

HPE Nimble Storage, HPE Alletra 5000, and HPE Alletra 6000 storage architecture

The history and implementation of the Cache Accelerated Sequential Layout



HPE Nimble Storage

HPE Alletra 5000
HPE Alletra 6000

Ultra efficient
Optimized for data efficiency

6-nines availability
Guaranteed as standard benefit

AI-driven
Built with industry's leading AIOps

Simple to use
No domain expertise required

Ideal for:



VM farms



General purpose workloads



Test/Dev



Backup and DR

Contents

Executive summary	3
The history of CASL	3
The design center	4
The building blocks	4
RAID layer	4
RAID layout.....	5
Extreme data protection.....	6
HPE Nimble Storage write to stable media	7
Triple+ Parity RAID resiliency compared to traditional RAID types	7
Cascade multistage checksums.....	9
Hard-to-detect errors.....	9
Block uniquifier store layer.....	12
Fingerprint index.....	12
Block index layer	13
Volume manager layer.....	13
SCSI layer	14
Garbage collection.....	14
NVRAM/NVDIMM	15
Write path.....	15
Read path	16
Cache types	16
Caching in hybrid platforms	17
Block eviction in hybrid platforms	17
Putting it all together.....	18
HPE Nimble Storage terminology guide	18

Executive summary

The Cache Accelerated Sequential Layout (CASL) architecture from Hewlett Packard Enterprise provides the foundation for the strengths of HPE Nimble Storage technology—high performance and capacity efficiency, integrated data protection, and simplified management. CASL is a log-structured file system (LFS) that incorporates the best properties of both spinning media (sequential I/O) and flash (random I/O).

Target audience

The audience for this paper includes chief information officers (CIOs), chief technology officers (CTOs), data center managers, enterprise architects, and anyone who wants to learn more about how CASL works.

Document purpose

This white paper breaks down the layers of the CASL architecture, explaining their various purposes and showing how they interact with each other. It also explains the data protection and data integrity mechanisms, along with the benefits of the architecture.

Branding transition

In May 2021, HPE unified its primary storage portfolio under the name of HPE Alletra. The HPE Alletra 6000 All Flash series and the HPE Alletra 5000 Hybrid series are based on the original HPE Nimble Storage architecture with the same features and functionality. Therefore, all content within this white paper is applicable to the original HPE Nimble Storage models as well as the new HPE Alletra 6000 and HPE Alletra 5000 models.

The history of CASL

CASL was designed in 2008 with flash in mind. At the time, the founders (Umesh Maheshwari and Varun Mehta) had initially designed a 2U 24x SSD system that would act as an all-flash cache accelerator by intercepting NFS client/server requests. The following figure shows their description of its benefits.

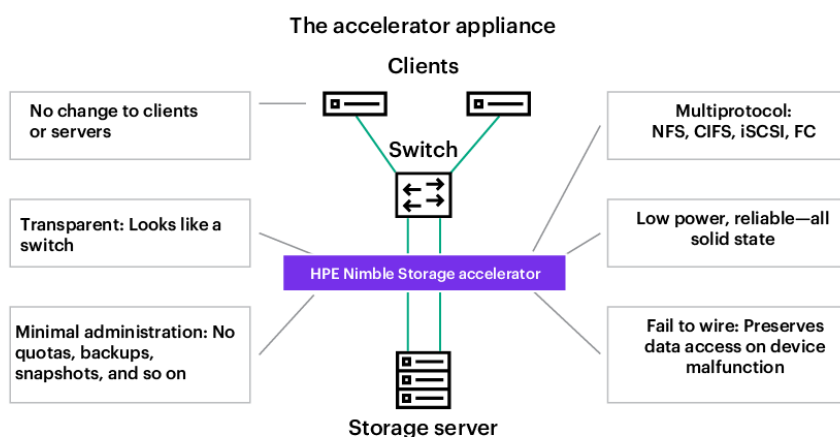


Figure 1. Original graphic of the storage accelerator from the HPE Nimble Storage founders' slide deck in 2008, prior to the HPE acquisition

Then, in 2009, they decided to forgo the accelerator and build a stand-alone storage platform, a hybrid flash array, by extending the use of the existing architecture and file system that they had already developed to better serve the majority of customer storage needs.

From all-flash to hybrid and again to all-flash, the HPE Nimble Storage CASL architecture has made an unprecedented industry transition that showcases its flexibility and adaptability to changing market demands.

The design center

The design principles that underpin CASL enable the platform to deliver excellent reliability, industry-standard protection and data integrity, and deterministic performance at scale—while also lowering risk and accelerating business outcomes. These principles are wide-ranging:

- Support a fully sequential layout by always writing full stripes
- Use variable block I/O sizes
- Scale performance by increasing the number of CPU cores
- Use scale-up and scale-out architecture for consistent IOPS/TB
- Buffer writes in nonvolatile memory continuously before forming full RAID stripe writes
- Cache random reads in the flash cache on hybrid arrays
- Cache random reads in the all-flash array (AFA) storage
- Ensure that all writes, including overwrites, always occur in free, contiguous space
- Avoid filling holes and provide zero fragmentation
- Always use inline adaptive compression techniques for best immediate storage efficiency
- Use inline deduplication, which takes advantage of spatial locality, for best immediate storage efficiency
- Employ data protection and data integrity mechanisms against insidious errors that other RAID and checksum systems can't fix
- Incorporate internal auto-QoS prioritizing of different workload characteristics and internal housekeeping tasks

The architecture implements a fully sequential layout for four important reasons:

- **Efficiency:** When storage media performs sequential writes faster than random writes, the underlying file system needs to convert random writes into sequential writes. This requirement applies to spinning disk because random writes require disk head movement.
- **Longevity:** It also applies to NAND flash because, before a single page composed of a few kilobytes can be written, an entire multi-megabyte block (many NAND pages) must first be erased. Writing in large blocks sequentially creates a NAND-friendly workload that extends the longevity of the flash media.
- **Resiliency:** Writing sequentially makes RAID calculations and overheads much lighter on systems, which in turn enables extreme resiliency RAID constructs such as Triple Parity+ RAID to be used without a performance penalty.
- **Stability:** The fully sequential layout provides long-term performance stability at different levels of capacity utilization.

The building blocks

The CASL architecture is organized into multiple layers that work together:

- The RAID layer
- The segment layer (SL)
- The LFS layer
- The block uniquifier store (BUS) layer
- The block index (BI) layer
- The volume manager (VM) layer
- The SCSI layer

RAID layer

The design goal of HPE Nimble Storage / HPE Alletra 5000 / HPE Alletra 6000 Triple+ Parity RAID was to far exceed the level of resiliency compared to all other methods of RAID protection, even for extreme cases of temporally correlated unrecoverable read errors (UREs)—while maintaining an impressive usable:raw ratio and high performance.

Triple+ Parity RAID design center and benefits

Different media types exhibit varying failure characteristics. Some media may fail abruptly (for example, the motor of a spinning drive may simply stop working). Other media types fail more gradually (For instance, as they age, SSDs start exhibiting more frequent UREs.). This behavior is not related to the wear level of SSDs nor to whether they are SLC, eMLC, or 3D-TLC.

For more background: An interesting academic study was done in conjunction with Google™ and covered many millions of SSDs of all types. Media such as QLC (and beyond) may exhibit such URE behaviors even sooner, especially with large, archive-class drives.

