a single large LUN, files with different I/O profiles will result in poor performance due to lack of sequential detect for the large block sequential I/O requests. In the second case, where multiple LUNs share a RAID Group, files having different I/O profiles will result in sequential I/O dominating the drive access time, due to pre-fetching, thereby creating high response times and low IOPS for the small-block random I/O. The resolution to these two issues is to use different LUNs and different system Groups for different data types and I/O profiles.

Appendix 1. Open-V Volume RAID Mechanisms

This section describes how the Hitachi Open-V volumes function on both the VSP and the HUS VM. The track size (chunk) used for Open-V is 256KB, but pairs of consecutive tracks are always used as explained below.

RAID-5

When using generic RAID-5, one would normally calculate the stripe width (row size) by multiplying the RAID chunk size (or “stripe size”) by 7 (if 7D+1P). In the case of Open-V volumes, a native chunk size of 256KB is used. Hence you would expect a row size of 1,792KB. However, Open-V manages RAID-5 somewhat differently than any other vendor. For RAID-5, there are two adjacent 256KB RAID chunks taken from the *same* drive before switching to the next drive in the row, so the effective chunk size is 512KB. This is illustrated in the two Figures below. Thus, there are twice as many chunks per row as normal. So the RAID-5 (7D+1P) stripe width is 3,584KB (7 x 512KB). The RAID-5 (3D+1P) stripe width is 1,536KB (3 x 512KB).

Figure 1. RAID-5+ (7D+1P) Layout (3,584KB stripe width).

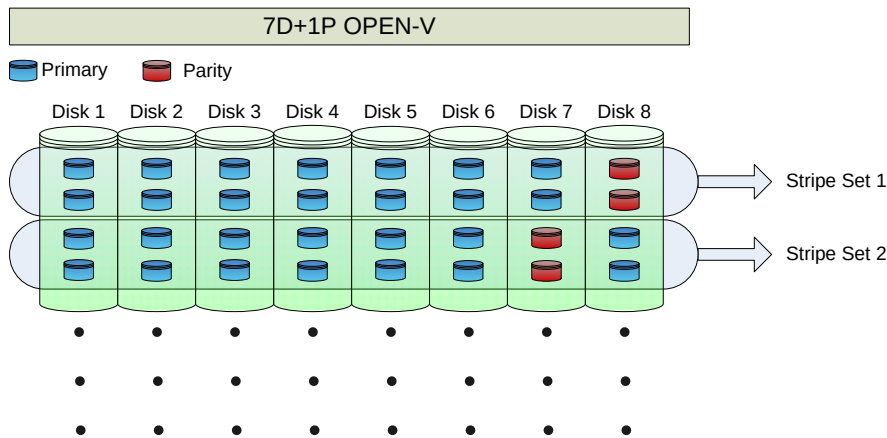
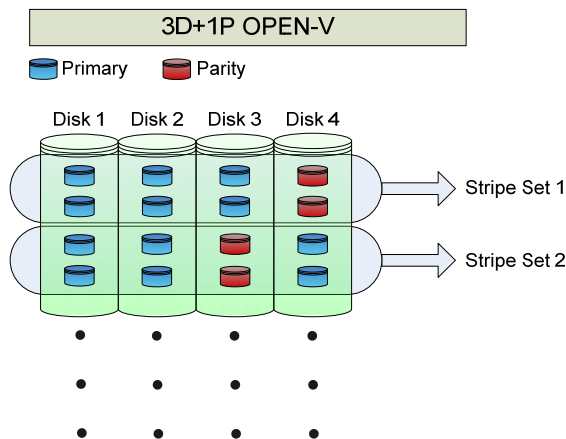


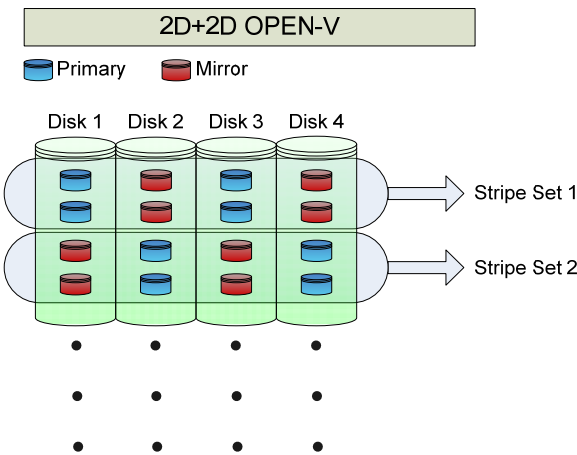
Figure 22. RAID-5+ (3D+1P) Layout (1,536KB stripe width).



RAID-10

Open-V manages RAID-10 in a similar manner as RAID-5, where there are two adjacent RAID chunks taken from the *same* drive before switching to the next drive in the row. This is illustrated in the next Figure. So the Open-V RAID-10 (2D+2D) stripe width is 1,024KB (4 x 512KB).

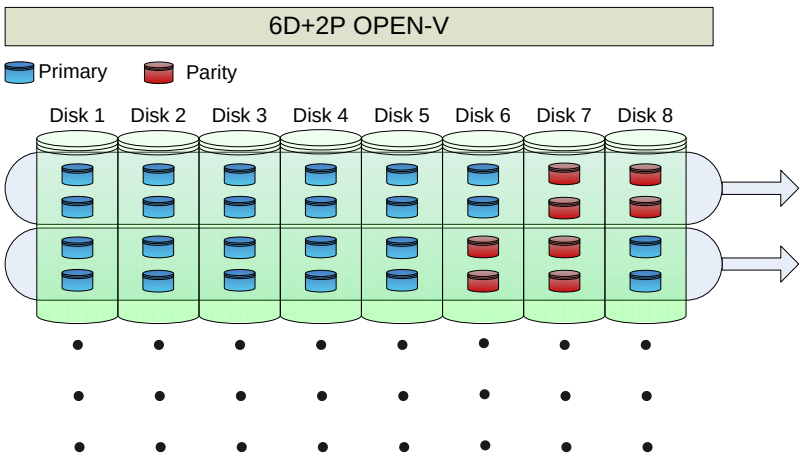
Figure 3. RAID-10+ (2D+2D) Layout (stripe width 1024KB).



RAID-6

Open-V manages RAID-6+ in the same basic manner as RAID-5 but with two parity chunks, where there are two adjacent RAID chunks taken from the *same* drive before switching to the next drive in the row. This is illustrated in the Figure below. So the Open-V RAID-6 (6D+2P) stripe width is 3,072KB (6 x 512KB), and the (14D+2P) is 7,168KB (14 x 512KB).

Figure 4. RAID-6+ (6D+2P) Layout (stripe width 3,072KB)



Appendix 2. HUS VM: LDEVs, LUNs, VDEVs and More

By Ian Vogelesang

This is a generic discussion of ideas and terms that apply to the USP, USP V, VSP, and HUS VM models. An LDEV is a logical volume internal to the system, which can hold host data. LDEVs are “carved” from a logical container called a VDEV. In the diagram below, an internal VDEV is shown, which maps to the formatted space (net of parity) in a Parity Group.

Figure 1. Overview of Logical Devices.

