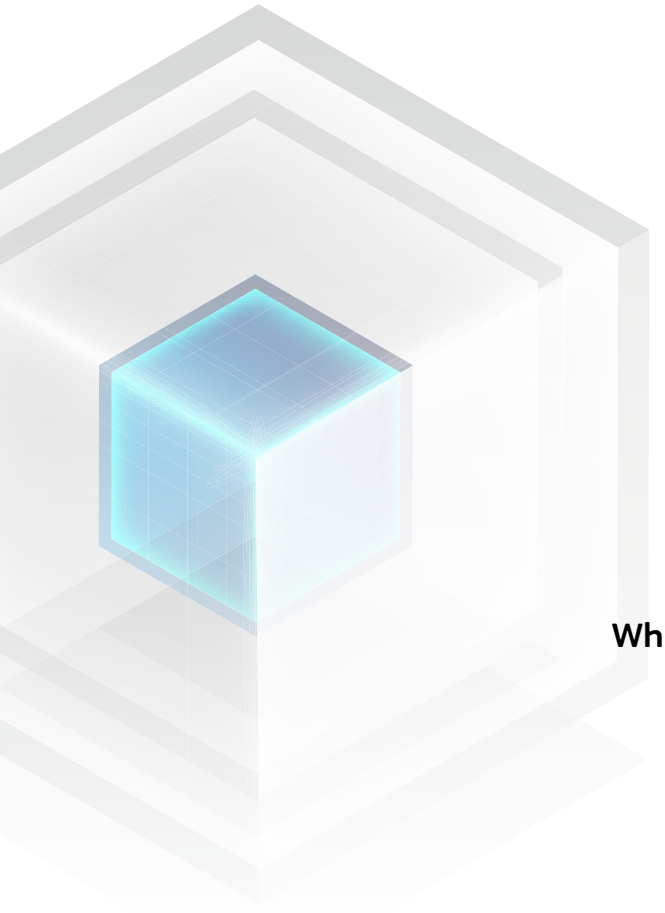
DATA REDUCTION

ALL DATA REDUCTION IS NOT CREATED EQUAL

As data reduction became more commonplace, too many storage and backup administrators started viewing compression and deduplication as checkbox features, ignoring the many approaches to data reduction and their diverse degrees of reduction. This paper explores how vendors implement data reduction by examining and comparing the compression and deduplication in Dell's market-leading PowerProtectDD appliances, formerly known as Data Domain, to those techniques and similarity-based reduction in VAST's Universal Storage.



White Paper | July 2022

TABLE OF CONTENTS

Introduction **3**

Data Compression 4

Storage Allocation Blocks and Data Compression 4

Data Deduplication 5

Chunking 5

Fixed, Variable and Adaptive Chunking 6

The Rehydration Tax 8

Deduplication Scope 9

Multiple Deduplication Realms Reduce Reduction 9

Comparing VAST and PowerProtectDD **11**

PowerProtectDD Summary 11

VAST Universal Storage Summary 11

Solution Summary **12**

Comparing Compression 13

PowerProtectDD Compression 13

VAST Compression 13

Data Reduction Chunking and Scope 14

PowerProtectDD Chunking and Scope 14

VAST Chunking and Scope 14

Similarity Reduction 15

Data Reduction in Action **16**

VAST Data vs Dell EMC PowerProtectDD Virtual Edition (DDVE) 16

Conclusion **16**

Appendix: Data Reduction Testing Details **17**

INTRODUCTION

In the twenty years since Data Domain brought data deduplication to the data center, deduplication and its older brother, lossless compression, have become commonplace across a range of storage solutions. Unfortunately, that has led many users to view data reduction as a check-box item, as if all data reduction was created equal when nothing could be further from the truth.

This paper compares the data reduction techniques used by Dell's PowerProtectDD appliances (Data Domain's direct descendants) to VAST Data's Similarity Reduction exploring the trade-offs Dell takes in their reduction and how VAST avoids those trade-offs to deliver what we guarantee is the best data reduction in the industry[1].

Before we dive deep into the differences between PowerProtectDD's and VAST's data reduction technologies, let's review how data compression, data deduplication, and VAST's global similarity reduction work and the choices system architects face when implementing each. At its most basic, data reduction is a technique that trades compute power for storage capacity; when a storage architect can trade a little compute for a significant increase in effective capacity, it's a valuable technique.