Most importantly, as drives get bigger, the statistical likelihood of temporally correlated UREs during a RAID rebuild becomes much higher. This is because RAID rebuilds need to go through a much larger amount of data, which increases the chances of simultaneous multiple URE occurrence (A common misconception is that rebuild time is the only major consideration—the reality is that how much data needs to be processed during a rebuild is a much more important factor since that's directly related to the UREs per data read drive specifications.).

Traditional RAID is ill-suited to very large drives and cannot cope when facing more UREs than parity count. For instance, RAID 5 can withstand a single URE. RAID 6 can withstand two UREs in parallel. Normal Triple Parity RAID can withstand three UREs in parallel.

Triple+ Parity RAID can withstand N UREs in parallel, where N is the number of drives in the RAID group, even if all normal parity has been lost. This means that a system can completely lose three drives in a Triple+ Parity RAID group and have simultaneous UREs on all remaining drives, and that system will not suffer data corruption. By comparison, with RAID 5, if a single drive has been lost (no parity left), zero UREs can happen without the system losing data. With RAID 6, if a single drive has been lost, only a single URE can be sustained until a rebuild finishes.

As a result, the design of Triple+ Parity RAID has the following characteristics:

1. It can accommodate both current and future media types, including media that are exhibiting an incredibly high incidence of temporally correlated UREs, to the extent that UREs are occurring in parallel across all drives (which would break any other protection mechanism).
2. It can accommodate extremely large drives and tolerate very long rebuild times safely (for example, rebuilding a failed 100 TB drive while maintaining extremely high resiliency for the remaining drives).
3. It provides extremely high usable:raw space since large RAID groups (up to 24 drives in a single RAID group) can be used safely, which means less parity space waste.
4. It becomes safe for deployments where one may have problematic environmental and/or poor access to provide spares (for instance, very remote installations for critical systems).

RAID layout

Media (whether flash or spinning) is automatically organized into RAID groups that are completely abstracted from the end user. In addition to RAID groups, the architecture implements a **Triple+ Parity** RAID algorithm that uses distributed parity, with an additional intradrive parity associated with each chunk (the "+" in Triple+).

Note

A chunk is part of a stripe. It defines the amount of data that is written into a single drive and has its special additional parity and checksums. Chunk size and RAID layouts vary between HPE Nimble Storage AFAs and hybrid systems.

Extreme data protection

In addition to Triple Parity, which is calculated horizontally across an entire stripe, the system also calculates intradrive parity. This type of parity, also referred to as Triple+ Parity RAID, is calculated by using chunk data and metadata that is written in each drive and stored locally.

Intradrive parity can be used to recover from UREs, even if there is no redundancy left in the stripe (for example, if multiple drives fail and multiple UREs occur in parallel during rebuild).

Intradrive parity is implemented in both HPE Nimble Storage All Flash (AF) and Hybrid Flash (HF) platforms, as well as their successors, HPE Alletra 6000 and 5000 respectively. It enables the recovery from four consecutive ECC errors on SSDs, two consecutive sector read errors on 512B formatted HDDs, or one sector read error on 4 KB formatted HDDs. In addition, recovery is independent of the condition of the remaining drives. This means that a Triple+ Parity RAID group could lose three drives and still have protection from sector read errors, a feat utterly impossible with standard RAID.

Note: A recent heuristics enhancement with Triple+ RAID enables systems to automatically shut down if there is a high likelihood that too many drives will fail in rapid succession (Support can disable this enhancement if desired—HPE recommends not to.).

In hybrid systems, this means that if three drives are lost, the system will shut down since if a fourth one is lost, there would be a potential for data loss.

In all-flash systems, if three drives are lost rapidly before the system can fully rebuild one, then it means that something very wrong is probably happening, and as a result, the system will shut down before a fourth one is lost.

However, if two drives are lost, the rebuild happens successfully, and then a third drive is lost. The system will continue operating since it still has one full parity available. But if another drive is lost, then it will shut down.

This behavior is another example of the extreme data integrity approach of these systems (Most other arrays, if not all, will continue operating without any parity left, even if they can't rebuild a drive.).

The following figure shows how each drive has a chunk written to it and how that chunk has its own extra parity (intradrive parity) in addition to the full three parities available.

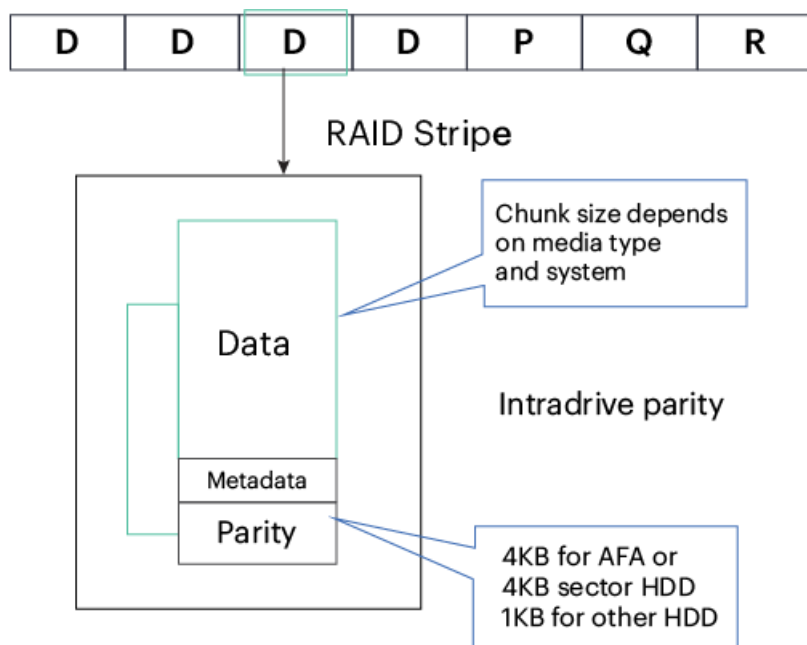


Figure 2. Intradrive parity per chunk

Unlike reads and writes in spinning disks, reads and writes in flash are dealt with asymmetrically by the media. For instance, it takes much more time to erase and write a flash cell than it takes to read from it. Flash also has a finite write lifetime; therefore, the way storage systems write to it becomes very important.

CASL always writes data in full stripes by using a fully sequential layout without a read-modify-write penalty, even at very high-capacity utilization rates. This approach differs from that of other LFS implementations (such as ZFS and WAFL), which are fragmenting, hole-filling file systems.

The stripe size or segment size is determined by multiplying the number of data drives in a RAID group by the chunk size, and it varies between AFA and hybrid platforms. The following figure shows how variable size ingest blocks are coalesced into a chunk per drive to build the stripes.

