Panther V Storage Accelerator: Taking Storage to the Next Level

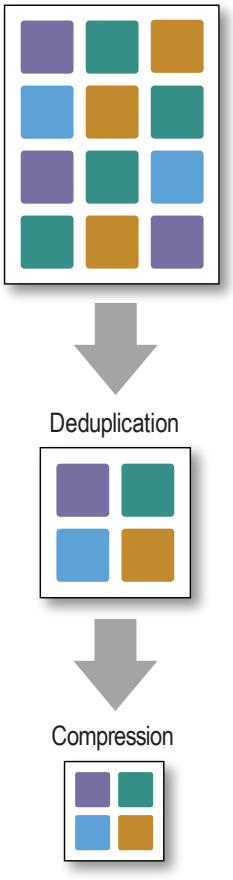
Focus on Data Reduction

WHITE PAPER

Authors:

Mike Ham
Technical Director of Applications Engineering, Storage Accelerators

Pinaki Chanda
Director of Software Engineering



Introduction

The MaxLinear's Panther V storage accelerator (Panther V) has been developed to optimize the performance and efficiency of storage appliances by offloading complex tasks that require long CPU cycles to execute. The key features of Panther V are as follows:

- **Data reduction**
  - High-quality compression to minimize the size of data before storage.
  - Hash generation to drive deduplication to further reduce the size of data before storage.
- **Data integrity**—Panther V provides the most robust protection against data corruption available.
  - Real-time verification (RTV)—All encode operations are automatically followed by a decode operation and verification that the fully decoded data matches the original data.

    Note: Transform commands are not completed until all requested data protection/verification operations are completed successfully.
  - Panther V provides full support for the T10-DIF, T10-DIX, and NVMe Protection Information (PI) data protection functions.
- **Data security**—Panther V supports NIST-certified Advanced Encryption Standard (AES) security standards used for SSL and IPsec transforms. They are optimal for securing data at rest (on the storage device), being transferred to/from the remote client or to/from a backup/replication storage appliance.
- **Highest performance**—Support for 450Gbps throughput with the lowest latencies available.

This document focuses on how Panther V optimizes data reduction ratios (DRR).

Data Reduction

What is data reduction and how does Panther V optimize it?

Data reduction is the process of reducing the physical storage area required to store data while maintaining the ability to regenerate the exact original data. Panther V achieves this by accelerating compression/decompression transform operations and hash generation typically used to drive data deduplication with several patented features.

Compression

What is data compression and why is Panther V data compression better?

Data compression is the process of analyzing data, identifying matching data segments, and replacing the matching data strings with smaller markers or symbols identifying the location of the actual data so that it can be stored in a smaller physical area.

There are two compression classes:

- **Lossy compression—**It does not guarantee exact regeneration of the original data when decompressed.
  Example: MPEG compression for videos and audios where minor differences are not detectable by the human eye or ear.
- **Lossless compression—**It guarantees exact regeneration of the original data when decompressed.

Panther V supports lossless compression algorithms.

There are many lossless compression algorithms, and each has its own advantages. When using software to perform compression, LZ-based algorithms such as LZS, LZO, and LZ4 are most often used because they offer modest compression levels while limiting the CPU overhead to run the algorithms. They typically deliver compression ratios of 1.5:1 to 2:1. This means that after compression, a typical 3MB file size is reduced to an average of 1.5MB to 2MB.

Other compression algorithms can offer higher compression ratios, but at the cost of CPU cycles and lower throughput when run in software.

The general class of dictionary-based compression algorithms (such as Deflate, zlib, gzip, or XP10) compresses data by replacing repetitive patterns with shorter representations based on a predefined or dynamically generated dictionary of symbols or phrases. A dictionary can be previously seen input data. The *history* in a compression algorithm, particularly in data compression techniques such as dictionary-based methods, refers to a parameter that defines how far back in the input data the algorithm can search when trying to find matching patterns. One of the methods to improve the compression ratio of a dictionary-based compression algorithm is to increase the history size or previously seen data to search for matching data strings. The history size can range from 2KB to megabytes depending on the algorithm.

Another method used is to determine the search intensity of these matching data strings within the history. The more intensive the search, the higher the compression ratio.

