

FLASH-SPECIFIC DATA PROTECTION

ABSTRACT

Every enterprise storage system utilizes multiple forms of redundancy in order to protect data against loss when an inevitable array component failure occurs. RAID (Redundant Array of Independent Disks) was invented to allow data to be spread across multiple drives for performance, but also to protect the data when a disk drive (and in later iterations, multiple disk drives) fail. RAID implementations have always been created with the assumption of spinning disk as the media choice, and as a result force undesirable trade-offs between performance, capacity overhead, and the level of data protection desired.

Rather than simply adopt pre-existing RAID algorithms, XtremIO has reinvented data protection from the ground up to leverage the specific properties of flash media. The resulting data protection algorithm (XtremIO Data Protection, or XDP) in use on XtremIO arrays simultaneously combines the most desirable traits of traditional RAID algorithms, avoids their pitfalls, and brings entirely new and previously impossible capabilities to the XtremIO storage system. XtremIO Data Protection also significantly enhances the endurance of the underlying flash media compared to any previous RAID algorithm, an important consideration for an enterprise flash array.

This white paper examines the XDP implementation and discusses its benefits and advantages over RAID; with special consideration given to the unique requirements of enterprise flash storage arrays.

Copyright © 2013 EMC Corporation. All Rights Reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided "as is." EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

For the most up-to-date listing of EMC product names, see EMC Corporation Trademarks on EMC.com.

EMC², EMC, the EMC logo, and the RSA logo are registered trademarks or trademarks of EMC Corporation in the United States and other countries. VMware is a registered trademark of VMware, Inc. in the United States and/or other jurisdictions. All other trademarks used herein are the property of their respective owners. © Copyright 2013 EMC Corporation. All rights reserved.

Published in the USA. 02/13 White Paper

TABLE OF CONTENTS

- TABLE OF CONTENTS3
- INTRODUCTION4
- A BRIEF HISTORY OF RAID5
- COMPARING RAID 1, 5 AND 68
- AN INTRODUCTION TO XTREMIO'S ADVANCED DATA PROTECTION SCHEME9
- EFFICIENT STRIPE UPDATES WITH XDP10
- EFFICIENT REBUILDS14
- XDP ADMINISTRATIVE BENEFITS17
- CONCLUSION18
- HOW TO LEARN MORE.....18

INTRODUCTION

Every enterprise storage system utilizes multiple forms of redundancy in order to protect data against loss when an inevitable array component failure occurs. RAID (Redundant Array of Independent Disks) was invented to allow data to be spread across multiple drives for performance, but also to protect the data when a disk drive (and in later iterations, multiple disk drives) fail. RAID implementations have always been created with the assumption of spinning disk as the media choice, and as a result force undesirable trade-offs between performance, capacity overhead, and the level of data protection desired.

Rather than simply adopt pre-existing RAID algorithms, XtremIO has reinvented data protection from the ground up to leverage the specific properties of flash media. The resulting data protection algorithm (XtremIO Data Protection, or XDP) in use on XtremIO arrays simultaneously combines the most desirable traits of traditional RAID algorithms, avoids their pitfalls, and brings entirely new and previously impossible capabilities to the XtremIO storage system. XtremIO Data Protection also significantly enhances the endurance of the underlying flash media compared to any previous RAID algorithm, an important consideration for an enterprise flash array.

This white paper examines the XDP implementation and discusses its benefits and advantages over RAID; with special consideration given to the unique requirements of enterprise flash storage arrays.

A BRIEF HISTORY OF RAID

Disk failures remain one of the primary failure mechanisms, and a potential source of data loss, in modern storage systems. In the world of spinning hard disks, failures typically mean the simultaneous loss of the ability to read and write. RAID was invented specifically to recover from spinning disk failures by segmenting data across multiple physical drives and in most RAID algorithms calculating and storing redundant data that can be used to reassemble the information lost on one or more failed drives.

However with solid-state drives (SSDs) the failure mechanisms are often different. SSD failures may be complete, partial, or transient and an SSD data protection scheme must address all situations. The topic of why SSDs fail and how they behave exactly upon failures is beyond the scope of this paper, but XtremIO has considered all SSD failure mechanisms in the design of XDP.