1 VAST Data guarantees that a VAST Universal Storage system will reduce any set of unencrypted data to use less space than any other storage solution. For details see: VAST Data Zero Compromises Guarantee

DATA COMPRESSION

Most data compression algorithms are based on two primary techniques. First, dictionary compression methods like the LZ family of compression algorithms, find repeated patterns across a dataset and replace them with smaller symbols, building a dictionary of symbols and the strings they represent. The second technique, Huffman coding eliminates entropy at the bit level by aligning the number of bits to store each value with the value's frequency so common values are stored with fewer bits.

The degree to which any data set can be compressed is a function of the compression algorithm used, and the computation time the system allocates to compression.

With modern compression algorithms, decompression is much less computationally expensive than compression. Decompression is in fact so computationally inexpensive that most storage systems provide better read performance with compression on than with compression off. This is especially true for hard drives, where the additional bandwidth of reading compressed data can be significant.

The most significant limitation of most compression is its limited scope. Most dictionary coding algorithms only build their dictionaries across 64KB or less, starting a new block and storing another dictionary every 64 KB.

Data-Aware Compression

Dictionary compression and Huffman coding are general-purpose methods designed to reduce any type of data. Data-aware methods, including MP3 and AAC, can be even more effective at compression by taking advantage of their knowledge of the data itself.

For example, a data-aware algorithm such as MPEG can reduce data more effectively by storing keyframes and the changes from frame to frame instead of saving every frame as an independent still image. Some compression methods, including MP3, use lossy compression where the data doesn't decompress to the original ones and zeros, but to something that sounds like the original data when played.

Because storage systems must return precisely the same bits the user saved, any compression they perform, data-aware or general-purpose, absolutely must be lossless.

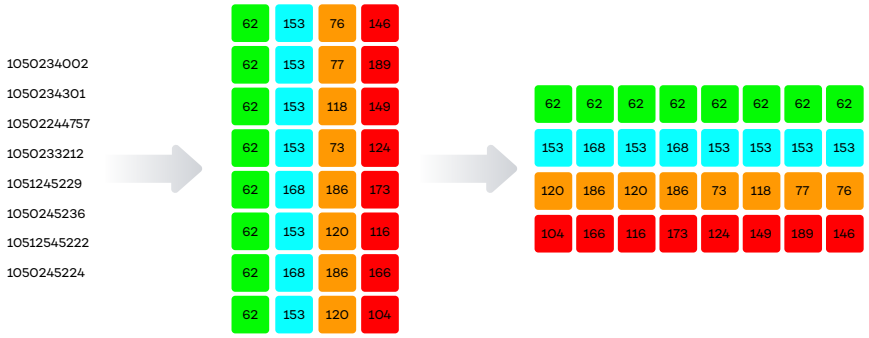
VAST Puts Data-Aware Compression in Action

VAST systems include data-aware compression algorithms designed to improve the reduction of the type of numeric data generated by instruments, from simple temperature and pressure sensors to the seismic surveys used to find oil and gas.

This real-world data generally has a lot of entropy (randomness), but because sensors frequently return values within a range, that entropy is mainly contained in the lower order bits of each value.

Numeric Optimization

VAST systems optimize the reduction of numeric data by separately storing each of the four bytes that make up a 32-bit float or a 32-bit integer. Instead of storing all four bytes of each reading together the system stores the most significant byte of all the values in the block followed by the next most significant byte of all the values and so on as shown in the figure below.



Floating Point Optimization

Since most instrument, or stock trade, data varies over a limited range, the area containing the most significant bits will be very compressible, holding a small range of values that repeat often. Even if the areas storing the third and low-order bytes only compress a little, the net result is a significant increase in overall compression.

Delta Encoding

VAST's second data-aware reduction technique is delta encoding. Delta encoding steals an idea from MPEG's book and saves the difference between each value and the preceding value in the way MPEG stores each frame as changes from the preceding frame. If we apply delta encoding to our example temperature data, the system stores the difference between each reading. Combine this with the trick of separating the four bytes of the 32-bit values into different blocks, and the system only stores the changes in those high-order bytes. Because temperatures change relatively slowly, that difference will be 0 for many samples.