To improve compression ratios, a delayed matching technique is often used, where the algorithm takes a non-greedy approach to select substrings from the look-ahead buffer to match against the history buffer. Initially, the algorithm finds the longest match starting from the first symbol. In subsequent iterations, it skips the first x symbols (from none to several) and searches for the longest matches in the remaining substring. After all iterations are completed, the longest match candidates are compared to determine the best match, improving the efficiency of capturing longer repeating patterns and ultimately achieving better compression. The number of iterations is called the delayed-match window, and compression efficiency can be improved with a longer delayed-match window.

Most algorithms make a trade-off between history size and search effort to optimize the algorithm for the application. Panther V maximizes the search effort while supporting a large history size and longer delayed-match window to produce compelling results at line rate.

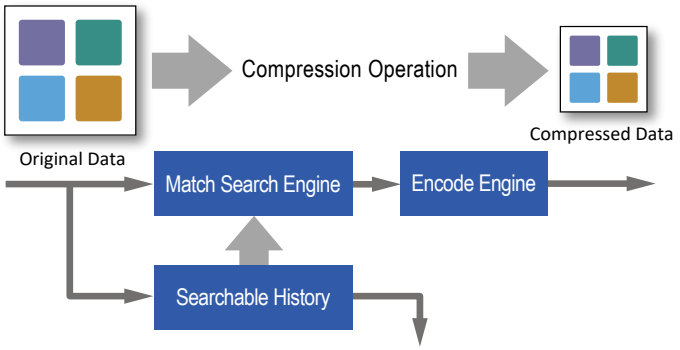


Figure 1: Compression Operation

Panther V supports the following compression algorithms:

- Deflate, gzip, and zlib (gzip and zlib use deflate as their compression algorithm)
  - They offer a typical compression ratio of 4:1 .
  - Panther V supports level 9 with a history of 32KB with a delayed-match window length of 2.
- XP10
  - It offers a typical compression ratio for unstructured data of 4:1.
  - It offers a typical compression ratio for structured data of 5:1.
  - Panther V supports level 6 with a history of 64KB.

Deflate/gzip/zlib level 9, an industry standard compression algorithm with a delayed-match window length of 2, produces a compression ratio of approximately 4:1. This means that the same 3MB file in the previous example would only require 0.75MB of storage space after compression. XP10 level 6 achieves the same 4:1 compression ratio for unstructured data, but when compressing structured data, it can improve the compression ratio to 5:1, further reducing the resulting file size to 0.6MB.

Note: Structured data is data organized in rows and columns. Examples include spreadsheets and Optimized Row Columnar (ORC) files. Unstructured data represents almost everything else.

As you can see, if your data contains structured data or a mix of both, using XP10 offers a significant advantage. Panther V can compress and decompress data much faster than a CPU while offloading the CPU to support other features for which customers will pay extra.

Deduplication

What is data deduplication and how does Panther V accelerate it?

Data deduplication can be considered as a macro form of compression. Compression algorithms search for data copies in a range of a few bytes to thousands of bytes in a data string. Data deduplication searches for exact data matches in thousands to millions of bytes. Typically, the size of a block is a power of 2, starting at 4KB. Thus, an appliance searches for exact matches between data blocks on the storage appliance (across all files) typically using only one block size (4KB, 8KB, 16KB, etc). The smaller the block size, the more likely it is to find matches, since there is less data that must match exactly.

A different approach is needed to find large-block matches, as the compression algorithm techniques described previously would not be efficient or effective. The method used in the industry is to create a unique identifier or digital fingerprint for each block. This is typically done by running a cyclic redundancy check (CRC) or hash algorithm over the data. The resulting CRC, hash, or equivalent operation is the fingerprint of the data block. This value can be used to point to a table entry that is empty when no processed data has generated this value. However, the table entry can be populated with the location of the data block and the number of copies it represents when a block generates this value. If all copies are deleted over time, the entry is deleted.

An entry-level solution might generate a CRC using software to create the fingerprints. This is not optimal because CRCs are not designed to create unique results for different data blocks at this scale. CRCs are likely to cause hotspots in the deduplication table where different data sets generate the same CRC value. This is known as a collision. Collisions have a significant impact on deduplication table management.

A SHA-based hash is a much better algorithm for creating the data fingerprints. They are specifically designed to optimally generate unique results for data sets that do not match. They can populate a deduplication table without hotspots and generally eliminate collisions as long as the table is not overpopulated. Panther V is optimally designed to generate SHA hash results to drive a deduplication table as a single task or while performing other transform operations such as compression.