HPE Nimble Storage write to stable media

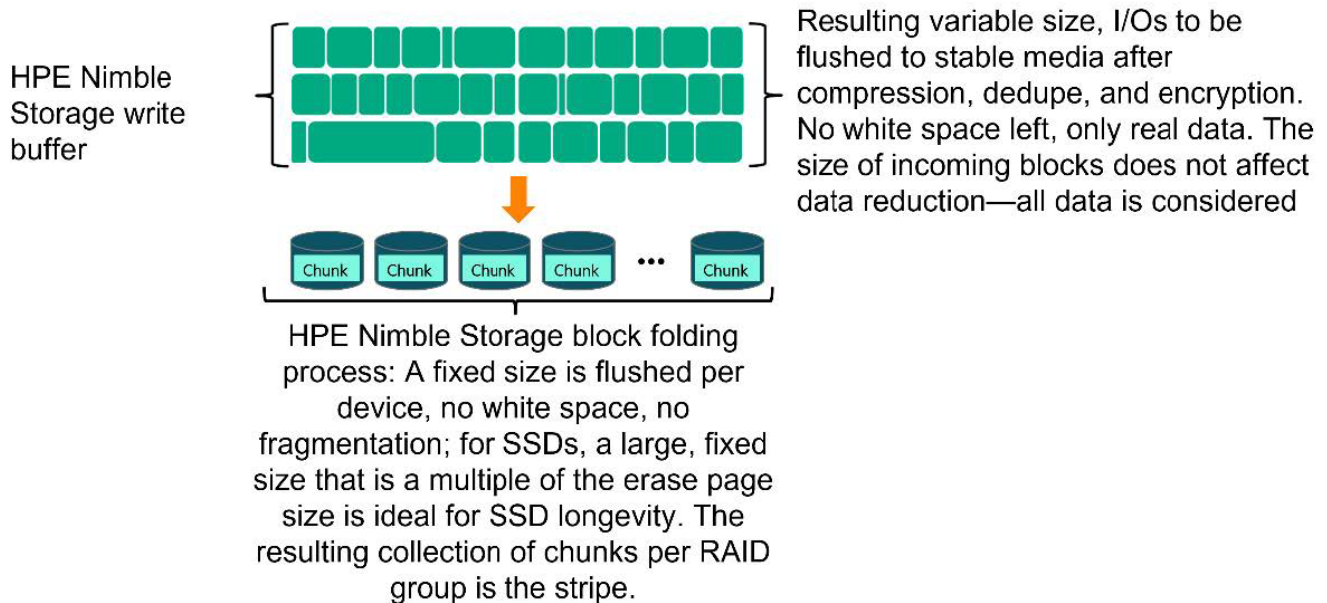


Figure 3. How HPE Nimble Storage writes to stable media in chunks

Triple+ Parity RAID resiliency compared to traditional RAID types

All data protection systems have a mean time to data loss (MTTDL). This is the average time by which there will be data loss that is utterly unrecoverable by the capabilities of the storage system (parity, sparing, and rebuilds). The goal of a storage system with strong data integrity and resiliency is to make the time to data loss so far in the future that there is no realistic reason to worry about it, even under extremely adverse conditions.

A high incidence of UREs lowers the MTTDL dramatically. Table 1 shows the MTTDL comparisons with different assumptions. This was done using standard MTTDL math and there are several online calculators if one wants to replicate this math. Find one that models UREs and shows a Triple Parity RAID such as Z3—then, multiply the MTTDL for Z3 by a factor of five to approximate what intradrive parity does for Triple+ Parity RAID.

The true delta between Triple+ and plain Triple is actually far more than five, but HPE engineering is conservatively using five.

Table 1. Triple+ Parity RAID MTTDL compared to typical legacy RAID types and group sizes

Scenario	RAID 5, 6x SSDs/RAID group—Baseline	RAID 10, 24x SSDs/RAID 6, 8 SSDs / RAID group	RAID 6, 20 SSDs / RAID group	RAID 6, 20 SSDs / RAID group	Triple+ Parity RAID, 24 SSDs / RAID group
3.84 TB SSDs, good quality drives, low URE	1	11.4x	2205x	14.4x	505654x (over five hundred thousand times better than R5 even with 24 drives per RAID group)
3.84 TB SSDs, high incidence of UREs but good MTBF	1	12.5x	2169x	11.8x	13244x
3.84 TB SSDs, high incidence of UREs, low MTBF (for example, a bad batch of drives and environmental challenges)	1	12.4x	66x	0.38—R6 with 20 drives in a RAID group stops working well in this kind of situation; it becomes less reliable than R5 with a small RAID group.	115.7x even for this extreme situation, and with a far better usable:raw than anything else
100 TB SSDs, normal quality drives	1	11.4x	84x	0.56—R6 is not quite suitable for 100 TB drives with large RAID groups.	20586x
100 TB SSDs, bad batch of drives	1	11.4x	2.65x	0.031—large R6 groups are simply unsuitable for this scenario.	95.6x—worst-case scenario, huge drives that are unreliable

Notice that Triple+ Parity RAID in these examples has a much larger RAID group size than the rest (24 drives). In RAID math, the more drives in a RAID group, the less reliable it is. Triple+ Parity RAID achieves incredibly high levels of reliability while allowing a large RAID group size, which helps achieve a very high usable:raw without compromises (The usable:raw in HPE Nimble Storage / HPE Alletra 6000 AFAs with 24 drives per RAID group is about 73%, after all overheads, including a virtual spare per 24 drives.).

Even a very conservative RAID 6 implementation with only eight disks per RAID group has poor usable:raw due to double the parity requirement across 24 disks versus Triple+ Parity RAID—yet offers far lower protection than Triple+ Parity RAID.

In all cases, Triple+ Parity RAID remains more resilient than all other RAID types. Note: For the MTTDL calculations in the table, the following variables were used:

- 150 MB/s rebuild speed
- Normal quality drives: 1.2M hours MTBF, 1 in 10¹⁷ URE

– Bad batch of drives: 36.5K hours MTBF, 1 in 10⁶ URE

Cascade multistage checksums

The design goal of HPE Nimble Storage checksums was to create safe storage by building a way to detect and correct internal array errors that are utterly undetectable by traditional checksum types—and extend this type of extreme error correction even to constructs such as snapshots and replicas.

Design center for HPE Nimble Storage checksums

One common misconception about RAID (and by extension, erasure coding) is that it ensures data integrity. RAID by itself, no matter how strong, does not protect against several types of data corruption.

Traditional, industry-standard checksums (as found in many storage devices) do not protect against errors such as lost writes, misdirected reads, and misplaced writes. These types of errors are referred to by the storage industry as silent errors. Unless a storage system provides the necessary detection and recovery mechanisms, these errors always lead to data corruption and can be very difficult to trace.

As a result, many users lack awareness that such errors exist, simply because most storage systems simply do not provide the necessary detection and correction mechanisms.

Furthermore, modern storage systems generate a lot of metadata, especially with deduplication. Therefore, any metadata corruption can have catastrophic consequences for a very large portion of data.

With deduplication, snapshots, clones, and replicas, any corruption on highly referenced data (pointers or reference counters) means that multiple logical blocks are now corrupt.

The worst and most dangerous corruptions of all are silent and result in the wrong data being read. Imagine this happening to mission-critical medical or financial data.

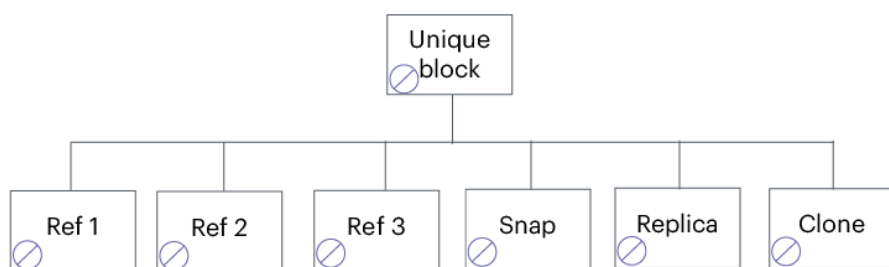


Figure 4. The catastrophic domino effect of silent corruption in a single block

Consider the scenario shown in Figure 5. Imagine a misdirected read error affecting the unique block—an error undetectable by standard checksums.