Delta encoding is also effective on uncompressed images including .TIFF files created by archival scanning and some DICOM images in medical archives and is included in the .PNG file format.

Choosing A Compression Method

Most data-aware data compression algorithms perform compression in the application as part of the encoding process. For example, when you save a video in MP4 format, your video editor compresses it before saving it to flash,.

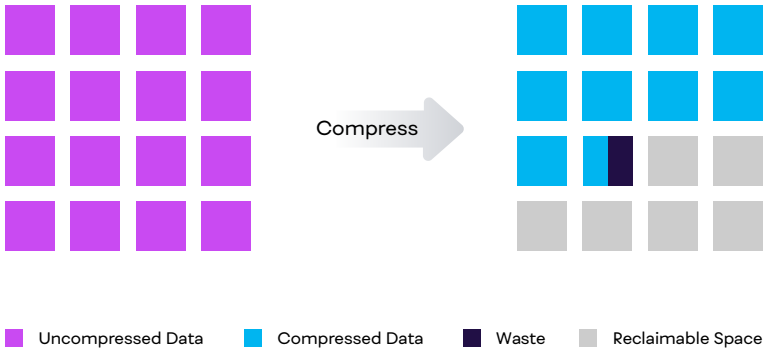
Data-aware compression is a bit more complicated for storage systems; they have to figure out what compression method to use without being explicitly told. Users may give them hints through file extensions, but new file formats crop up all the time, making file extensions an unreliable indicator.

VAST systems automatically determine if data should be compressed with ZSTD or with some combination of floating-point optimization, delta-optimization, and ZSTD by taking a small sample of each element and compressing it with each of our data-aware compression methods. The system compresses each element with the method most effective for its contents.

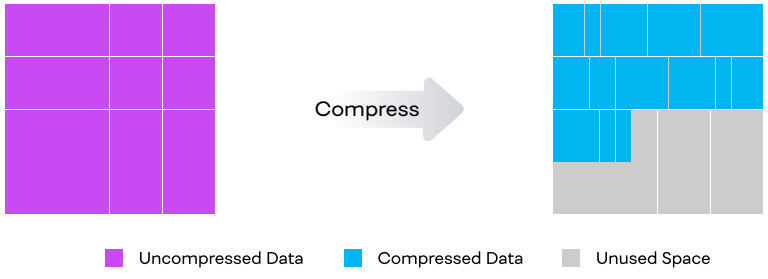
STORAGE ALLOCATION BLOCKS AND DATA COMPRESSION

Most storage systems, including NetApp's OnTap and Dell's PowerScale, manage their data in fixed-size blocks. The pointers in their metadata point to a logical block, and since all the blocks are 4KB or 64KB, the system knows where the data is. If the data in a 4KB block compresses to 2344 bytes, the system still must use a whole 4096-byte block to hold it. Even though the data is "compressed", no storage would be saved.

The solution these vendors have settled on is to compress the data in some group of blocks, typically 8 or 16 blocks. When the 64KB of data in 16 4KB blocks compresses to 37631 , the system stores that in 10 4KB blocks, releasing 6 blocks for other data. While this reduces required capacity by ~37%, it's still inefficient because it uses 40960 bytes of storage to hold 37631 bytes of data.



Instead of managing data in fixed-size blocks, the VAST Element Store uses byte-oriented metadata to manage blocks of any size without having to align Element Store allocation blocks with drive logical block addressing (LBA) boundaries. If a 32KB write shrinks to 14501 bytes, the VAST Element Store writes 14501 bytes to flash, and the next block starts at byte 14502. The result is that compressed data is stored waste-free, providing more usable storage capacity.



DATA DEDUPLICATION

Data deduplication recognizes chunks of identical data within a dataset and, through clever use of metadata pointers, stores only one copy of each data chunk. There are many details beyond that simple description, and getting the details right can make a big difference for some datasets. Two primary parameters determine how effective a deduplication method will be:

- Chunking: How the system divides the data into chunks
- Deduplication Realm: How much data the system can deduplicate across

CHUNKING

Because deduplication relies on identifying multiple logical data chunks that contain the same data, how the system defines those chunks can significantly impact how well it identifies duplicates to eliminate.