Let's begin with a review of the most common existing RAID schemes (RAID 1, 5 and 6). By analyzing their space efficiency and performance overhead, we will discover why they are not best suited for protecting against SSD failures. Then we will discuss how XDP compares against these algorithms and innovates in dimensions that are not possible with spinning disks.

RAID 1 (mirroring) – this scheme replicates all writes to two different disks in parallel. Every user write is translated into two simultaneous writes by the storage array, and reads can be served from either copy. When one of the mirrored disks fails, no information is lost. The failed disk can be rebuilt by simply copying data from the remaining disk on the other side of the mirror.



Figure 1: RAID 1 – all incoming writes are mirrored to two separate disks. This is the highest performing disk RAID algorithm, but requires 50% capacity overhead. In the diagram, 4K chunks of data are mirrored.

Benefit: High performance, especially for reads, since reads can be directed to both copies of the data on separate drives.

Drawback: High cost due to capacity overhead.

Capacity Overhead: Very high. Usable capacity is only 50% of raw capacity.

Block Update I/O Overhead: Two writes, one to each side of the mirror.

RAID 5 (disk striping with parity) – a RAID 5 "stripe" is made up of blocks of data spread across a number of disks and then a single parity block (calculated by performing logical XOR operations against the data blocks) stored on an additional disk. The disk containing the parity block is rotated with each new stripe written. RAID 5 stripe sizes are often described as N+1 (where N is the number of data disks in the stripe), with 4+1 and 5+1 being the most common. With RAID 5, the simultaneous loss of two or more drives results in data loss, which limits the practical stripe width.

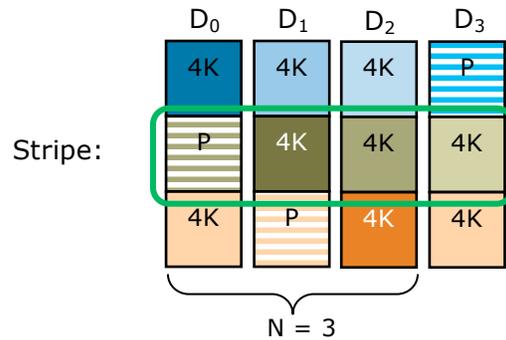


Figure 2: A 3+1 RAID 5 arrangement. Each stripe consists of three data blocks and a parity block.

Benefit: Good overall performance with reasonable capacity overhead.

Drawback: Not the best at anything. RAID 1 has better performance. RAID 6 has better data protection.

Capacity Overhead: Calculated as $1/(N+1)$. Thus a 3+1 RAID 5 has 25% capacity overhead and a 5+1 RAID 5 has 16.7% capacity overhead.

Block Update I/O Overhead: Two reads (existing data block and parity block) and two writes (new data block and parity block update).

RAID 6 (disk striping with double parity) – RAID 6 is similar to RAID 5 except that each stripe contains not one, but two parity blocks (one calculated by performing logical XOR operations across the data columns, and the other parity block calculated by encoding diagonals in a matrix of stripes). This gives RAID 6 the additional benefit of being able to survive two simultaneous drive failures. RAID 6 was originally conceived for large capacity (1TB or more) SATA drives where long rebuild times left RAID 5 storage arrays in an extended degraded state resulting in an unacceptably high probability of data loss. RAID 6 stripe sizes are often described as N+2 (where N is the number of data disks in the stripe). With RAID 6, the simultaneous loss of three or more drives results in data loss.

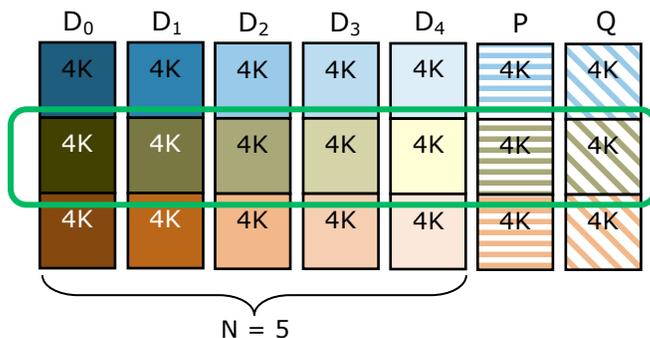


Figure 3: A 5+2 RAID 6 arrangement. Each stripe consists of five data blocks and two parity blocks.