All the logical blocks that rely on this unique block will be corrupt without the users having any knowledge of the error—all the duplicates, any snapshots, clones, and remote replicas, affecting multiple LUNs in the system that share the same unique block. Potentially hundreds of virtual references of this block, when read, would return the wrong piece of data.

This kind of domino-effect logical corruption makes ensuring internal data correctness even more important than it ever was. In legacy storage systems with LUNs living in a single RAID group, any such corruption would be limited to a LUN and some snapshots. In today’s systems, such corruption would be pool-wide—which is why HPE Nimble Storage built a comprehensive system to fully protect against such errors.

Hard-to-detect errors

There are three main categories of hard-to-detect errors:

- **Lost write:** The write initially looks as if it completes and goes through the entire RAID process correctly. But the storage media may not commit the written data. When reading this piece of data, the older data present in that location will be read instead. It may not technically be corrupt data, but it will be temporally wrong data

(not the most up-to-date version). Because this data doesn't look corrupt (It is correct bitwise but out of date.), traditional checksum systems do not detect the error.

- **Misdirected write:** The write completes and goes through the entire RAID process correctly. It is also fully committed to the storage medium—but not at the correct location. When trying to read the data, the old (previously correct) data will be instead read from the correct location. Once again, this is not detected by traditional checksums because it is not data corruption, but instead reading an older version of the data.
- **Misdirected read:** The read is not attempted at the correct location, but instead from the wrong area of the storage media. The data read is correct (there is no corruption) but it is the wrong piece of data. Traditional checksums will again not detect this kind of problem.

Research shows that these kinds of errors are usually due to firmware bugs on drives. These errors are extremely hard to detect (and actually impossible to detect with many arrays). As a result, the errors may not be fixed until someone with the means to detect the errors, and with enough field density and telemetry, alerts the drive manufacturers. Fortunately, the HPE Nimble Storage checksums are so strong that even incredibly problematic drives will result in no data corruption.

How HPE Nimble Storage cascade multistage checksums work

In addition to storing the checksum data, a self-ID is assigned to each stored object. The self-ID is not a single number—it includes multiple pieces of data including the block address, along with a unique, never-repeating, monotonically increasing serial number identifier. In the “**Block unifier store layer**” section of the document, there is an explanation regarding the sequential block number (SBN). The SBN will be different for all data even if the same data is trying to be overwritten (for example, trying to update a block with the same data). This is achieved by using various indexes, CASL maps and tracks SBNs, checksums, and the block location on disk, which makes it easier to detect and correct silent errors and maintain strict write order fidelity.

Traditional checksum systems omit the self-ID part (or have a simplistic, small set of non-unique numbers) and it is precisely the unique self-ID that allows complete detection of errors, such as lost writes.

When reading, both the checksum and self-ID are compared to what is stored on the storage media. If either does not match what is expected, the corrupted data is transparently rebuilt from Triple+ Parity RAID:

1. If data is corrupted, the checksum fixes it.
2. If data is read from the wrong location, it will have the wrong self-ID—the correct data will be rebuilt from Triple+ Parity RAID.
3. If a write is lost, the old data will have the wrong self-ID—the correct data will be rebuilt from Triple+ Parity RAID.

This strong checksumming **is performed on multiple levels, not just per block**. This results in a cascade of checksums across multiple stages of the I/O path. All the stages of checksums have to be correct to mark a piece of data as correct:

- As data is ingested into memory and NVDIMM
- Before and after data reduction
- Per stored block
- Per chunk
- Per snapshot
- For replicated data

Many systems carry out checksums as per stored block, which means logical corruption on replicated data may not be detected. With CASL, data correctness is strictly enforced and never assumed, at all stages.

Performance characteristics of cascade multistage checksums

This extensive protection is a fundamental part of the system and cannot be turned off (just like there is one type of RAID: Triple+).

A commonly asked question is whether this extreme level of protection has a performance penalty. Since CASL writes all data sequentially, storing this extra information is not difficult from an I/O standpoint. It does need some

extra CPU calculations compared to checksums that do not protect against insidious errors, but the trade-off is worthwhile since the data integrity payoff is immense.

The extended checksum information takes a little extra space and is accounted for in sizing.

HPE sizing tools consider the complete system, which includes all overheads, metadata, checksumming, and Triple+ Parity RAID, when doing any performance and capacity estimations.

Disk scrubbing

Disk scrubbing is a technique whereby a storage array reads existing data by itself, scanning for errors and fixing read errors as it finds them. The HPE Nimble Storage systems carry out continuous disk scrubbing. It is important to note that disk scrubbing may be useful, but it is not a substitute for strong RAID and checksums. It cannot fix errors beyond what the checksums and RAID can and does not work in real-time but is low priority in nature (user I/O always has higher priority than disk scrubs in all disk arrays).

Segment layer

On top of the RAID layer, CASL implements a logical SL. The SL divides the disk into contiguous physical space called slots. Slots map into segments, which in turn map into full RAID stripes. Therefore, there is a 1:1 relationship between a segment and a full RAID stripe.

The SL exposes slots to the upper layers in the form of logical segments. To enable the tracking and management of all resources logically, the architecture also implements segment identifiers (IDs). These unique IDs are assigned to allocated segments sequentially from zero to infinity. Therefore, for a given segment ID, the system can easily map the physical slot location on disk.

The following figure shows how the segment layer is logically above the RAID Layer.

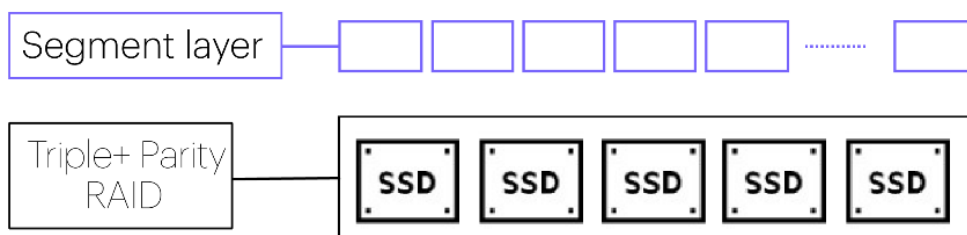


Figure 5. The segment layer

Finally, CASL never performs in-place overwrites. For that reason, existing segment IDs are not reusable, which in turn means that the same segment cannot be used until it has been reclaimed via garbage collection.

LFS layer

The LFS sits on top on the SL. The LFS organizes variable-length user data blocks information segments in a self-descriptive manner that enables the system to quickly identify and reliably return the location of a block in the LFS to the requestor. After the segments are filled, the LFS layer flushes them into persistent storage. The following figure shows the logical position of the LFS layer.

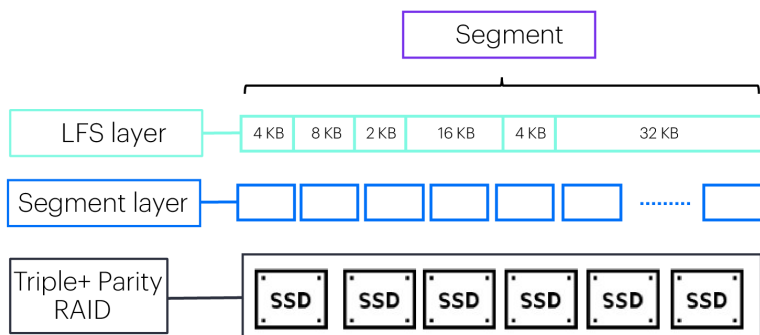


Figure 6. The LFS layer

This layer is also responsible for data reduction services and data integrity techniques that use checksums. In addition, it identifies insidious errors (for example, a drive might signal that a block was successfully written to it when the block was not written or was written elsewhere). These error conditions are known in the storage industry as **lost writes or misdirected writes/reads**. The LFS layer uses checksums and block identifiers to determine whether the requested block is read from the correct location and—very importantly—whether it contains valid data. This is a critical data integrity attribute which, surprisingly, few storage platforms implement.