From an efficiency point of view, smaller chunks are better. When a user or an application performs a small write, the system has to store one deduplication chunk. For example, when a user makes a 4KB change to a file, a system that deduplicates on 8KB chunks will store 8KB, but a system that deduplicates on 128KB chunks will store 128KB (16 times as much).

The problem is that using smaller chunks to store the same amount of data means more chunks and more metadata to manage those chunks. Systems that deduplicate data inline must very quickly identify whether any new data written to them is unique or a duplicate of some existing data chunk. That means they need to keep the hash table of previously seen data in DRAM where it can be rapidly searched.

The result of this limitation is that storage architects must balance the efficiency of small chunks against the higher compute and much higher memory costs of small chunks. Small blocks also exacerbate the IOPS expansion problem inherent to reading deduplicated data from hard drives.

Sequential reads from a deduplicated data store such as restores, must re-assemble the original data from chunks stored across the system, a process known as rehydration. So a 1MB read creates 32 read I/Os if the system deduplicates in 32KB chunks. The same 1MB read becomes 128 I/Os for a system using 8KB chunks. Since a 7200 RPM hard drive can only perform ~100 I/Os per second (IOPS), this "rehydration tax" is the limiting factor in read/restore performance for most disk-based systems. All-flash systems can easily pay the "rehydration tax" from the hundreds of thousands of IOPS (580,000 for the P4326 we use) provided by each of their SSDs. We'll discuss this "rehydration tax" in more detail below.

FIXED, VARIABLE AND ADAPTIVE CHUNKING

While using smaller chunks makes fixed-size deduplication more efficient, any fixed chunk deduplication method remains blind to data repetitions that don't align with their arbitrary chunk boundaries. To see just how badly a small change can affect fixed chunk data deduplication, let's look at Dicken's famous opening to A Tale of Two Cities. In a late draft it read:

> *It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of stupidity, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us...*

A deduplication system that stores the data in 16 character (byte) blocks would break that up as shown below:

It was the best	of times, it was	the worst of ti	mes, it was the
age of wisdom, i	t was the age of	stupidity, it w	as the epoch of
belief, it was t	he epoch of incr	edulity, it was	the season of Li
ght, it was the	season of Darkne	ss, it was the s	pring of hope, i

Then Mr. Dickens decided that foolishness was a better word than stupidity, so he changed it and saved a new file. When the 16-byte fixed-size deduplication engine examines this data, it breaks it up like this:

It·was·the·best·	of·times,·it·was	the·worst·of·ti	mes,·it·was·the
age·of·wisdom,·i	t·was·the·age·of	·foolishness,·it	·was·the·epoch·o
f·belief,·it·was	·the·epoch·of·in	credulity,·it·wa	s·the·season·of·
Light,·it·was·th	e·season·of·Dark	ness,·it·was·the	·spring·of·hope,

The first 96 characters are the same as in the original version, creating the duplicate chunks outlined in blue. Because 'foolishness' at 11 letters is 2 characters longer than 'stupidity,' every subsequent chunk in the book is shifted down 2 bytes and, therefore, different from the original file. This forces the system to store all the chunks outlined in red.

As its name implies, adaptive chunking adapts the size of each data chunk based on the data's contents. The system takes moving samples across the data to be divided into chunks and uses some state of that moving sample to determine where chunks begin and end.

Let's apply an adaptive chunking algorithm to Dickens to see how much more effective adaptive chunking can be. We'll start a new chunk every time we see the word it while keeping chunks between 8 and 32 bytes.

It·was·the·best·of·times,·	it·was·the·worst·of·times,·
it·was·the·age·of·wisdom,·	it·was·the·age·of·foolishness,·
it·was·the·epoch·of·belief,·	it·was·the·epoch·of·incredulity,·
it·was·the·season·of·Light,·	it·was·the·season·of·Darkness,·
it·was·the·spring·of·hope,·	

Now only the chunk containing the change is affected. The chunks line back up at the next it, and instead of storing the remaining 788 KB of ***A Tale of Two Cities*** as changed data, the system can store just one new 31-byte chunk.

This adaptive chunking scheme not only achieves higher deduplication rates than smaller fixed-size chunks, it also uses less metadata. Some systems attempt deduplication at multiple block sizes (EG 4, 8, 16 and 32 KB) storing the largest block that duplicates existing data to minimize metadata.