Benefit: Strong data protection.

Drawback: High computational overhead can lead to lower performance.

Capacity Overhead: Calculated as $2/(N+2)$. Thus a 5+2 RAID 6 has 28.6% capacity overhead and an 8+2 RAID 6 has 20% capacity overhead.

Block Update I/O Overhead: Three reads (existing data block and two parity blocks) and three writes (new data block and two parity block updates).

COMPARING RAID 1, 5 AND 6

Storage administrators are often required to make a hard choice between capacity, data protection level, and performance. Performance-oriented workloads are typically provisioned on RAID 1, but at the high cost of 50% capacity overhead. Less sensitive workloads can use RAID 5, and large data sets with lower performance needs can be highly protected with RAID 6.

The challenge is that it isn't always possible to know which type of data is being stored. Data centers are dynamic and today's choice might leave data on a less-than-optimal RAID level in the future. Some storage systems do allow live migrations between RAID levels, but this requires proactive administration and may need to be repeated as data continues to evolve. This is part of what makes storage administration more an art than an exact science.

Rather than simply adopt one (or more) of the existing RAID algorithms and implement them on top of SSDs, XtremIO saw an opportunity to develop a new type of data protection scheme that combined the best attributes of pre-existing RAID levels, while avoiding their drawbacks. Furthermore, since flash endurance is a special consideration in an all-flash array, XtremIO felt it was critical to develop a data protection algorithm that requires fewer write cycles. This maximizes the service life of the array's SSDs while also delivering higher performance since more I/O cycles are available for host writes (front-end I/Os) as compared to internal array operations (back-end I/Os).

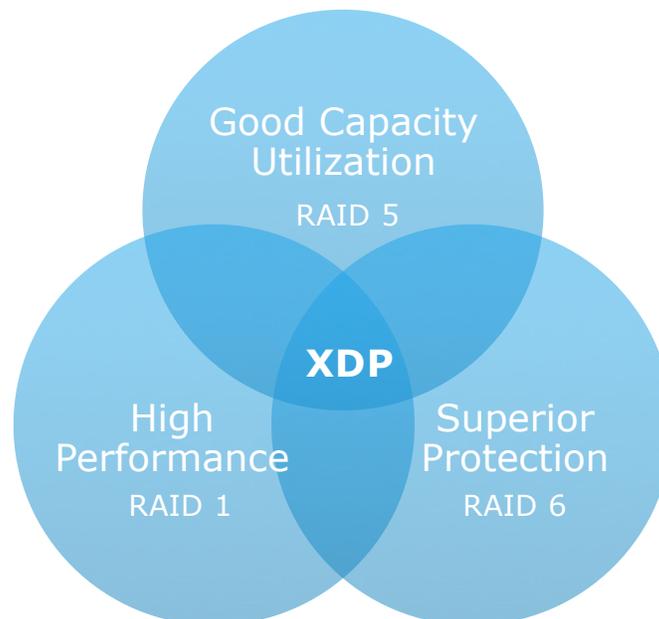


Figure 4: Existing RAID schemes make many tradeoffs. XDP combines the best attributes of various RAID levels and extends them, providing unique new capabilities previously not possible.