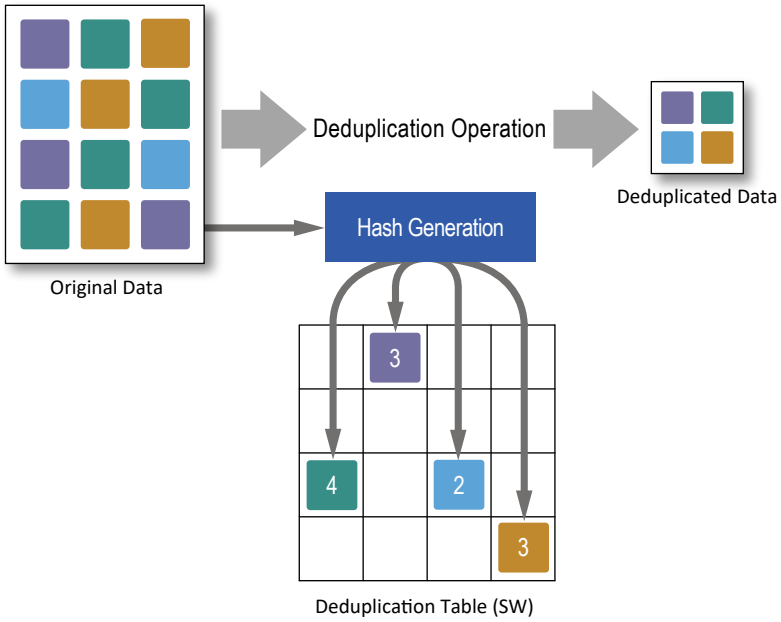


Figure 2: Deduplication Operation

Note: The aim of this document is not to guide you on how to implement deduplication, as it will be performed differently depending on the appliance requirements, capacity and service-level agreements (SLAs). The table methodology above is simplified to provide an overview of the deduplication feature.

Panther V supports an advanced set of capabilities to offload hash generation to drive the deduplication table feature, including the following operations:

- Generate one hash per submitted data block in a single command.
- Generate one hash per programmed sub-block size for a larger submitted data block (for example, submit 64KB of data to process and receive eight hash results, one per 8KB block) in a single command.
  - This optimizes command efficiency by reducing the number of commands from eight to one for the example above, thus reducing overhead and increasing performance and efficiency.
- Panther V supports the MaxHash™ feature which, in addition to generating the hashes described in the previous bullet point, can also generate three additional sets of hashes with programmed start offsets in a single command.
  - This helps identify more matches for additional data reduction opportunities. For more information on MaxHash, refer to the *Panther V Storage Accelerator: Next-Generation MaxHash™ Deduplication White Paper* (006-GWP).

Note: You can combine any of the three hash options mentioned above into a single step and with the compression operation described in the "Compression" section. This is the most efficient method of offloading to the Panther V storage accelerator.

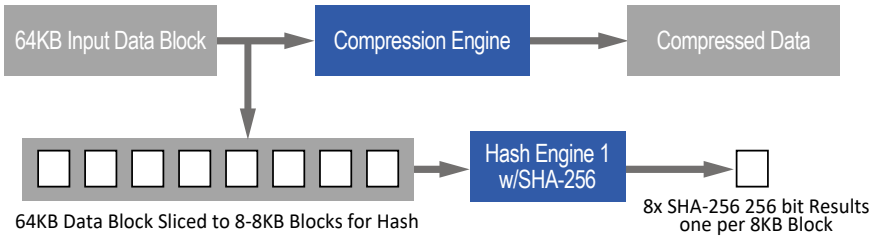


Figure 3: Example of one Command to Compress 64KB of Data and Generate Eight SHA-256 Hashes on 8KB Boundaries

In Figure 3, 64KB of data are submitted in one command so that multiple transform operations can be performed.

To do this, perform the following steps:

1. Compress the 64KB of data and return the resulting compressed data.
2. Split the 64KB data into eight 8KB blocks.
3. Submit each 8KB block separately to the hash engine to generate eight SHA-256 results.
4. Return the eight SHA-256 results for use in deduplication.

By being able to combine these operations, one command replaces what would otherwise have been nine separate commands. This reduces latency, command overhead, CPU memory bandwidth utilization, and increases performance.

With MaxHash, you can program three additional hash engines at different offsets, creating 24 additional hashes in this use case that you can use to search for other potential matches.