THE REHYDRATION TAX

Conventional storage systems that don't deduplicate data generally store data in roughly the same form as it is written. When a backup application writes 40 GB of data sequentially, that data is stored sequentially on the system's disks. When that data is restored, the system can read that data from the disks sequentially, and since disk drives aren't bad at sequential reads, deliver the data quickly.

Things get more complicated when we add data deduplication to the mix. On a deduplicated datastore, that 40GB of data won't be written sequentially; it will be broken into blocks. The blocks that contain new data will be written close together, but the blocks that duplicate existing data will be replaced by pointers to the existing data .

When it comes time to restore that 40GB, the storage system follows those pointers and moves the heads of its disk drives to read blocks of data that are somewhat randomly scattered across the system's disks. This rehydration process extracts a performance tax by turning the large, sequential read requests from the backup application into small block random reads, as the system reconstructs the original data from its deduplicated chunks across the drives .

That rehydration tax means a disk-based purpose-built backup appliance (PBBA) like PowerProtectDD can only feed restore jobs 1/3rd to 1/5th as fast as it can accept backups. As a result, the makers of disk-based PBBAs, including Dell, have stopped publishing restore (read) performance numbers for their systems.

DEDUPLICATION SCOPE

As we saw in the data chunking discussion above, deduplicating with small blocks results in better data reduction, but tracking millions of small chunks requires a lot of metadata. In turn, managing that metadata requires a lot of compute cycles and expensive DRAM.

Conventional deduplication systems like PowerProtectDD Appliances hold their hash lookup tables in memory for speed. This approach limits the scope of their deduplication realms to the amount of data their dual controllers can manage and hash in memory, (1-2PB for the biggest appliances).

Maintaining this deduplication metadata has been especially challenging for the designers of scale-out storage systems. Even if shared-nothing[2] nodes only cache portions of the deduplication metadata, keeping that cached data coherent across many nodes creates so much CPU and network overhead that it makes inline deduplication impractical. Scale-out systems like Dell's PowerScale have made compromises like maintaining independent deduplication hash data in each node, effectively making each node an independent deduplication realm.

Dividing the system, or data, into multiple deduplication realms by volume, file system, backup repository, media server or even backup job is a common way for a deduplication system to minimize its compute requirements. The system only needs to load the hash table for the active deduplication realm into memory, reducing the amount of expensive memory required. While this approach addresses the issue of memory utilization, it creates a different challenge as detailed below.

MULTIPLE DEDUPLICATION REALMS REDUCE REDUCTION

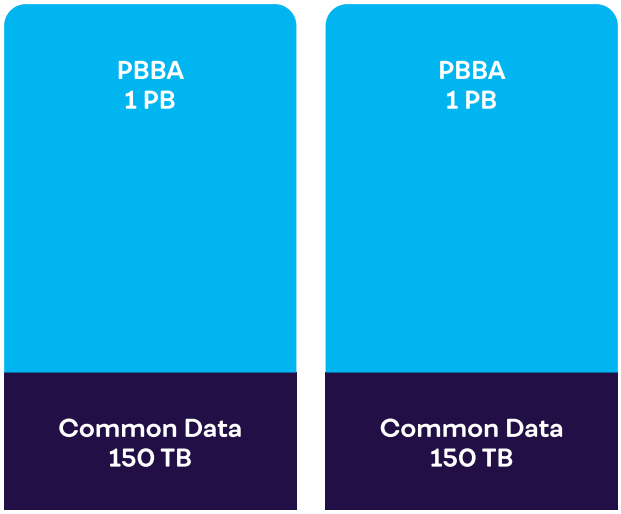
While creating multiple deduplication realms makes life easier for the storage system architect, the user ultimately pays the price by storing multiple copies of whatever data they send to more than one realm. It's easy to think that if a dataset, especially a backup dataset, reduces 3:1 or 10:1 when written to a single deduplication realm, then all of the data reduces 3:1 or 10:1.

2 The Shared-Nothing architecture is described in depth here: https://vastdata.com/blog/exploring-shared-nothing-storage-part-1-what-is-shared-nothing/

In reality, most reduceable data sets break down more like this:

- 30-80% of the data is unique
- 10-40% of the data duplicates a small number of times (2-10)
- 5-30% of the data duplicates many times (100s to 1000s)

If a customer has 500 Windows Server 2019 VMs, when they back up those VMs, Windows itself will be duplicated 500 times. If backed up to a single deduplication realm, 1 copy of Windows will be stored. If backed up to multiple deduplication realms, a copy of Windows has to be saved in each and every realm.