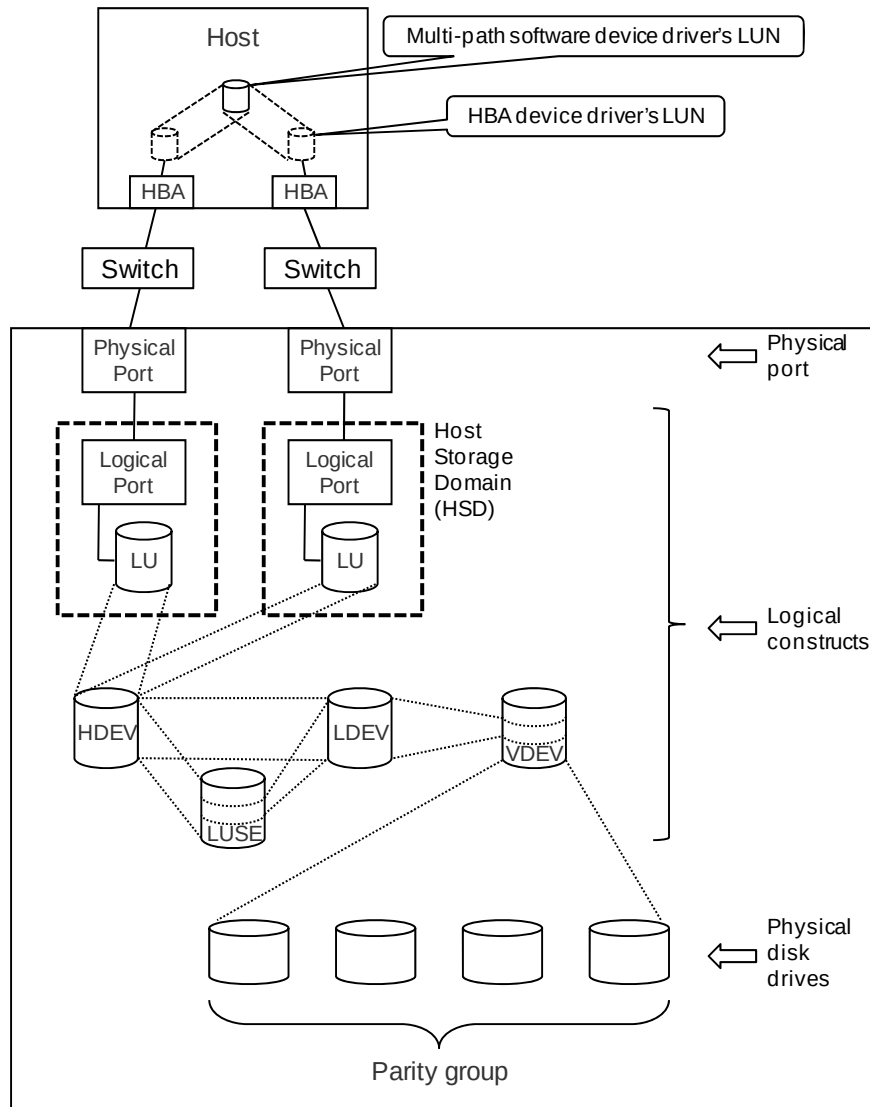
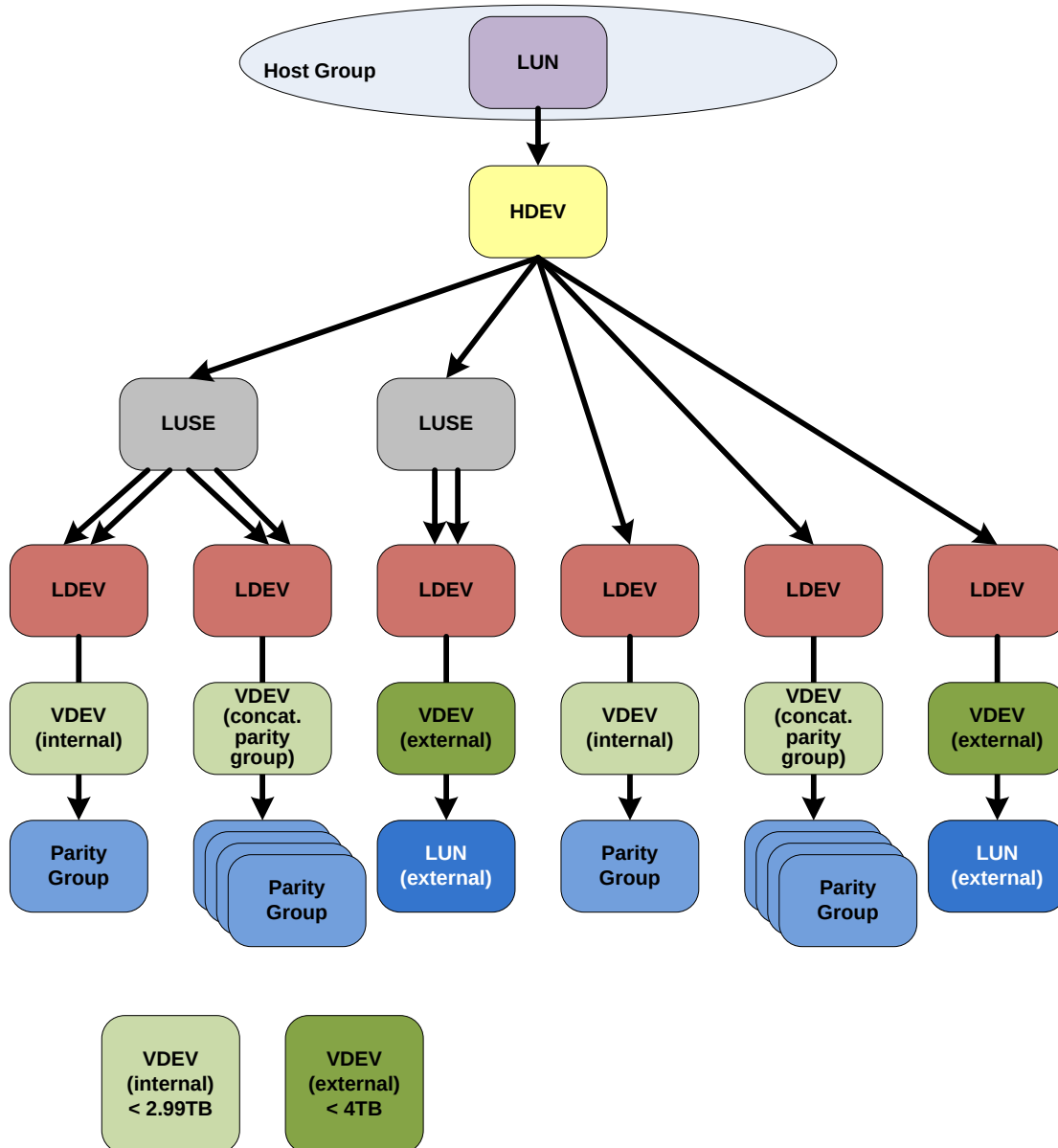


Figure 2. Alternate Overview of Logical Devices



LDEVs are uniquely identified across the system by a six hexadecimal digit identifier usually written with a colon between each pair of hex digits, for example 00:00:01. This format is called the LDKC:CU:LDEV format, and thus the subfields with LDEV names are called the LDKC (Logical DKC) number, the CU (Control Unit) number, and the LDEV number (within the CU).

The maximum number of LDEVs that may be created within an HUS VM system is 16K. [Historical note: The LDKC field, the first 2 hex digits of an LDEV identifier, was added with the introduction of the VSP in order to provide for the ability at a later date to possibly increase the number of LDEVs per system above 64K.]