When implementing deduplication on a primary storage appliance with the traditional method of using one hash engine, a typical data reduction ratio (DRR) of 2:1 is expected, which reduces the stored data by half. When implemented using MaxHash with four hash engines, it increases the typical DRR to 3:1.

Total Data Reduction Ratio (DRR)

When deduplication and compression are combined, the total DRR is the deduplication ratio multiplied by the compression ratio (reduction ratio).

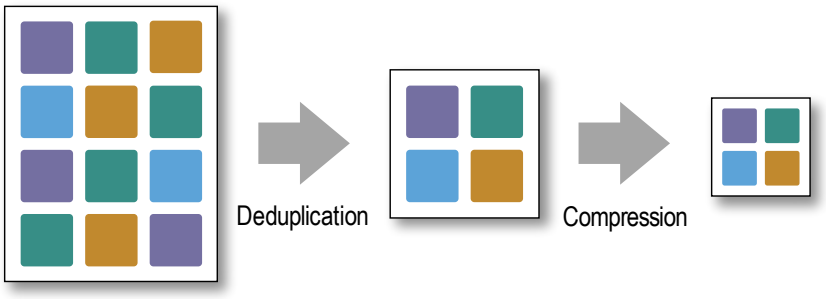


Figure 4: Data Reduction Using Deduplication and Compression

The Panther V DRR for typical data is:

- 12:1 reduction for Deflate/gzip/zlib compression (4:1) and MaxHash-driven deduplication (3:1).
- 15:1 reduction for XP10 compression for structured data (5:1) and MaxHash-driven deduplication (3:1).

This DRR can be compared to a software solution, which DRR is:

- From 3:1 to 4:1 reduction using LZ compression (from 1.5: to 2:1) and traditional deduplication (2:1).

This increase in DRR results in a lower cost of the storage appliance for the same effective capacity (less physical storage required), and possibly in a smaller appliance (from 2U to 1U) depending on the actual capacity with lower power. It can also result in a significant increase in effective capacity if the same number and size of drives are used. By reducing the effective data size so significantly, it is possible to greatly increase the Drive Writes Per Day (DWPD) rating and reduce the replication/back-up times, potentially enabling the use of a lower speed/cost WAN connection. In all cases, CPU cores are freed up with an offloading that can be used to add features to further differentiate the storage appliance.

Highest Performance Storage Acceleration with Lowest Latencies

Panther V offers a range of performance options up to 450Gbps on a single card. This is the highest performance solution available on a single card (or device) on the market. These transform operations can be performed with the lowest latencies of any product available. Whether performing one or multiple transform operations, Panther V can complete these tasks in the shortest time compared to any competing product. Panther V's maximum latency times from API call (command submission) to completion notification (command complete) are as follows:

- Encode operations:
  - 8KB command—15µs
  - 32KB command—35µs
  - 128KB command—75µs
- Decode operations:
  - 8KB command—10µs
  - 32KB command—20µs
  - 128KB command—50µs

Panther V also scales linearly up to 3.2Tbps with multiple cards transparently using the provided software development kit (SDK) driver.

Performance

Panther V can perform supported transform operations at up to 200Gbps encode and 450Gbps decode for the MxL8817 PCI or OCP cards. This includes single or multiple transform operations for encode or decode operations with real-time verification.

Real-Time Verification (RTV)

Panther V supports the use of RTV, which MaxLinear highly recommends. When enabled, all encode operations are automatically and transparently fully decoded. The fully decoded data is then verified against the original data to ensure that all transform operations have been completed successfully. This ensures that Panther V self-detects any errors and reports them if they occur in real time, thus avoiding any risk of silent data corruption, with no impact on performance and minimal overhead due to Panther V's heavily pipelined architecture.

RTV for hash operations is implemented as two parallel engines whose two outputs are verified byte by byte since hash operations have no decode capability.