AN INTRODUCTION TO XTREMIO'S ADVANCED DATA PROTECTION SCHEME

XtremIO's Data Protection scheme is very different from RAID in several ways. Since XDP will always be working within an all-flash storage array, several design criteria were important:

- **Ultra-low capacity overhead** – flash capacity is more expensive than disk capacity. Thus it is desirable to use very wide striping to lower capacity overhead. XDP uses a 23+2 stripe width, equating to capacity overhead of just 8%.
- **High levels of data protection** – XDP is an N+2 scheme like RAID 6, making it tolerant of two simultaneous SSD failures in each X-Brick.
- **Rapid rebuild times** – XDP allows for very fast rebuilds, not just because flash is a fast underlying media technology, but also because XtremIO's content-aware architecture only needs to rebuild the written space on a drive. Empty space is detected and skipped. Furthermore, XDP has flash-specific parity encoding algorithms (described later in this paper) that enable rebuilds to occur with fewer I/O cycles to the drives.
- **Flash endurance** – XDP requires fewer writes per stripe update than any RAID algorithm. This extends flash endurance up to 2.5x longer than with standard RAID implementations.
- **Performance** – By requiring fewer I/O operations per stripe update, XDP leaves more drive I/O cycles available for host (front-end) I/O, resulting in superior array performance.

How does XDP achieve these seemingly contradictory goals simultaneously? The answer lies in the algorithm's ability to place and access data in any location on any SSD. Past RAID algorithms had to consider how to keep data contiguous so as to avoid disk drive head seeks. XDP presumes random access media such as flash is present in the array, and thus it can layout data and read it back in highly efficient ways that would heavily penalize a disk-based RAID algorithm, but with no ill effects in XtremIO's all-flash architecture.

XDP uses a variation of N+2 row and diagonal parity. Refer to the figure below for a sample of XDP's data layout.

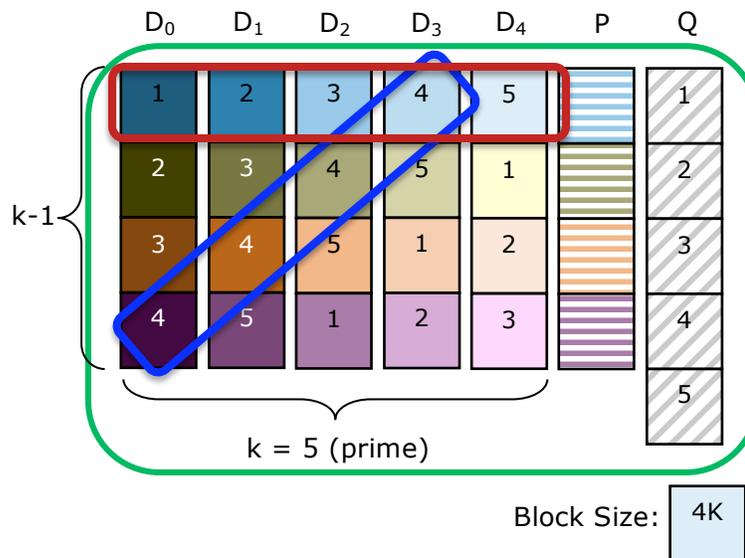


Figure 5: Sample XDP data layout. XDP makes use of both row (P) and diagonal (Q) parity calculations. Here a 5+2 stripe is shown, but in actuality XDP uses a 23+2 stripe. Each XDP protected block uses K (23) columns and K-1 (22) rows.

In the figure, XDP's row-based parity is shown by the red rectangle and the parity block is stored in the 'P' column. XDP's diagonal parity is shown by the blue rectangle and is stored in the 'Q' column. The location of the 'Q' parity block corresponds to the numbering scheme (number four in the diagram) of the diagonal.

In order to effectively calculate the diagonal parity (which spans multiple row-based stripes), XDP writes data in large chunks consisting of K * (K-1) data blocks, allowing the diagonal parity to be calculated while all data is still in memory. Since XtremIO uses a 4KB internal block size, each XDP stripe consists of 23 x 22 x 4KB = 2,024KB of user data.

EFFICIENT STRIPE UPDATES WITH XDP

But how does XDP's data layout benefit the XtremIO array as compared to traditional RAID? The answer lies in the I/O overhead required to perform a stripe update.