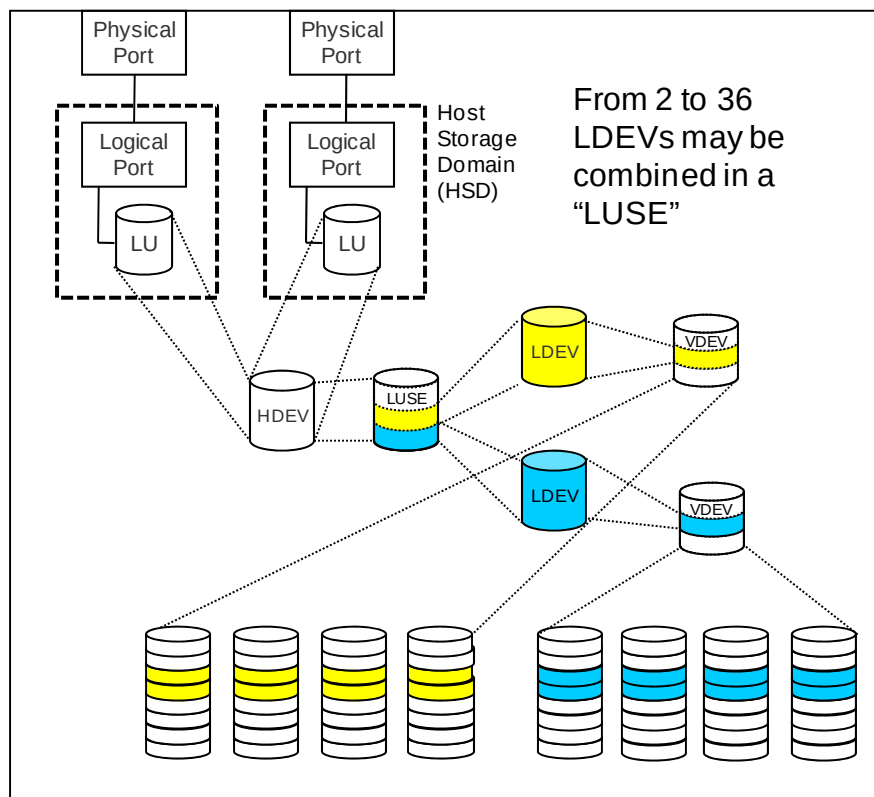
Note that the term “LDEV” as used with HDS enterprise systems (and the HUS VM) to describe an internal logical volume - such as can be carved from a parity group - is somewhat confusingly called an “LU” or a “LUN” for HDS modular systems (AMS and HUS modular family). What is called an LU or LUN in HDS enterprise systems is called an “H-LUN” in modular systems. Sorry for the confusion.

Note that a LUN, the way that a host “sees” an LDEV, emulates a physical drive. As such, accessing a LUN consists of reading or writing drive sectors comprised of 512 bytes. Thus the smallest amount of data that a host can read or write is one whole 512-byte sector. The size of a LUN or an LDEV is therefore a multiple of a 512-byte “block” (sector).

When creating an LDEV, if you specify the size in blocks (512-byte sectors) you will get an LDEV of that exact size. If you specify the size you want using any other unit, such as MB, there will be a rounding operation that will take the size you asked for and will round up to the next higher multiple of an internal allocation unit (logical cylinders of 15 logical tracks). Suffice it to say that if you want to make an LDEV of an exact size, specify the size in blocks (sectors).

Note in the diagram above that LDEVs are not directly presented to hosts as a Logical Unit (LU) or LUN, but rather there is an intermediate construct called an HDEV (Head LDEV). An HDEV may either map directly to a single LDEV, or it may be mapped to what is called a LUSE (Logical Unit Size Expansion). A LUSE is a combination of from 2 to 36 LDEVs:

Figure 3. Overview of LUSE Devices.



The only reason that one should make a LUSE is to create a LUN that is larger than the maximum available LDEV size. For internal LDEVs, the size of an LDEV is limited by the amount of formatted space

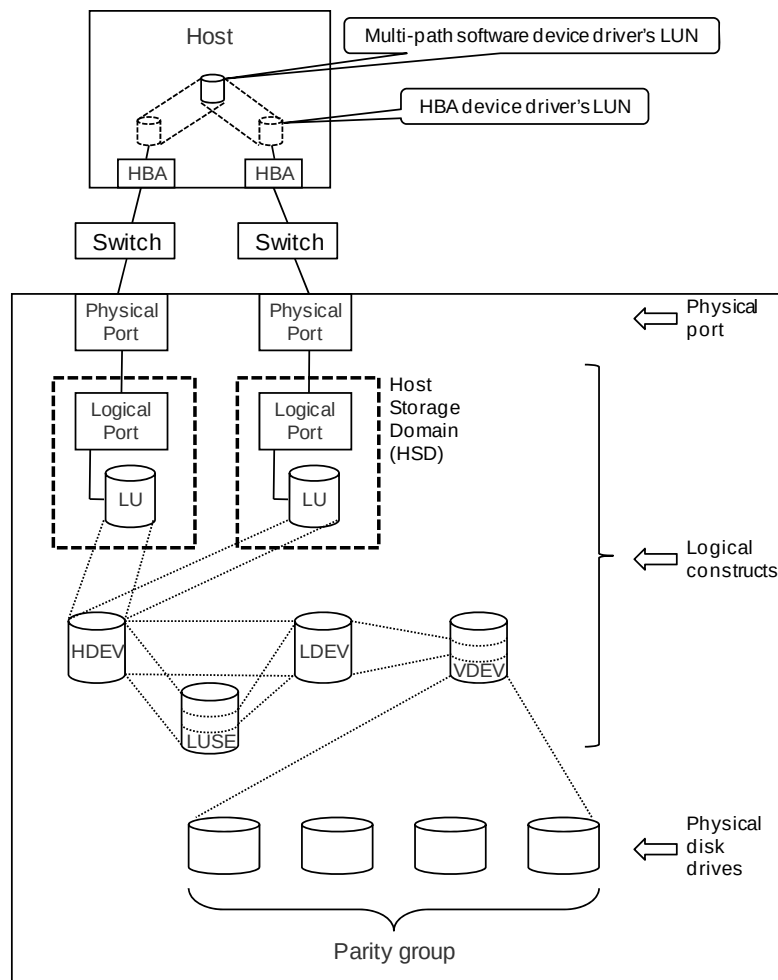
in one parity group, or about 2.99 TB, whichever is smaller (more on this below when we talk about VDEVs). For other types of LDEV, the maximum size is 4 TB.

The maximum size of a LUSE, on the other hand, is 60 TB.

Note: The LDEVs within a LUSE are not interleaved or striped. LBAs (Logical Block Addresses, or sector numbers) for the LUSE start at the beginning of the first, or “head” LDEV, and then proceed to the end of each LDEV before starting on the next one. Thus when or writing sequentially, the performance will be limited by the performance of a single LDEV at a time.

Later on, we will talk about what are confusingly called “concatenated parity groups” which do not give the ability to make larger LDEVs, but which do interleave the RAID stripes of an LDEV across multiple parity groups.

Figure 4. Overview of LUSE Devices.



Going back to our original diagram for a moment, we can see that a single LDEV (system internal logical volume) may be exposed as multiple different LUNS to hosts. The diagram above shows a single host with two paths to an HDEV.

Granting a host access to an HDEV (LDEV or LUSE) is done by creating a Logical Unit (LU) within a Host Storage Domain. This mapping from a host to an HDEV is sometimes called a “path”. A path consists of three parts, the physical port (such as port 1A), the logical port, or Host Storage Domain within that physical port, and then the Logical Unit Number, or LUN within that HSD. Host Storage Domains may be identified by name, or by relative number within the physical port. Note that it is possible to map a particular LDEV as one LU number in one HSD and a different LU number in a different HSD.

From the host’s perspective, the host doesn’t see the two-level structure of the physical port and HSD (logical port). When the HSD is configured, you tell the system which host(s) is to be mapped to each HSD. (A host can only be mapped to one HSD on a given port.) Thus the host “sees” a series of logical units (LUs) behind the port, where each LU is identified by its LU number.

The diagram above also illustrates how multiple paths from a host to an HDEV are implemented. The key is the use of multipath software in the host, such as Hitachi’s HDLM (Hitachi Dynamic Link Manager). The multipath software is a device driver which presents a single logical device for applications to use. But when I/Os are directed to that logical device, the multipath software device driver receives those I/O operations and re-issues them to one or the other of the separate LUNs presented by each HBA’s device driver. The HBA device driver’s LUNs are hidden from applications, and are only used by the multipath driver software. It is only the multipath software that “knows” that the (hidden) LUNs from the two HBAs really lead to the same HDEV inside the system.

Correspondingly at the system end, when a host has multipath software running, the system is not aware of that fact. The system does not “see” I/O operations coming down separate paths as being related in any way. For this reason, the “extended round robin” host multipathing algorithm will produce better results for sequential I/O for active/active configurations. Extended round robin issues a series of I/O operations for given multi-path LUN over one of the paths before issuing the next series of operations over the other path. In this way, the system’s “sequential detection” algorithm which is not aware of multipathing will still see sequential host I/O as sequential at the individual system port.