Net Total 1850 TB

In the case of the PowerProtectDD Appliances, when the customer adds a second PowerProtectDD appliance, they add a second deduplication realm. If 15% of the customer's data repeats like those Windows VMs, they will only be able to store 85% more data because 15% of the new system will be filled with duplicate copies of data already stored on the first appliance. The result is that the data that used to reduce 10:1 now only reduces 9:1 or so. Each PowerProtectDD may report 10:1, but the aggregate deduplication ratio across the two appliances isn't reflected in each appliance's dashboard.

COMPARING VAST AND PowerProtectDD

PowerProtectDD SUMMARY

Dell EMC's PowerProtectDD PBBAs are the descendants of the Data Domain PBBAs EMC acquired in 2009. Dell currently makes six PowerProtectDD models with usable capacities ranging from 4TB to 1.9PB and backup speeds from 4.2 to 41TB/hr plus a virtual machine edition.

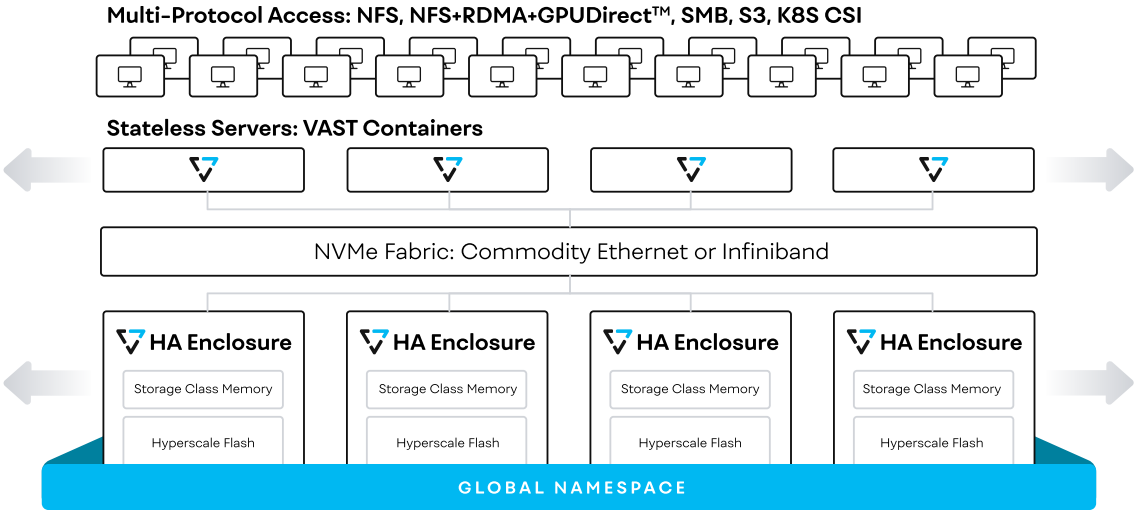
PowerProtectDD appliances use a dual controller, active-standby architecture using SAS-attached 7200 RPM disks for capacity and a minimal amount of flash for metadata storage. As PBBAs, PowerProtectDD appliances support backup-specific access methods like NDMP and tape library emulation in addition to NFS and CIFs (SMB 1.0).

PowerProtectDD appliances perform both compression and deduplication inline as data is written to disk.

VAST UNIVERSAL STORAGE SUMMARY

VAST's Universal Storage system is an all-flash file and object storage system using VAST's DASE (Disaggregated Shared Everything) scale-out architecture to scale from hundreds of TB to hundreds of PB in a single cluster/namespace. As shown below, the VAST systems use stateless protocol servers to process user requests and manage the global namespace that's stored across the Storage Class Memory (SCM) and Quad-Level Cell (QLC) SSDs in the HA Enclosures. This allows VAST users to scale their cluster's capacity and performance by adding enclosures. In addition, this design enables performance and capacity to scale independently.

VAST's Element Store includes several features to maximize storage efficiency and minimize flash wear, including locally-decodable erasure codes that protect against as many as four device failures with just 2.7% overhead and VAST's Similarity-based data reduction.