This protection against insidious errors is critical because standard RAID by itself cannot protect against such issues. In addition, modern storage devices that perform deduplication rely heavily on metadata tracking. Therefore, any metadata corruption can be extremely damaging to storage systems, which is why the architecture implements such strong data integrity mechanisms.

Block uniquifier store layer

After the system determines how to read and write through the LFS layer, it becomes important to track the location of the writes. To accomplish block tracking, every block in the system is assigned a unique ID called an **SBN**, which is different from a LUN ID or an offset. Tracking occurs through a highly optimized indexing mechanism called the **disk index (DI)**, which maps a unique block ID (SBN) to an LFS location and maintains other block-related metadata, such as the block checksum, the reference count, and the block fingerprint value.

The following figure shows the logical position of the BUS layer.

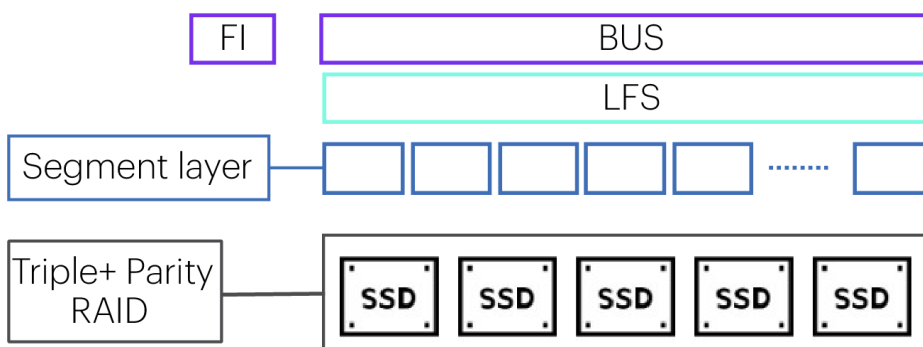


Figure 7. The BUS layer and the fingerprint index

In hybrid systems, DI is pinned in the flash cache and cached in **memcache**. In AFAs, DI is cached in memcache.

Memcache is an in-memory data structure present in both hybrid and AFA platforms that is responsible for storing indexes.

Fingerprint index

For deduplication, along with the SBN, every unique block that enters the system is also assigned a unique 256-bit fingerprint. The architecture uses a two-layer fingerprint design that is composed of long and short fingerprints. It uses the short fingerprint value to quickly identify potential duplicates, which then confirms by performing full fingerprint comparisons. When blocks are updated or deleted, the **fingerprint index (FI)** is also updated to reflect these changes.

Because SBNs are assigned sequentially, fingerprint matching becomes trivial. It enables better spatial locality for SBN lookups because duplicate blocks tend to be adjacent to each other and in many cases occur as a result of data cloning.

In addition, because of the two-layer fingerprint design, losing the FI does not result in data loss. The two-layer FI design provides multiple benefits:

- It enables the system to keep DRAM requirements small.
- It provides a very quick hash comparison mechanism to identify potential fingerprint matches.
- It prevents the platform from being constrained by DRAM-to-flash capacity ratios, as compared to similar industry offerings.
- It reduces controller costs and passes on the savings to the end user.
- It increases per-controller supported capacities while requiring fewer controllers for full-capacity scale.

Block index layer

The block index (BI) layer helps provide space-efficient snapshots, clones, and deduplication. As mentioned previously, every block in the system is assigned a unique block ID.

From a protocol perspective, read and write requests coming into the system do not contain block IDs. They contain SCSI commands with logical block addresses, offsets, LUN IDs, and so forth. BI is a data structure that is maintained on a per-volume generation. A **volume generation** is a view of the volume at a particular point in time.

BI maps a volume, and its offsets to a corresponding SBN for every volume in the system. It also enables block sharing for use by snapshots and deduplication because it is possible for multiple LBAs, from multiple volumes, to point to a single SBN.

Another BI component is the **dirty block buffer (DBB)**, which tracks data blocks in NVDIMM. DBB is responsible for flushing these blocks to disk, based on system-defined watermarks and the number of incoming blocks. The following figure shows the logical position of the BI layer above the BUS layer.

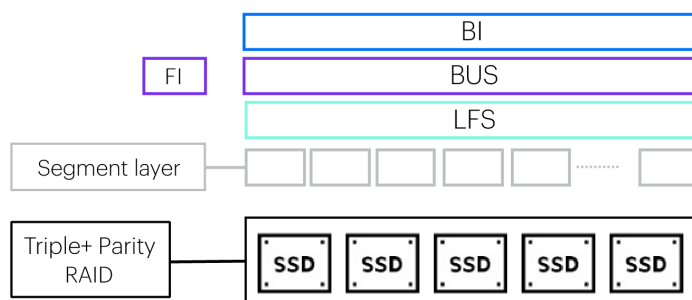


Figure 8. The block index layer

Volume manager layer

The volume manager (VM) layer manages the lifecycle of generations, maintains several metadata objects, and orchestrates system checkpoints and recovery workflows. It also provides striped volume functionality, which makes it possible to stripe a volume across multiple HPE Nimble Storage systems and pools. The following figure shows the Volume Manager layer above the BI layer.

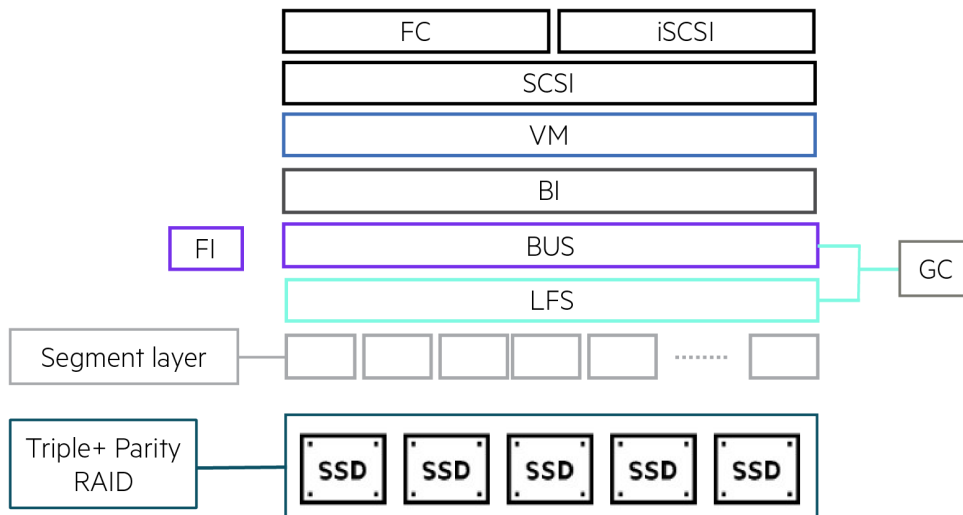


Figure 9. The VM layer and the protocols

SCSI layer

On top of the VM layer is the SCSI layer. This layer implements SCSI commands and communicates with the Fibre Channel and iSCSI protocols, which in turn satisfy host read and write requests.