Recall that we said that LDEVs are “carved” from a logical container called a VDEV. There are two types of visible VDEV on the HUS VM, and internal and an external VDEV. (The DPVOL VDEVs are no longer visible to the user.)

Internal VDEV

The original type was the internal VDEV, which is a logical container that maps to the formatted space within a parity group, net of parity or mirror copies. Since LDEVs are carved from VDEVs, the size of the largest internal LDEV that can be created is one that entirely fills the VDEV that it is carved from.

There is a maximum size for internal VDEVs, which is about 2.999 TB. If there is more formatted space than this in the parity group, then as many maximum size VDEVs as will fit are first allocated, then any remaining space is allocated to one more VDEV.

You can see this if your parity group has more than 3 TB of space in it. In the following example, we have a RAID-5 7+1 parity group consisting of 750 GB drives.



The screenshot shows a storage management interface. On the left, a tree view displays the hierarchy: Subsystem -> Internal -> Box1 -> 1-1 RAID5(7D+1P). Below this, two LDEVs are listed: 1-1-(1) and 1-1-(2). On the right, a table provides details for the first LDEV.

No.	LDKC:CU:LDEV	Emulation	Capacity
1	00:02:00	OPEN-V	3145663.500 MB

The effective capacity of each drive is not quite 750 GB, but just as a rough estimation the total formatted capacity of the parity group is ~0.7 TB per drive * 7 drives’ worth of formatted space = ~4.9 TB. This is bigger than the max internal VDEV size, so you can see above that two VDEVs were created to represent the formatted space on this parity group. Here you can see that the parity group name is 1-1, and the VDEV names on this parity group are “1-1-(1)”, and “1-1-(2)”. That’s what those numbers in parentheses are at the end of a “parity group” name. They are the number of the VDEV within that parity group. Note that the LDEVs are within the VDEV, not the parity group.

While we are at it, parity groups have names like 1-1, 1-2, 3-1, and so on. These names have physical meaning, as they indicate the BED pair and physical drive location on a VSP. However, with Disk Roaming on the HUS VM, where any disk from any tray can be selected as a member of a Parity Group, this name is now nothing more than a label.

Thus the names for internal VDEVs are purely determined by the physical layout of the system.

External VDEV

Each External VDEV maps to a LUN on an external (virtualized) system. Thus each LUN on the external system corresponds to the equivalent of a parity group. It is possible to create an LDEV on an External VDEV that is the exact size in sectors of the external LUN. This makes it handy to do replication or migration.

The names for external VDEVs are in the form of either “E1-1” or “1-1 #”. Both forms of the name are equivalent, thus E 1-1 and 1-1 # both refer to the same external VDEV. In some places one form of the name is used, and in other places the other form is used.

Note that there is a separate name space for each of the four types of VDEV, so that even if you have an internal parity group called 1-1, you can create an external VDEV called E 1-1, and these are not the same.

The maximum size of an external VDEV is 60TB.

Layout of internal VDEVs on the parity group

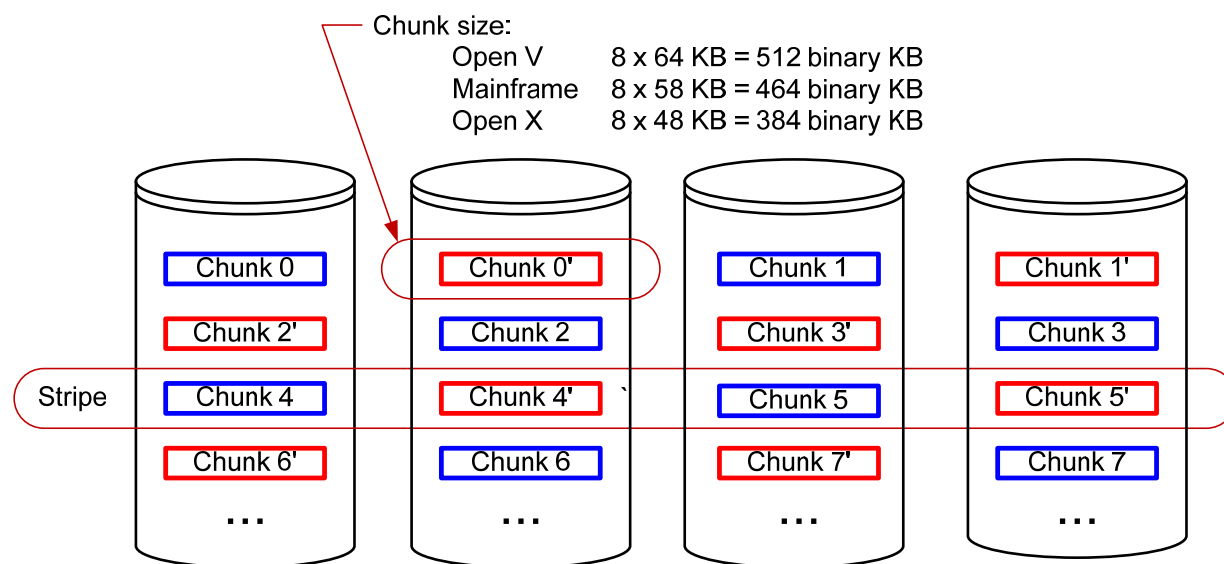
Recall that the VDEV is the logical container from which LDEVs are carved, and that internal VDEVs represent the formatted space within a parity group.

This abstraction of a VDEV as a logical container allows operations that deal with LDEVs to be implemented largely independently of the type of VDEV, and thus LDEVs work the same way whether they are internal, external or a DPVOL.

Thus it's in the internal type of VDEV where the mapping happens to the formatted space within a parity group.

In the next two diagrams, we see the VDEV layout on first RAID-1 (2D+2D) and then RAID-5 (3D+1P) parity group types.

VSP RAID-1 2+2 VDEV chunk layout



Thus the VDEV is mapped to “chunks” on the drives. In the Hitachi Virtual Storage Platform (VSP), the chunk size varies only with the emulation type, and not by the RAID level or number of drives in the parity group.

Note that the chunks are relatively big. About a hundred 4KB blocks fit within one chunk.

For random I/O, only the data requested by the host is read or written. If the host reads a 4 KB block, only 4KB is read from the drive, not an entire chunk.

The time it takes to do a random I/O is composed of over 99% mechanical positioning and less than 1% data transfer. This is because a block size in the 4KB to 8KB range represents on the order of only 1% of a drive physical track size, and mechanical positioning (seek + latency) takes on average more than one full drive rotation.

That's why for random I/O workloads, all we care about are the drive seek times and RPM.

The chunk size is chosen so that the amount of time it takes to transfer a whole chunk (the size of the chunk divided by the read/write head sustained data transfer rate) is of the same order of magnitude as the mechanical positioning time, that is, seek and latency time.

It's designed this way because sequential I/O operations read or write an entire chunk at a time. With having such big chunks, very roughly one half of the drive busy time when reading or writing sequentially is mechanical positioning and one half is data transfer.

This makes sequential transfer relatively productive.

It also makes the I/O service time to read or write a chunk about twice as long in time as a small-block random I/O. The mechanical positioning is the same for both, but for large block sequential I/O after the mechanical positioning is complete it takes roughly one drive rotation worth of time to transfer the chunk.