VAST systems reduce data as it is migrated from SCM write buffers to hyperscale flash.

SOLUTION SUMMARY

	Dell EMC PowerProtectDD Series	VAST Data Universal Storage
System type	Purpose-Built Backup Appliance	Scale-Out File and Object Storage
Architecture	Single or Dual Controller	DASE Scale-Out Archtecture
Protocols	NFS, CIFS (SMB 1), NDMP, Virtual Tape Library	NFS, SMB 2, S3
Max Useable Capacity	32 TB - 1.5 PB by model	300 TB to 100s of PB by cluster size
Max Backup Throughput	4.2 TB/hr-41 TB/hr by model 1.2-11.4 GB/s	18 TB/hr - >1 PB/hr by cluster size 5-100s GB/s
Max Restore Throughput	1.4-14 TB/hr (estimated) 0.4-4 GB/s	90->5 PB/hr 40 GB/s- PB/s
Data Reduction Scope	Realm per Appliance	Realm per Cluster
Compression Algorythm(s)	LZ, gzfast, GZIP (DEFLATE)	Zstandard
Compression Time	Inline, limited time for compression	On-migration to QLC, deep compression without latency impact
Compression Granularity	Block	Byte
Chunk size	Variable 8-16KB	Adaptive 16-64 KB
Similar (non-identical) Chunk Reduction	No	Yes

COMPARING COMPRESSION

PowerProtectDD Compression

PowerProtectDD appliances compress data immediately as it is received with only a small NVRAM buffer to hold uncompressed data. This small buffer gives the PowerProtectDD appliance little time to compress the data and write it to disk before the buffer fills, requiring PowerProtectDD appliances to use compression algorithms that compress data quickly.

The default compression algorithm on PowerProtectDD appliances has historically been LZ, a fast dictionary compression algorithm. The latest PowerProtectDD appliances include a hardware compression card that compresses data using gzfast or the optional gz algorithm. Gzfast and gz are based on the DEFLATE method used in the GZIP utility that adds varying levels and Huffman coding to the dictionary compression.

With the addition of hardware compression, today's PowerProtectDD appliances can achieve 15-30% better reduction at their full rated speed than earlier generations.

VAST Compression

In VAST's DASE architecture, data is written to an SCM write-buffer and acknowledged to the client. Data services, like data reduction, are performed during the asynchronous process that migrates data from the SCM write buffer and erasure codes that data across the hyperscale SSDs in the cluster.

Because VAST systems have terabytes of write buffer, as long as the system migrates data as fast as the users write to it, the time it takes to compress any chunk of data doesn't matter. The migration process runs in parallel across all the protocol servers (CNodes) in the cluster, providing plenty of CPU power to compress data with the Zstandard protocol developed initially at Facebook. Zstandard combines dictionary and Huffman coding techniques at multiple levels to optimize between CPU time/cycles and compression. VAST systems also optimize the reduction of numerical data as described in the Data-Aware Compression section above.

Advantage: VAST.

DATA REDUCTION CHUNKING AND SCOPE

PowerProtectDD Chunking and Scope

PowerProtectDD appliances chunk data for deduplication and compression into variable-length blocks from 8KB to 16KB, which is relatively fine-grained for a deduplication system. Each PowerProtectDD appliance is an independent deduplication realm, so if users write any data to multiple PowerProtectDD appliances, a copy of that data will be stored on each appliance as described in the deduplication scope section.

VAST Chunking and Scope

VAST systems chunk data into adaptively sized chunks from 16KB to 64KB with an average of 32KB. This is a little coarser-grained than the PowerProtectDD.

A VAST cluster creates a single data reduction realm across the entire namespace created by that cluster. Where PowerProtectDD appliances must hold their deduplication hash tables in controller memory to lookup hash values quickly, VAST stores this and other data reduction metadata in the enclosures' (Dboxes') SCM SSDs, where it is shared by all the cluster's protocol servers (Cnodes).

As enclosures are added to a cluster, they add additional SCM for this metadata, as well as hyperscale flash for the data so the data reduction realm grows with the cluster to hundreds of petabytes or even exabytes.