Garbage collection

Garbage collection (GC) is the process responsible for freeing contiguous segments. Although the system always writes in full segments, over time, as blocks get deleted or updated, the number of valid blocks in an allocated segment decreases. At that point, the system intelligently selects low-utilized segments based on feedback from the BUS layer. It copies the existing valid blocks into a new clean segment while maintaining block sequentially (that is, valid blocks that came together are written together), and it marks the old segment as clean and available for new writes.

This ultralightweight GC technique is implemented in HPE Nimble Storage AFAs and hybrid platforms (which is an industry-unique feature). It ensures there is always free, contiguous space to accommodate write requests and provides deterministic performance at high-capacity utilization rates.

GC is a significant differentiator as compared to other LFS industry implementations. At high-capacity utilization rates, those implementations fill in scattered file system holes, thereby converting every read/write request to a random I/O request. That approach lowers performance, elongates response time, and vastly increases CPU load. Figure 10 shows how Garbage Collection happens at the BUS/LFS layers.

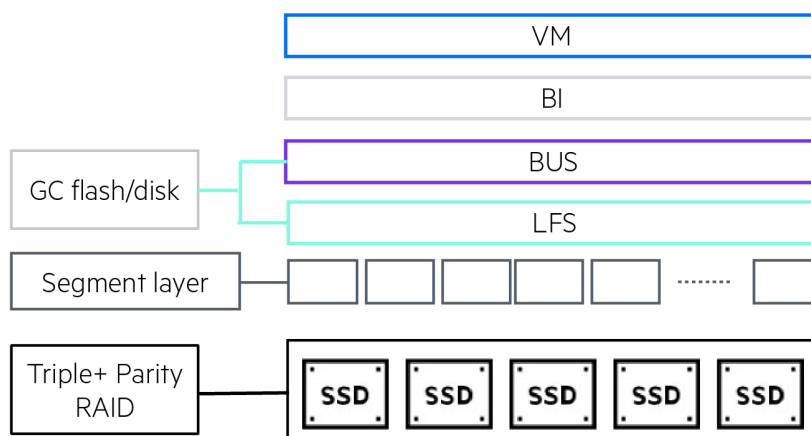


Figure 10. Garbage collection

NVRAM/NVDIMM

All incoming write requests, regardless of I/O size, must land in NVRAM/NVDIMM. HPE NimbleOS protects the requests by copying them to the secondary controller's NVRAM/NVDIMM before they are acknowledged to the host. The following figure shows the internal logical structure of the NVDIMM-N used for NVRAM.

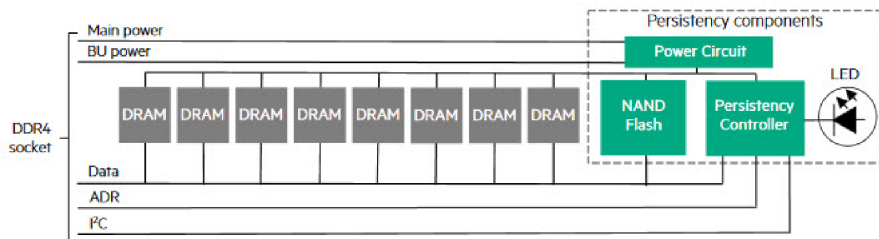


Figure 11. NVDIMM

NVRAM is implemented as a byte-addressable **NVDIMM-N** technology in each controller. NVDIMM-N is JEDEC's persistent memory implementation, which uses DRAM (DDR4) and low-latency flash as a persistent backup layer and a supercapacitor.

During a power failure, the supercapacitor enables the NVRAM contents to be safely offloaded to the onboard flash module. When power is restored, DRAM contents are restored from flash.

The system continuously checks (and alerts) the capacitance levels and ensure enough charge is always available.

Write path

In AFA environments, the write path includes the following steps, also shown in Figure 12.

1. Writes from the host land in NVRAM/NVDIMM.
2. The writes are mirrored through PCIe DMA to standby controller NVRAM/NVDIMM.
3. The standby controller acknowledges the writes to the active controller.
4. The active controller acknowledges the writes to the host.
5. The NVRAM/NVDIMM contents are organized in the following ways:
 - a. The data is deduplicated by using variable block deduplication.
 - b. The data is compressed by using variable block compression.
 - c. Blocks are formed into multiple MB segments.
 - d. Segments are sequentially written to SSDs and indexes are updated.

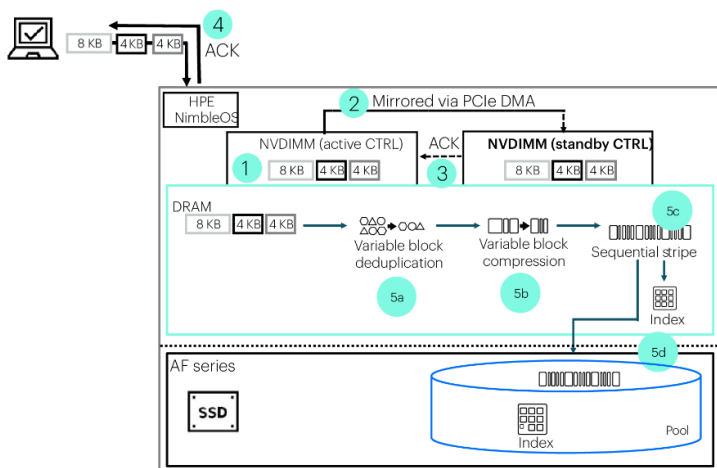


Figure 12. Write I/O path for AFA

Writes in hybrid platforms generally follow a similar write I/O path, with a few differences. The segment size differs from that of the AFA and cache worthy blocks are immediately cached in the SSD flash cache layer based on system heuristics and caching policies.

Read path

In AFA environments, the read path for incoming host read requests includes the following steps, also shown in Figure 13.

1. The algorithm checks NVDIMM for the block.
2. If the block is not found, the algorithm checks DRAM.
3. If the block is not found there, the algorithm performs a fast index lookup and reads from SSD.
4. If the block is located, the system validates its checksum and its block ID to detect misdirected reads, misdirected writes, or lost writes.
5. The system then decompresses the block and returns it to the host.

Although the read paths are similar for AFA and hybrid platforms, the hybrid path includes an additional **cache index (CI)** implementation, which enables quick identification and retrieval of cached blocks from the SSD flash cache layer.