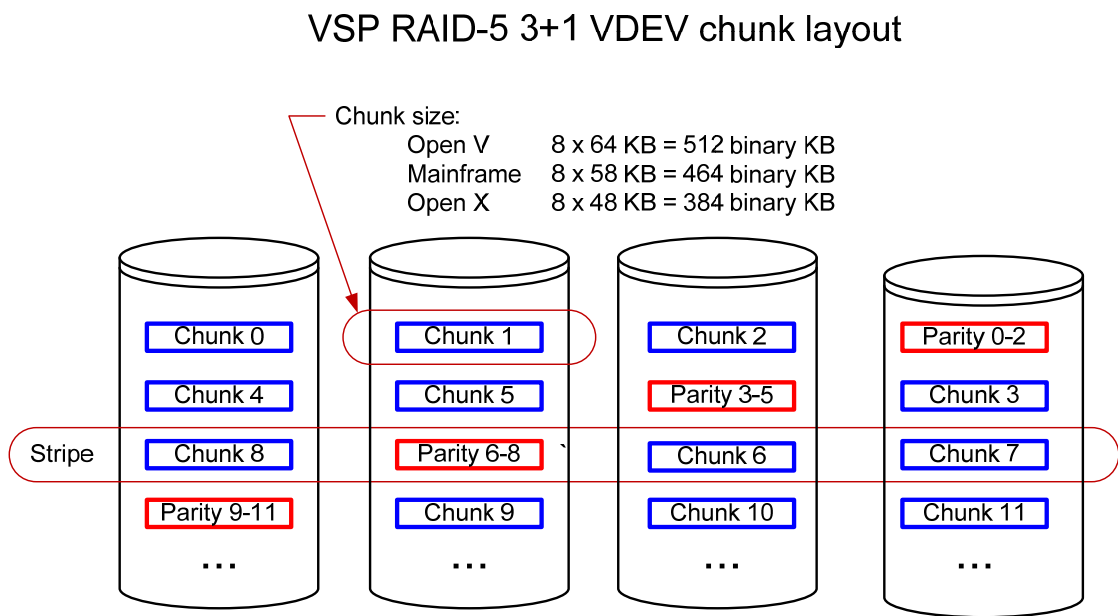
You might ask why not make the chunks even bigger? If transferring a 512KB chunk represents almost ½ of the total I/O service time, then why not make the chunks 1MB? That way you would get a 50% throughput increase for sequential I/O, because then almost 2/3 of the drive busy time during a sequential I/O would be data transfer.

The reason is that if a random read miss occurs while a sequential I/O (pre-fetch stage or write destage) is in progress, the host has to wait. Sequential I/O operations already take about twice as long as random I/Os, and if we made the chunks even bigger the host would have to wait even longer if a random read miss occurs during a sequential I/O. Thus there's a trade-off as a larger chunk size would increase sequential throughput at the expense of adversely impacting random read performance.

Note also in the RAID-1 VDEV chunk layout diagram that the primary chunks (shown colored blue) and the mirror copy chunks (colored red) are in alternating positions from row to row. This is done because the mirror copies are normally “write only” as a backup. All read operations are directed to the primary copy unless the drive containing the primary copy has failed.

Thus the effect of the alternating primary and mirror chunks in the layout from row to row in RAID-1 tends to balance the number of read operations directed to each drive within the parity group.

Let's take a look at the chunk layout for a RAID-5 (3D+1P) VDEV in the HUS VM:



You might ask “Why does the 2nd row have the chunks numbered in the order 4, 5, 3?” This is confusing.” The reason is that sequential reads will read a chunk from each drive in the parity group in parallel. The layout above enables chunks 1,2,3,4 to be read at once. In other words, sequential reads do not read

one parity group stripe at a time. However, for sequential writes one stripe at a time (in this case 3 data blocks and one parity block), again performing one I/O to each drive in the parity group at once.

Note: The HDS AMS 2000 and HUS 100 modular families of systems use a different RAID-5 chunk layout scheme which is more like what you might have expected. The analogous diagram for 3+1 for a modular system has in row 2 from left to right chunks 3, 4, parity 3-5, and chunk 5.

We said that the abstraction of a VDEV as a logical container allows operations that deal with LDEVs to be implemented largely independently of the type of VDEV. One small dependency on the type of VDEV is that LDEVs always start on a RAID stripe boundary. That is, the first chunk of an internal LDEV is always the first data chunk of a RAID stripe, or parity group “row”.

This is done to make it easy to calculate the physical location of where a particular LBA is stored. Each LDEV has an attribute which is the LBA address of where that LDEV starts on every drive in the parity group. Thus you can divide an LDEV LBA by the RAID stripe size (data-only stripe size, exclusive of parity or mirror copies), and the quotient gives you the relative stripe offset in chunks on the drive, relative to where the LDEV starts on each drive, and the remainder tells you where you are within the stripe and therefore which drive contains that LDEV LBA. (The number of the stripe on the drive also tells you which chunks on that stripe are the data chunks and which are the mirror copy / parity chunk(s).)

Appendix 3. Concatenated Parity Groups

By Ian Vogelesang

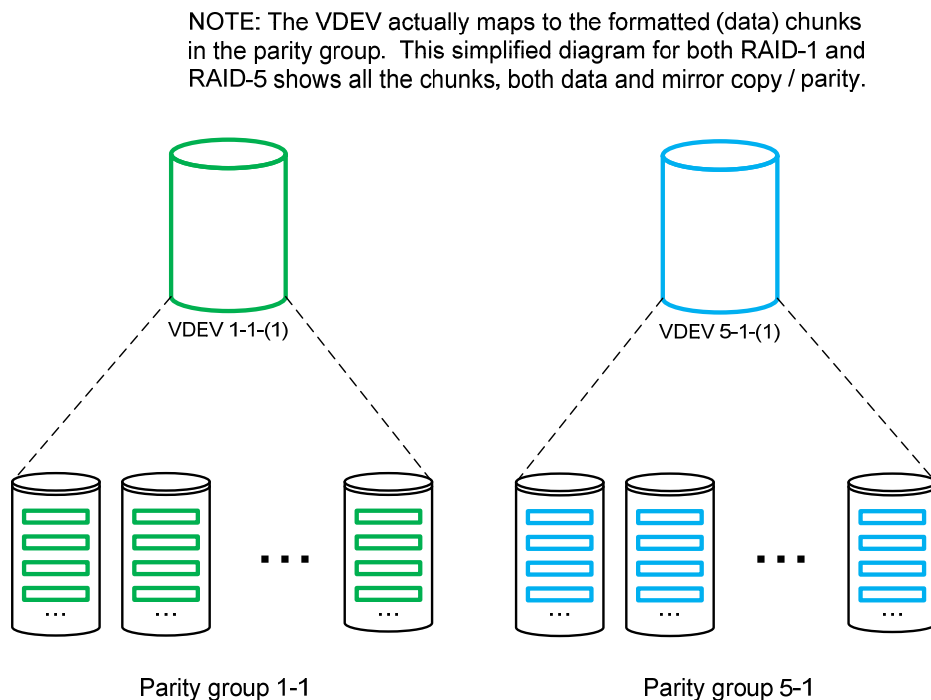
The VSP and HUS VM have the following 5 base Parity Group types:

- RAID-1 (2D+2D)
- RAID-5 (3D+1P)
- RAID-5 (7D+1P)
- RAID-6 (6D+2P)
- RAID-6 (14D+2P)

* Note that there is no “4+4” as a base Parity Group type. “4+4” is created by interleaving a pair of 2+2 Parity Groups.