Advantage: VAST

Similarity Reduction

VAST systems don't just test to see if data chunks are identical; for data deduplication, they go a step further and use an additional set of hash functions to determine if multiple chunks are similar, even when they are not identical. Chunks that are identified as "similar" contain strings in common. VAST systems take advantage of this by compressing similar chunks together using a common compression dictionary. The system then saves the small, compressed chunk without the overhead of a dictionary.

These similarity algorithms and adaptive chunking maximize similarity, reduce misaligned data well, and are responsible for much of the additional reduction achieved when reduced data is written to a VAST system (such as when data is deduplicated by the backup application).

Advantage: VAST

DATA REDUCTION IN ACTION

VAST DATA VS DELL EMC POWERPROTECTDD VIRTUAL EDITION (DDVE)

The following table summarizes the total data reduction of three datasets as measured on a VAST Cluster versus PowerProtectDD DDVE.

Data Type	Initial Size	Metric	DDVE	VAST	Savings
Virtual Machines	128 GiB	Reduced Size	24.4GiB	18.6GiB	24%
		Ratio	5.3	6.83	
Unstructured Data	455 GiB	Reduced Size	348.6GiB	238.89GiB	31%
		Ratio	1.3	1.96	
Database Servers	1.2 TiB	Reduced Size	250GiB	149GiB	40%
		Ratio	4.7	7.88	

The UI showed GiB, Linux listed GB and the VAST cluster showed GB. Regardless of the units used, all the numbers were roughly identical so it is assumed that all numbers were GiB.

CONCLUSION

Data Domain kicked off the backup-to-disk era of data protection with deduplication. In the ensuing twenty years the increasing size and static performance of disk drives have made the rehydration tax that slows restores a significant burden for today's PowerProtectDD. This rehydration tax is imposed by the I/O amplification created by sequential reads from a deduplicated data store, causing PowerProtectDD appliances to restore data at a fraction as fast as they accept backups.

VAST Data is leading the backup-to-flash era with Universal Storage systems that, unlike PowerProtectDD, deliver data for restores several times faster than they accept backups. With ransomware and other concerns increasing the size of restores by orders of magnitude, fast restores are becoming more important than fast backups.

VAST makes this all-flash system affordable compared to the disk-based PowerProtectDD with superior data efficiency. The technologies behind this efficiency include:

- Highly efficient erasure codes with 2.7% overhead vs 20% for PowerProtectDD
- Better Compression
    - More modern ZSTD method
    - Large write-buffer to allow asynchronous, therefore deeper, compression
    - Data-aware compression for numeric data and uncompressed images added
    - More Data-Aware functions in development
- Better Deduplication
    - Adaptive chunking
    - Single deduplication realm scales to 100s of PB vs 1-5PB
- Similarity Reduction
    - Applies ZSTD compression globally
    - Especially effective with reduced data

APPENDIX: DATA REDUCTION TESTING DETAILS

PHYSICAL SETUP

- Vast Cluster - release-4.1.0-sp9-552796
- Dell EMC PowerProtectDD Virtual Edition (DDVE) - 7.7.1.0-1007743

DATASETS

VMware VMs

- 500GB across 10 virtual machines consisting of:
  - 3 x Win 2019
  - 2 x Windows Server 2016
  - 3 x Fedora
  - 2 x Centos 7

Unstructured Data

- 455GB of files including:
  - 4k MP4
  - CAD drawings
  - GIS Geospatial Json
  - Genomics FASTa
  - Deep learning training sets (Imagenet png +jpg)
  - User data (Word, Excel, CSV, RTF, PDF, MP3, text and HTML files)
  - Database Servers

A 1.2 TiB dataset comprised of:

- GIS
- Exported web server logs
- Public records of topological features.

TESTING METHODOLOGY

A Linux host was used as the central point of measurement for VAST and performing copy functions. The source data was housed on a VAST Cluster. A view was created and mounted on the Linux system. To measure the potential data reduction rate within the source data, VAST has an internal script called the "Probe". This is used to ascertain the exact data amount and data reduction on a specific directory. The probe was run on each dataset directory with fixed chunking and adaptive chunking and the DRR was recorded.

To copy the datasets onto the DDVE data directory, an export of that directory was presented out and mounted on the same Linux host. The data was copied en masse from the Vast Cluster directory onto the DDVE export. Total reduction numbers were gathered off the UI.

© 2022 VAST Data, Inc. All rights reserved. All trademarks belong to their respective owners.