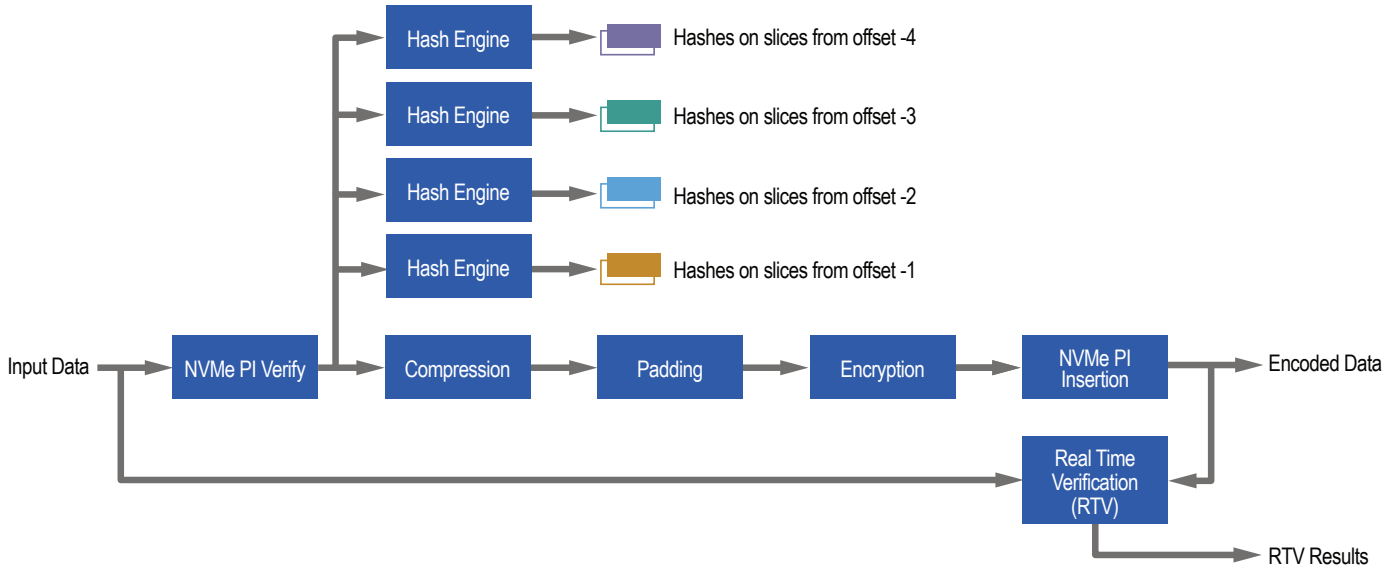


Figure 5: MaxHash Using Four Engines before Compression Operation in Panther V's Encode Pipeline

Conclusion

Panther V has been optimized to deliver maximum flexibility, performance, and feature set with minimum latency to optimize any storage appliance, from the entry level to the enterprise class. Panther V is easy to integrate and has been designed with maximum uptime and absolute data/transform reliability in mind. There is no better solution to today's challenges for maximizing a storage appliance while optimizing effective capacity.

References

- *Panther V Storage Accelerator: Next-Generation MaxHash™ Deduplication White Paper* (006-GWP).
- *MxL890x Software Development Kit (SDK) Getting Started User Guide* (210-CUG).
- *MxL890x Software Development Kit User Guide* (209UG).
- *MxL890x Raw Acceleration Application Program Interface (API) Reference Guide* (200AG).
- *MxL890x Storage Accelerator User Guide* (204-UG).
- *MxL890x Performance Evaluation Tool User Guide* (219UG).
- *MxL890x Linux Performance Application Note* (294AN).
- *MxL890x Linux Performance Tuning Application Note* (298AN).
- *MxL890x Software Development Kit Linux Release Notes* (202-CRN).
- *MxL890x FreeBSD Performance Application Note* (297AN).
- *MxL890x FreeBSD Performance Tuning Application Note* (299AN).
- *MxL890x Software Development Kit FreeBSD Release Notes* (209-CRN).

MaxLinear, Inc.
5966 La Place Court, Suite 100
Carlsbad, CA 92008
Tel.:+1 (760) 692-0711
Fax: +1 (760) 444-8598
www.maxlinear.com

The content of this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment, representation, or warranty by MaxLinear, Inc. or any of its affiliates (collectively, "MaxLinear"). MaxLinear assumes no responsibility or liability for any errors, omissions, or inaccuracies that may appear in the informational content contained in this document.

Reproduction, distribution, modification, or creation of derivative works, in part or in whole, without the express prior written consent of MaxLinear is prohibited. MaxLinear, the MaxLinear logo, and any other MaxLinear trademarks (including but not limited to MxL, Full-Spectrum Capture, FSC, AirPHY, Puma, AnyWAN, VectorBoost, MXL WARE, and Panther) are all property of MaxLinear and/or its subsidiaries in the U.S.A. and other countries. All rights reserved. All third-party marks and logos are trademarks™ or registered® trademarks of their respective holders/owners.

© 2026 MaxLinear, Inc. All rights reserved.