The following conceptual diagram, showing two Parity Groups that are not concatenated, will be used as the starting point to explain two-way concatenation:

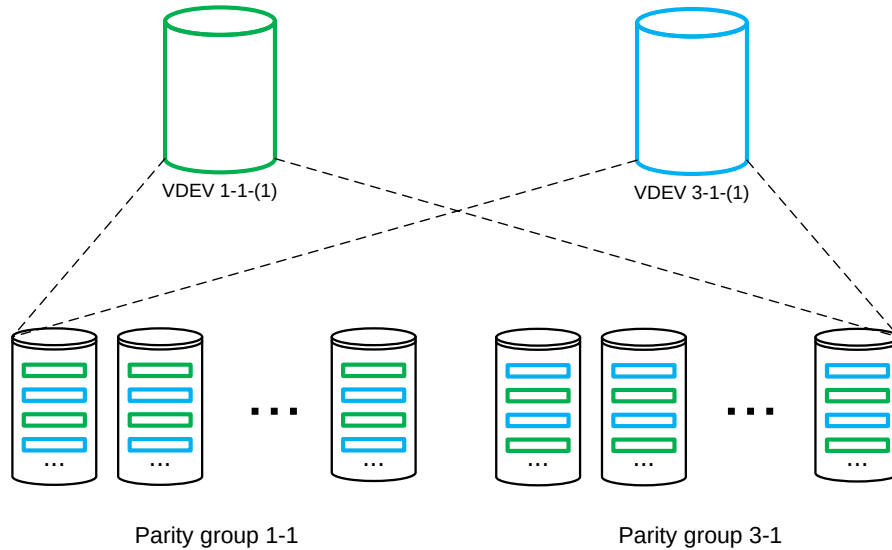
Two separate parity groups, without concatenation



Concatenated Parity Groups are derived from the base Parity Group types by interleaving their VDEVs. The following diagram shows a two-way concatenation or interleaves:

Two-way parity group concatenation (VDEV interleave)

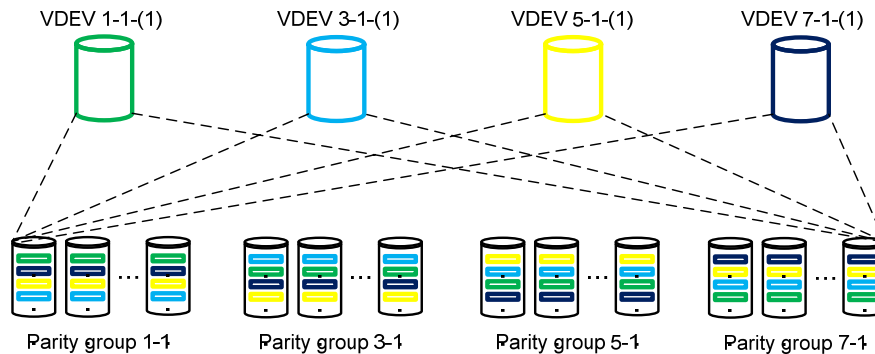
NOTE: Concatenation does not create one big VDEV for both parity groups. Each VDEV has its first stripe on the parity group that it is named after. Thereafter, RAID stripes for a given VDEV alternate between the parity groups in the concatenated set.



There are two types of two-way interleaved Parity Groups.

- The two-way interleave of RAID-1 (2D+2D) is what we usually call '4+4'.
 - Note that there is a VDEV for each Parity Group that is the same size as the VDEV would have been without concatenation. Thus the biggest LDEV you can make on a 4+4 is the same size as the biggest LDEV you can make on a 2+2.
- The two-way interleave of RAID-5 (7D+1P) doesn't have a shorthand name, but if you write $2 \times (7+1)$ this is clear. Sometimes people write "14+2" – if they are writing about the HUS VM family you know that a RAID-5 14+2 could only be $2 \times (7+1)$.

Four-way parity group concatenation (VDEV interleave)



There is one type of four-way interleaved Parity Group.

- The four way interleave of RAID-5 7+1 may be clearly written as $4 \times (7+1)$. If you see someone write $28+4$, this is what they must be talking about.

Yes, the name is confusing.

You will sometimes see the terms “VDEV disperse” or “VDEV interleave”. These are equivalent to the official name “concatenated Parity Groups”. Sometimes people will talk about LDEV striping, and here it is possible that they are talking about concatenated Parity Groups, because the LDEVs that are carved from a VDEV that is part of a concatenated Parity Group set are indeed striped across the Parity Groups that have been concatenated together.

The term “concatenated” is unfortunate, as many native English speakers would assume that this refers to something more like a LUSE. A colloquial English expression often used to describe LUSE is “spill and fill”, because in a LUSE, reading or writing sequentially through the LUSE, one proceeds through each constituent LDEV in its entirety before moving to the next. Thus LUSE does not improve sequential throughput.

Advantages of concatenated Parity Groups

- Individual LDEVs are striped across the set of concatenated Parity Groups. This means that sequential performance is increased, as when staging or destaging sequential data, an I/O to read or write a chunk may be issued to all the drives in the concatenated Parity Group set at the same time.
- Random I/O operations are effectively dispersed across the set of concatenated Parity Groups. If small-block random I/O operations are clustered to within up to a hundred blocks or so, they will likely go to the single drive that contains the chunk (512 KB for Open-V) that contains those small blocks. This is true for any Parity Group type. However, the advantage of concatenated Parity Groups is that if random I/O is clustered within tens or hundreds of MB, these I/Os will be dispersed across the whole set of concatenated Parity Groups. In the case of a LUSE volume, random I/O clustered within 10s or 100s of megabytes will likely be concentrated on a single LDEV that is on a single Parity Group.

- Concatenated Parity Groups reduce to some degree individual Parity Group “hot spots”. The total activity across all the LDEVs on a set of concatenated Parity Groups is distributed more or less evenly across that set of Parity Groups. Thus this “averages out” Parity Group utilization across the concatenated set, and thus reduces average Parity Group busy. Reduced “Parity Group busy” yields better overall average response and offers the opportunity to increase overall throughput.

Note that Dynamic Provisioning offers the benefits of I/O activity dispersal across much larger numbers of Parity Groups.

Disadvantages of concatenated Parity Groups:

- Since I/O activity is dispersed over a concatenated set of Parity Groups, all LDEVs carved from VDEVs in the set are sharing the same performance characteristics. Thus there is less “performance isolation”. Normally dispersing activity is a good thing, but it should be pointed out that any of the LDEVs in the combined set have access to the full bandwidth of the drives in the set, and thus high-activity LDEVs will make the entire concatenated set of drives busy, and thus reduce the performance of the other LDEVs sharing those drives.
- Should a drive fail, performance could be affected for a larger set of LDEVs.
- Most people expect that when you “concatenate” Parity Groups together, you will be able to make bigger LDEVs. As we saw, concatenation does not alter the number or size of the VDEVs that would otherwise be assigned to the same Parity Groups if they were not concatenated together. To fix this problem, if you want the performance characteristics of LDEVs that are striped across a concatenated set of Parity Groups, but you also want a LUN that is as big as the combined formatted size of the Parity Groups in the set, then carve a single maximum-size LDEV out of each VDEV belonging to a set of concatenated Parity Groups, and LUSE these LDEVs together to make one LUN.

Using LDEVs on concatenated Parity Groups as Dynamic Provisioning pool volumes

Recall that DPVOLs are divided up into 42 MB pages, and that each individual page when it has been written to is assigned a location in the pool.

42 MB is “fairly big”, in the sense that the biggest RAID stripe is an Open-V RAID-5 (7D+1P) stripe which has 7 data blocks of 512 KB each for a total of 3.5 MB of data.

What we thus see is that each 42 MB DP/CoW page is comprised at the Parity Group level of a series of RAID stripes. Each page starts on a RAID stripe boundary. The 42 MB size of a page was picked so as not to waste any space in any of the base Parity Group types. One 42 MB page is comprised of:

- 42 stripes of a 2+2, where the stripe size is $2 \times 512 \text{ KB} = 1 \text{ MB}$.
- 28 stripes of a 3+1, where the stripe size is $3 \times 512 \text{ KB} = 1.5 \text{ MB}$.
- 12 stripes of a 7+1, where the stripe size is $7 \times 512 \text{ KB} = 3.5 \text{ MB}$.
- 14 stripes of a 6+2, where the stripe size is $6 \times 512 \text{ KB} = 3 \text{ MB}$.
- 6 stripes of a 14+2, where the stripe size is $14 \times 512 \text{ KB} = 7 \text{ MB}$.

We have not specifically tested the performance of LDEVs on concatenated Parity Groups used as DP/CoW pool volumes. However, there is a theoretical argument that says this might improve sequential performance. Since sequential performance is higher for normal LDEVs on concatenated Parity Groups due to the characteristic of reading / writing a chunk to all the drives in the concatenated set at once, (reading / writing one “round” of interleaved RAID stripes at ones) it would be worth trying to see if this also applies to sequential processing of DP/CoW pages. This may or may not provide a benefit in the context of DP/CoW sequential processing. This document may be revised in future to clarify this point, but for the time being this remains an open question.