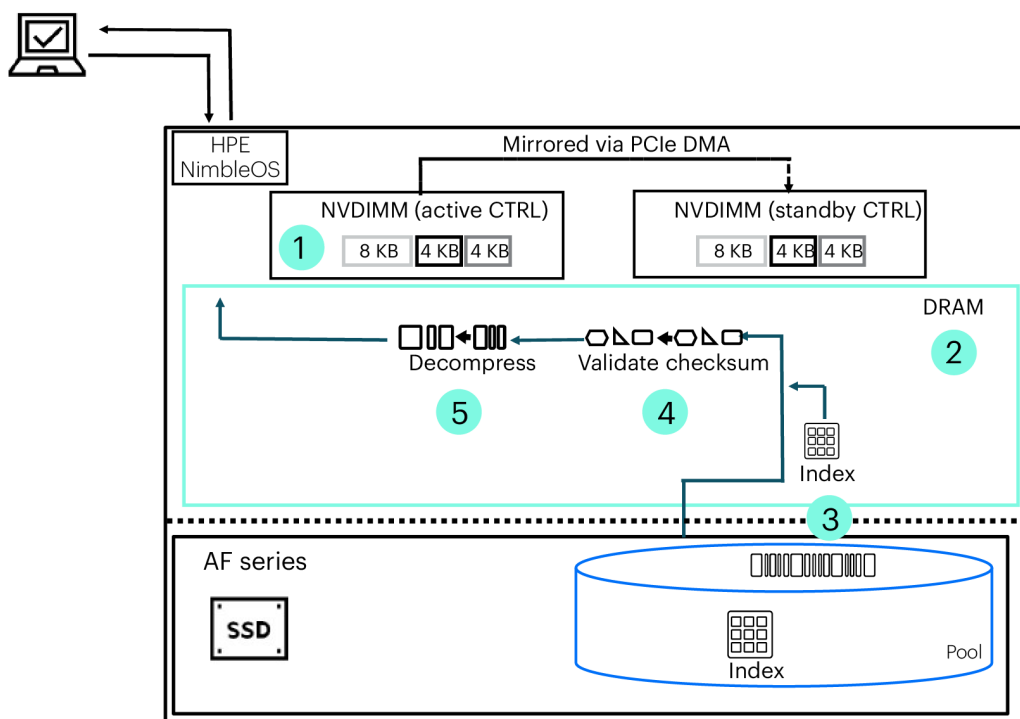


Figure 13. Read I/O path for AFA

Cache types

The architecture leverages three types of caches:

- **Flash cache** is present only in hybrid systems that use SSDs as a caching layer that holds copies of blocks. Flash cache is also organized as an LFS, using the same principles mentioned previously but with different segment size. Blocks in the flash cache are stored, compressed, deduplicated, and encrypted (if encryption is used).

The implementation makes it possible to dynamically increase or decrease a platform’s cache size without having to worry about forming, managing, or modifying underlined SSD RAID groups. Furthermore, an SSD failure does not require data recovery because data blocks are also stored permanently on a disk. Finally, indexes are also pinned in the flash cache and cached in memcache.

- **Memcache** is a portion of DRAM that serves as an index caching layer. It is present in both AFA and hybrid platforms.
- **Storage class memory (SCM)** is supported as of the HPE NimbleOS 5.2.x release. These Intel® Optane™ PCIe SSDs (3D XPoint™) substantially improve read latency as compared to typical SAS/SATA and NVMe NAND SSDs. SCM devices can be present in AFAs and act as a DRAM extension layer for index and block caching. The design gives rise to a new hybrid 2.0 architecture that leverages DRAM and SCM as caching layers while using back-end SSDs as a persistent storage layer.

The result is a tenfold improvement in read latency as compared to typical AFAs that use NVMe NAND SSDs and uses much-improved cache lookup algorithms to accommodate the extremely low latency of SCM. Figure 14 shows latencies for various media types.

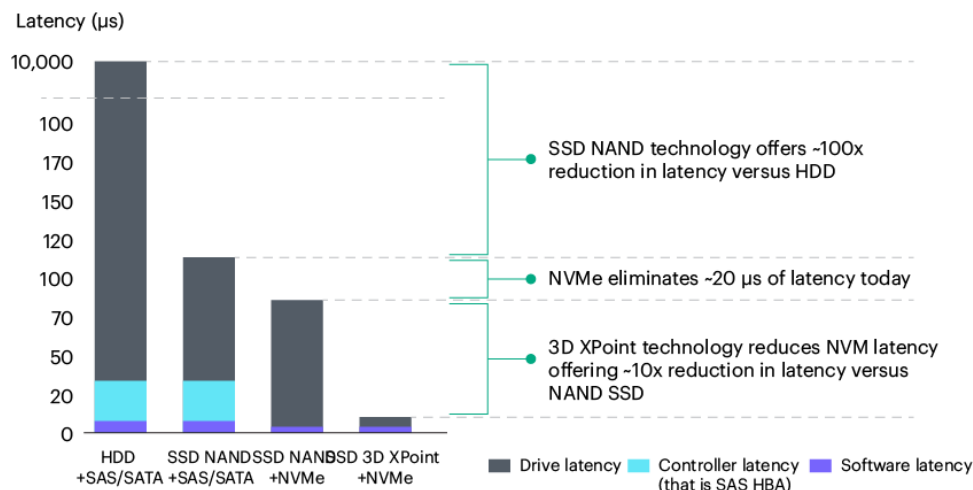


Figure 14. Performance comparisons: SCM (3D XPoint) vs. SAS/SATA NAND vs. NVMe NAND (graphic courtesy: Intel®)

Caching in hybrid platforms

Broadly speaking, blocks that are written to HDDs are cached in the SSD flash cache layer. A read miss causes the block to enter flash cache. CASL uses two methods to determine the cache worthiness of a block:

- **Heuristics:** By default, sequential read/write streams are not cached. Caching these streams might pollute the cache with unwanted blocks and create block churn, causing useful blocks to be evicted. In contrast, random read/write streams are always cached.
- **Per-volume caching policies:** There are four selectable caching options:
 - **Normal:** The normal policy uses heuristics to determine what should be cached.
 - **Pinned:** The entire volume resides in the cache, so there is a guaranteed 100% cache hit ratio.
 - **Aggressive:** The aggressive policy ignores heuristics and caches everything, including sequential I/O (CLI option only). However, unlike the pinned caching policy, cached blocks can be evicted.
 - **Cache disable:** This policy provides the ability to disable caching on specific volumes.

Block eviction in hybrid platforms

After a block is inserted into the flash cache, a process called **access-based eviction (ABE)** identifies potential eviction candidates. ABE is a heatmap-based mechanism that maintains per-block temperature. At insertion, each block is assigned an initial temperature. The range of block temperatures can indicate a block's usefulness in a cache. A block's temperature increases with every access, but it can also decrease (cool off) when the block is not accessed.

Furthermore, if GC detects that enough flash cache is not available, it reduces the temperature of all blocks in a segment in the SSD flash cache by one. The heatmap is maintained in memory along with an on-disk copy. Finally, the heatmap allows the architecture to pin volumes and indexes in a cache, which in turn enables applications to achieve very high caching ratios.

Putting it all together

A fully sequential layout accomplishes three goals:

- Sequentializes random writes (converts many random writes into a more efficient small number of large sequential operations)
- Enables fast draining of memory write buffers to provide fast, deterministic write performance
- Accelerates parity-based RAID

For example, in a RAID stripe with 20 data and three parity chunks without a fully sequential layout, every random write might result in the updating of three parity chunks and multiplying the number of writes by four.

However, from a total I/O perspective, the effect would be even more dramatic because the ratio of writes to the total amount of I/O performed by the back end would increase by a staggering eight times.

In the previous example, using a full sequential layout, parity updates are amortized across the whole stripe, decreasing the write amplification factor to $(20+3)/20$, or 1.15.

The CASL Triple Parity+ RAID implementation provides continuous data access even in the event of a simultaneous triple drive failure. It also provides recovery mechanisms that allow access to data from UREs even when there is no redundancy in the stripe.

Because CASL supports a fully sequential layout, which supports variable block sizes, blocks are concatenated into segments without regard for aligning block boundaries with sector boundaries in underlying storage. In turn, variable block sizes support fast inline, variable block compression without read-modify-write implications, inline variable block deduplication, and application-specific tuning.

Very importantly, variable block sizes provide the ability to append to each block, a block identifier (SBN), and a checksum. Having the SBN and the checksum enables the platform to detect and recover from insidious error conditions, including block corruption, lost writes, misdirected writes, and misdirected reads.

From supporting all-flash accelerators, to spinning disks, to all-flash arrays—as the underlying media changes—CASL continues to easily adapt its cache-centric architecture to new technologies, with the next journey being SCM and NVMe.