Summary of the characteristics of concatenated Parity Groups (VDEV interleave):

- The size of the VDEVs is exactly the same as the size of VDEVs on similar Parity Group types that are not interleaved.
- No change to the maximum size of an LDEV.
- It is not possible to allocate a single LDEV that consumes all the space in all the Parity Groups being concatenated (interleaved) together. Instead, create a maximum-size LDEV on each VDEV associated with the set of concatenated Parity Groups and LUSE these LDEVs together.
- The I/O activity of each LDEV carved out of a concatenated Parity Group VDEV maps to the underlying RAID stripes comprising the VDEV, which means that the I/O activity spreads across all the drives in the concatenation set.
- Sequential I/O throughput is improved because you can read or write one chunk to each of the drives in the concatenated set in parallel.
- Random I/O operations benefit from having an LDEV’s I/O operations distributed across a larger number of drives.
- RAID group performance isolation happens at the level of the concatenated set, as all the LDEVs in all the VDEVs in the concatenation set share the same drives.
- In addition to the base Parity Group types 2+2, 3+1, 7+1, 6+2, and 14D+2P, concatenated Parity Groups yield the types 2x(2+2), which is commonly called 4+4, as well as 2x(7+1) and 4x(7+1).

Appendix 4. PCI Express Details

By Alan Benway

The HM ASIC Grid interconnection system uses PCI Express (PCIe) 3.x serial links for all of its internal connections instead of the PCIe 1.x links that are in use on the VSP model. There are several incremental versions of each PCIe level (such as 1.1, 1.2, 1.3) so I will use the notation of 1.x (and such) here as my general reference.

PCIe interfaces provide a set of “lanes” that are combined together to work in parallel in what is termed a “PCIe link”. The number of lanes present is noted as x1, x4, x8, and x16 in most discussions. [The specification allows for x32, but that isn’t seen outside of a lab.] Each lane has four wires: one differential pair for send and one differential pair for receive. Each pair moves 1-bit of data in a serial connection in one direction. The send and receive pairs operate independently of one another (full duplex) but share the same clock. Understand that the peak data rate of a lane is usually expressed as the aggregate of both directions working simultaneously even though the effective rate for just 100% send or 100% receive is only half of that rate.

PCIe (1.x and 2.x) uses a 10-bit protocol (like Fibre Channel and Ethernet) to move data. This means that every 8 bits of data have 2 bits of parity. There is also a header for each packet of data on a link. This means that only about 80% of the available protocol rate is used to move data (as is the case with FC and Ethernet). However, PCIe 3.x was redesigned to use a far more efficient 128bit/130bit packet which required very different sending and receiving interface hardware. So it can be up to 98.5% efficient.

PCIe is typically implemented as a 1-lane, a 4-lane, or an 8-lane link. The bus controller will send data packets across the link (the number of available lanes) in a parallel byte manner. So x1 sends a single byte (serially), while x4 sends four bytes and the x8 sends 8 bytes in parallel. Each link can also receive at the same rates at the same time as sending. The sender PCIe logic breaks down the data stream from the system processor into these parallel streams and the receiver logic recombines them into a packet.

Looking at the table below, note that each x1 PCIe 1.x link supports a 250MB/sec send plus a 250MB/sec receive data rate which (theoretically) provides a peak data rate of 500MB/sec. Each x4 1.x link supports a 1GB/sec send plus a 1GB/sec receive data rate. Each x8 1.0 link supports a 2GBps send plus a 2GBps receive data rate.

The FE and BE modules used in the HUS VM have one x8 link per board. This means that these modules can support an aggregate data rate (under heavy sustained usage) to and from the Grid Switches of 8GB/sec send and 8GB/sec receive, or an aggregate data rate of 16GB/sec. The MAIN blades use two x8 links between them for communication. There is one x4 link between each MAIN blade and MP blade.

PCIe 1.0 uses a 2500MHz clock whereas PCIe 2.0 uses a 5000MHz clock. This is the only significant difference between them (there is some signal handling difference in the chips). All of the performance numbers for 2.x are double those listed above for the 1.x bus. PCIe 3.x uses an 8000MHz clock with very different chips, and due to the greatly increased efficiency of the data encoders can achieve double the rates of PCIe 2.x even though the clocking rate didn’t double.

The table below summarizes all bus protocols for PCI, Fibre Channel, SCSI, and USB.

Protocol	Common	Performance	Bus	Bus	Aggregate	Aggregate
Type	Name	ratios to PCIe 2.0 x4	Width	Clock	Protocol Rate	Data Rate
PCI 32-33	PCI	0.03 x	32 bit	33 MHz	132 MB/s	106 MB/s
PCI 32-66	PCI	0.1 x	32 bit	66 MHz	264 MB/s	211 MB/s
PCI 64-66	PCI	0.1 x	64 bit	66 MHz	528 MB/s	422 MB/s
PCIx 100	PCI enhanced	0.1 x	32 bit	100 MHz	400 MB/s	320 MB/s
PCIx 100-64	PCI enhanced	0.2 x	64 bit	100 MHz	800 MB/s	640 MB/s
PCIx 133	PCI enhanced	0.1 x	32 bit	133 MHz	532 MB/s	426 MB/s
PCIx 133-64	PCI enhanced	0.2 x	64 bit	133 MHz	1064 MB/s	851 MB/s
PCIx 266-64	PCI enhanced	0.4 x	64 bit	266 MHz	2128 MB/s	1702 MB/s
PCIx 533-64	PCI enhanced	0.9 x	64 bit	533 MHz	4264 MB/s	3411 MB/s
PCIe x1 1.x	PCI Express 1-lane	0.1 x	1 bit	2500 GT/s	625 Mbit/s	500 MB/s
PCIe x4 1.x	PCI Express 4-lane	0.5 x	4 bit	2500 GT/s	2500 Mbit/s	2000 MB/s
PCIe x8 1.x	PCI Express 8-lane	1.0 x	8 bit	2500 GT/s	5000 Mbit/s	4000 MB/s
PCIe x16 1.x	PCI Express 16-lane	2.0 x	16 bit	2500 GT/s	10000 Mbit/s	8000 MB/s
PCIe x1 2.x	PCI Express 1-lane	0.3 x	1 bit	5000 GT/s	1250 Mbit/s	1000 MB/s
PCIe x4 2.x	PCI Express 4-lane	1.0 x	4 bit	5000 GT/s	5000 Mbit/s	4000 MB/s
PCIe x8 2.x	PCI Express 8-lane	2.0 x	8 bit	5000 GT/s	10000 Mbit/s	8000 MB/s
PCIe x16 2.x	PCI Express 16-lane	4.0 x	16 bit	5000 GT/s	20000 Mbit/s	16000 MB/s
PCIe x1 3.x	PCI Express 1-lane	0.5 x	1 bit	8000 GT/s	2000 Mbit/s	1970 MB/s
PCIe x4 3.x	PCI Express 4-lane	2.0 x	4 bit	8000 GT/s	8000 Mbit/s	7880 MB/s
PCIe x8 3.x	PCI Express 8-lane	3.9 x	8 bit	8000 GT/s	16000 Mbit/s	15760 MB/s
PCIe x16 3.x	PCI Express 16-lane	7.9 x	16 bit	8000 GT/s	32000 Mbit/s	31520 MB/s
FC-1	1Gb FC	0.024 x	1 bit	1000 MHz	120 MB/s	96 MB/s
FC-2	2Gb FC	0.048 x	1 bit	2000 MHz	240 MB/s	192 MB/s
FC-4	4Gb FC	0.096 x	1 bit	4000 MHz	480 MB/s	384 MB/s
FC-8	8Gb FC	0.192 x	1 bit	8000 MHz	960 MB/s	768 MB/s
UltraSCSI 80	SCSI	0.016 x	16 bit	40 MHz	80 MB/s	64 MB/s
UltraSCSI 160	SCSI	0.032 x	16 bit	40 MHz	160 MB/s	128 MB/s
UltraSCSI 320	SCSI	0.064 x	16 bit	80 MHz	320 MB/s	256 MB/s
USB-1.2	USB	0.000 x	1 bit	12 MHz	2 MB/s	1 MB/s
USB-2.1	USB2	0.012 x	1 bit	480 MHz	60 MB/s	48 MB/s
USB-3.0	USB3	0.125 x	1 bit	5000 MHz	625 MB/s	500 MB/s
IEEE-1394	firewire	0.010 x	1 bit	400 MHz	50 MB/s	40 MB/s

NOTE: GT/s means Giga-Transfers/sec, which is how most things (CPU front-side busses, graphics CPUs, PCIe) are rated now. I switched to using GT/s since that is how all PCIe 2.x and later documents will show it. For each clock cycle, all of the data in a transfer register is moved. These registers vary a lot in sizes (from 64-bit to 512-bits) and thus transfer rates, so it is more meaningful to use the GT/s notation.

Appendix 5. T10 DIF Details

By Ian Vogelesang

Normally, hosts perform I/O to storage subsystems using 512-byte sectors. All 512 bytes are user data. Hitachi subsystems have always added an 8-byte check-byte field to every 512-byte host sector. Thus internal to Hitachi subsystems, sectors are 520 bytes long, and on back-end SAS and Fibre Channel HDDs or SSDs, I/O is performed in 520-byte sectors. The purpose of the 8 check bytes is twofold.

- 1) There is a logical address check field. The logical address refers to the location of the sector as viewed by the host. It encodes both the name of the LDEV as well as the Logical Block Address, or LBA. The LBA is the number of the sector relative to the start of the LUN. Because the logical address guard field doesn't have room for the entire logical address, as LDEV names are 2 or 3 bytes and LBAs are 48 bits, we typically take a hash of the logical address to form the logical address guard field within the check bytes.

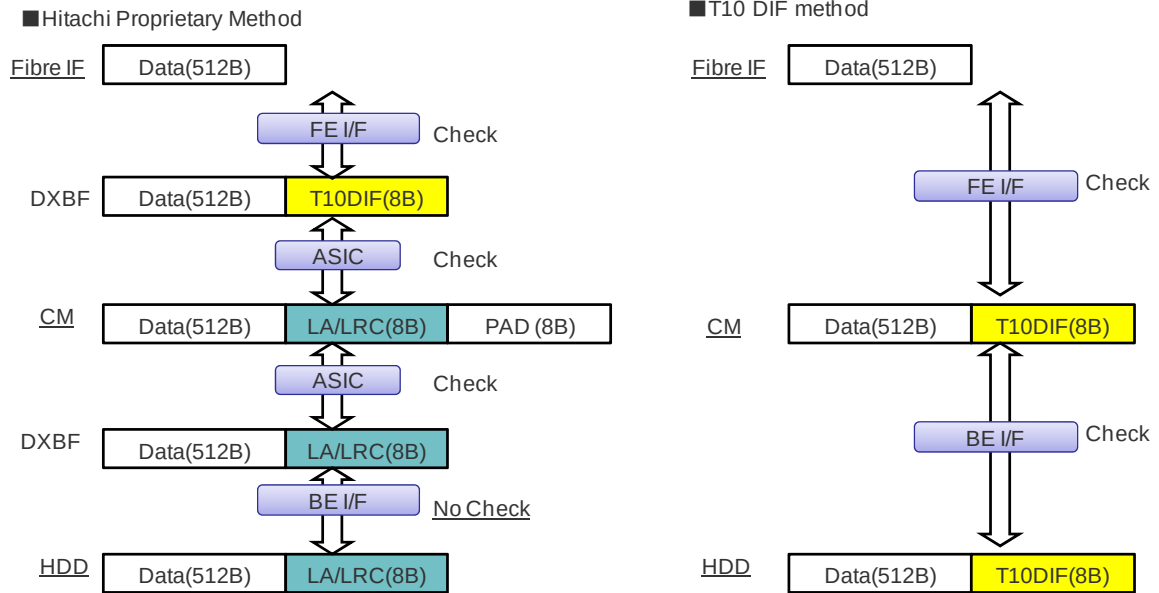
The purpose of the logical address guard subfield is to ensure that there have been no virtualization mechanism failures within the subsystem. The logical address guard field is computed and attached to the host sector before the sector is written to a back-end drive. Then later when the host reads the data, the host gives the subsystem the logical address of the sector and the subsystem fetches it from wherever it originally stored it. The subsystem computes a new logical address guard value based on the host read request, and then when the sector has been fetched from the back end, if the logical address guard subfield that was originally stored with the sector doesn't match, then you know that a mistake was made and the data was either originally written to the wrong location or it was read back from the wrong location.

- 2) The second subfield in the check bytes is a checksum over the payload of the sector. This ensures that as the sector moves through the subsystem's data paths, in to cache, over backend paths, etc. it hasn't been corrupted. If there is any data corruption, the checksum won't match.

Thus Hitachi subsystems have always added 8 check bytes to each host 512-bit sector to detect any subsystem errors, and to prevent the subsystem from presenting invalid data to the host.

The T10 DIF (Data Integrity Field) feature does the same thing, but it's host-based. With the DIF feature, the 8 extra check bytes are added at the host by the device driver before performing the I/O. That way you can get end-to-end data protection covering the HBA, the SAN, as well as within the subsystem. Hosts supporting T10 DIF can perform I/O to Hitachi subsystems using 520-byte sectors. Hitachi subsystems supporting DIF recognize the 8 check bytes from the host and use them in place of the normal 8 check bytes generated by Hitachi subsystems in the absence of the use of DIF. Thus even when the host is using DIF to talk to the subsystem, the subsystem still does back-end I/O in 520-byte sectors. The only difference is that when DIF is used, it's the host that generates the 8 check bytes attached to every 512-byte data sector. The subsystem just uses the host DIF bytes instead of generating its own check bytes. The subsystem still can detect exactly the same virtualization errors or data corruption as always. But with DIF, the host can extend this protection to higher layers in the stack.

Note that DIF is a T10 feature. T10 is the governing body covering the SCSI protocol. Both SAS and Fibre Channel are based on SCSI.





Corporate Headquarters 750 Central Expressway, Santa Clara, California 95050-2627 USA
Contact Information: + 1 408 970 1000 www.hds.com / info@hds.com

Asia Pacific and Americas 750 Central Expressway, Santa Clara, California 95050-2627 USA
Contact Information: + 1 408 970 1000 www.hds.com / info@hds.com

Europe Headquarters Sefton Park, Stoke Poges, Buckinghamshire SL2 4HD United Kingdom
Contact Information: + 44 (0) 1753 618000 www.hds.com / info.uk@hds.com

Hitachi is a registered trademark of Hitachi, Ltd., and/or its affiliates in the United States and other countries. Hitachi Data Systems is a registered trademark and service mark of Hitachi, Ltd. In the United States and other countries.

Microsoft is a registered trademark of Microsoft Corporation.

Hitachi Data Systems has achieved Microsoft Competency in Advanced Infrastructure Solutions.

All other trademarks, service marks, and company names are properties of their respective owners.

Notice: This document is for informational purposes only, and does not set forth any warranty, express or limited, concerning any equipment or service offered or to be offered by Hitachi Data Systems. This document describes some capabilities that are conditioned on a maintenance contract with Hitachi Data Systems being in effect, and that may be configuration-dependent, and features that may not be currently available. Contact your local Hitachi Data Systems sales office for information on feature and product availability.

Hitachi Data Systems sells and licenses its products subject to certain terms and conditions, including limited warranties. To see a copy of these terms and conditions prior to purchase or license, please go to <http://www.hds.com/corporate/legal/index.html> or call your local sales representatives to obtain a printed copy. If you purchase or license the product, you are deemed to have accepted the terms and conditions.

© Hitachi Data Systems Corporation 2009. All Rights Reserved
WHP-###-### November 2012