HPE Nimble Storage terminology guide

Concept name	Description
Controller	Physical hardware consists of CPU, memory, NVDIMM, PCIe slots, host ports, and disk ports.
Head shelf	4U base enclosure with dual controllers and disks.
Disk shelf	4U external disk enclosure with 48 SSD slots for AFA or 21 HDDs and six SSDs for HFA.
NVDIMM	A byte addressable NVDIMM consisting of DDR4, backed by flash and a supercapacitor. It is used as a write buffer.
Pool	A collection of one or more RAID groups and/or physical HPE Nimble Storage arrays.
Multiarray pool	Volumes striped across multiple HPE Nimble Storage arrays in a group.
Merge pool	Combining two or more HPE Nimble Storage array pools into a single virtual pool.
Group	A logical entity used to represent a collection of physical arrays. Arrays in a group can be managed together. Each HPE Nimble Storage array inherently belongs in its group.
Group leader (GL)	In a multiarray group, the array serving as the GL maintains configuration data and hosts the management IP address. The first array in the group is always the designated GL.
Multiarray group	A group with more than one HPE Nimble Storage array.
Merge group	Combining two or more HPE Nimble Storage arrays in a single group.
RAID layout	Data protection method for AFA and hybrid—AFA: 20D+3P, HFA: 18+3.

Segment	Logical layer on top of physical disks, which divides disks into slots each of which becomes a full RAID stripe.
Segment chunk	Represents the amount of data written to each drive.
Volume	Linear block space hosts can read and write.
VMware® Virtual Volumes (vVols)	vVols allows the management of virtual machines and their data (VMDKs) without having to understand the underlined storage layer. Both regular volumes and vVols can co-exist in the same group or pool.
Volume reserve	Amount of storage that is thick provisioned. Zero percent indicates thin provisioning whereas 100% indicates thick provisioning.
Volume pinning	The ability to keep the active blocks of a volume in cache as well as writing them to disk. Pinning guarantees a 100% read cache hit rate. It is available on HPE Nimble Storage HFA.
Volume collection (VolColl)	Sets of volumes that share the same data protection characteristics. For disaster recovery, all volumes in a volume collection can simultaneously fail over to a replication partner.
Volume move	The method used to migrate an HPE Nimble Storage volume from one pool onto another in a multiarray group.
Stand-alone volume	A volume that does not belong to any volume collection or is not associated with a protection template.
Snapshot	A virtual point-in-time copy of a volume that uses redirect-on-write technology to avoid a performance impact.
Hidden snapshot	For any unprotected volume (not in a volume collection), the array automatically generates and rotates a hidden snapshot hourly as a recovery point.
Unmanaged snapshot	A snapshot that is not linked to a protection schedule
Online snapshot	Enables host access to snapshots as volumes.
Snapshot collection (SnapColl)	Set of snapshots created on the volume collection at a given point in time.
HPE Nimble Storage snapshot framework	Allows the custom host or application-based plug-ins, to customer presnapshot and postsnapshot tasks for application without VSS-awareness.
Clone	A space-efficient point-in-time copy of a volume from a snapshot with read/write host access.
Folder	An organizational construct like a directory. It is used to organize volumes by tenant or application. Capacity thresholds and QoS can also be set at the folder level.
Quality of service (QoS)	Applied at the volume and/or folder level to control IOPS and/or throughput.
Performance policy	Defines attributes and the way data is stored in the array on a per-volume basis. For example, block size, compression, encryption, and deduplication.
Protection template	Sets of defined schedules and retention policies used for data protection.
Domain	The area where deduplication takes place. Volumes that share blocks are grouped in the same deduplication domain.
Initiator group	A logical construct that contains host FC WWPNs or iSCSI IQNs. Volumes are exported to hosts using initiator groups.
Upstream array	Source HPE Nimble Storage array in a replication relationship.
Downstream array	Destination HPE Nimble Storage array in a replication relationship.
Async replication	Asynchronous snapshot-based replication.
Sync replication	Protects from array or site failures with zero data loss.
Peer Persistence	Safeguards from array or site failures with zero data loss and implements an automatic switchover (ASO).
Replication partner	An array in a replication relationship.
Replica	A copy of a volume on a different HPE Nimble Storage array.

Discovery IP	An iSCSI discovery target portal on the array. A host uses the discovery IP address to discover all available iSCSI targets on the array.
iSCSI group-scoped target	Single iSCSI IQN with multiple volumes behind the target (default with HPE NimbleOS 5.0).
iSCSI volume-scoped target	Each iSCSI volume is also an iSCSI target with its IQN (pre-HPE NimbleOS 5.0).
Adaptive compression	The ability to leverage multiple compression algorithms (LZ4/LZO) based on system resource availability.
WebUI	Onboard HTML5-based array management and monitoring.
HPE InfoSight	Cloud-based predictive analytics platform with multiple recommendation engines for monitoring single or multiple arrays.
Cross-stack analytics	Feature of HPE InfoSight, which provides end-to-end correlational analytics for VMware environments.
StackVision	Feature of HPE InfoSight, which provides end-to-end correlational analytics for Microsoft Hyper-V environments.
HPE Nimble Storage Connection Manager (NCM)	Host-based utility, which manages iSCSI sessions and sets appropriate iSCSI timeouts. On certain hosts, it also sets the multipathing of PSP and SATP (VMware).
HPE Nimble Storage Protection Manager (NPM)	An interface between an HPE Nimble Storage array and the native interfaces of the VMware host or guest OS. This places the OS and application data into a consistent state that is safe for backup and recovery.
Windows Toolkit for HPE Nimble Storage	A Windows integration package, which contains volume shadow copy service (VSS) for Microsoft Exchange, SQL, Hyper-V, VMware vVols, iSCSI, Fibre Channel connection management, HPE Nimble Storage Setup Manager (NSM), the HPE Nimble Storage Device-Specific Module (NDSM) for MPIO, the HPE Nimble Storage Diagnostic Utility, and the HPE Nimble Storage and Microsoft Windows PowerShell module.
HPE Nimble Storage Setup Manager (NSM)	Allows for quick discovery and setup of an HPE Nimble Storage array from a Windows host or laptop. NSM is part of the NWT.
Linux Toolkit for HPE Nimble Storage	A Linux integration package, which includes the NCM, HPE Nimble Storage Oracle Application Data Manager, and Docker Volume plug-in for HPE Nimble Storage.
Oracle Application Data Manager for HPE Nimble Storage	Part of the NLT—Allows the use of Oracle consistent snapshots as backups and recovery points, without the need to understand the mapping between the database and objects on an array.

Learn more at

[HPE.com/us/en/storage/nimble.html](https://hpe.com/us/en/storage/nimble.html)

Visit HPE.com

[Chat now](#)

© Copyright 2025 Hewlett Packard Enterprise Development LP. The information contained herein is subject to change without notice. The only warranties Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

Docker is a trademark or registered trademark of Docker, Inc. in the United States and/or other countries. Google is a registered trademark of Google LLC. Intel and Intel Optane are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Linux is the registered trademark of Linus Torvalds in the U.S. and other countries. Hyper-V, Microsoft, PowerShell, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. VMware is a registered trademark or trademark of VMware, Inc. and its subsidiaries in the United States and other jurisdictions. Oracle is a registered trademark of Oracle and/or its affiliates. All third-party marks are property of their respective owners.

a50002410ENW, Rev. 2

HEWLETT PACKARD ENTERPRISE

hpe.com