With all parity-based RAID schemes, it is much faster (due to lower I/O overhead) to do full stripe writes (writing data to an empty location on disk rather than updating data in an existing location) than to update a single block within a stripe. A full stripe write simply requires that parity be calculated and the data be written. However, an update to an existing stripe requires data and parity blocks in the stripe to first be read, then new parity is calculated based on the updated data and the resulting data and parity blocks are written. For example, RAID 6 incurs three reads (the existing data block and both parity blocks) and 3 writes (the new data block and the recalculated parity blocks) for each single block update. The way to lower the average I/O overhead for writing to a RAID 6 volume is to try to do as many full stripe writes, and as few single block updates, as possible. However this becomes increasingly difficult as the array fills. This is one of the reasons why array performance tends to degrade substantially as capacity runs low. The array can no longer find empty full stripes and must resort to performing multiple partial-stripe updates. This incurs much higher levels of back-end I/O, consuming disk IOPS that otherwise would have been available to service front-end host requests.

The XtremIO array and XDP work in a fundamentally different way. The XtremIO array stores blocks based not on their logical address, but on their individual content fingerprint. At the physical layer, the XtremIO array has total freedom about where to write new unique data blocks (this capability only being feasible in an all-flash design). Traditional arrays will update logical block addresses in the same physical location on disk (causing the high I/O overhead of a stripe update). All-flash arrays typically have a garbage collection function that attempts to free up stale data into full, empty stripes. However, an update to a logical block address on XtremIO is written to a new location on disk based on the content's fingerprint (if the content happens to already exist in the array, the block is simply deduplicated).

Like with traditional RAID, XDP will try to do as many full stripe writes as possible by bundling new/changed blocks and writing them to empty stripes available in the array. However, with XDP the unavailability of a full stripe does not cause the high levels of partial stripe update overhead as found in traditional RAID nor the garbage collection present in other flash arrays because XtremIO doesn't update data in place. Rather, the array will always place data in the emptiest stripe available. And as we shall explain momentarily, XDP's I/O overhead for a partial stripe update is typically far lower than with traditional RAID.

The net result is that XtremIO almost never incurs the full RAID 6 I/O overhead of a stripe update. XtremIO's average I/O overhead for such an operation is much closer to that of a full stripe write. In fact, XtremIO's average update performance is nearly 40% better than RAID 1 – the RAID level with the highest performance!

FULL STRIPES VS. STRIPE UPDATES

In any parity-based RAID algorithm (including XDP), there is less I/O overhead for writing a "full stripe" than in updating an existing stripe with new data. A full stripe write only incurs the computational overhead of calculating parity, and the drives themselves need only write the new data and associated parity blocks. An update requires reading existing data blocks and parity blocks, computing new parity, and writing the new data and updated parity. Thus, full stripe writes are desirable, and when an array is empty, easy to perform.

However, as an array fills, most writes will be overwrites of existing data blocks since empty space in the array to perform a full stripe write becomes harder to find, forcing more stripe updates to occur and lowering performance.

The common strategy for handling this situation is to implement a garbage collection function that looks for stale data in existing stripes, and re-packs it into full stripes, allowing space to be reclaimed so that there are (hopefully) always full stripes available for new writes. In a flash array this is an undesirable approach because the act of garbage collection leads to unpredictable performance (the garbage collection function itself is consuming I/O cycles and array computational overhead) and wears the flash sooner (picking up data from one location and rewriting it to a new location amplifies the number of writes the flash media endures).

XtremIO engineered a completely unique solution to this problem. Rather than garbage collect, the XtremIO array works with the assumption that stripe updates will be predominant as the array is in service over time. XDP's algorithms efficiently place new writes into the emptiest stripes in the array while providing consistent and predictable performance as the array's capacity fills. XDP avoids the write amplification problem that garbage collection creates. The result is that the XtremIO array delivers consistent performance at all times while extending flash endurance up to 2.5x longer than with traditional RAID implementations.

To demonstrate how this works, consider the situation where the XtremIO array is 80% full – which means there is 20% free space. We use such an example because it illustrates how XDP maintains performance even when the array is nearly filled, a condition that causes problems for other flash array designs because they rely on garbage collection mechanisms to move data around in order to create a completely empty RAID stripe for new incoming data. This becomes increasingly hard to do as the array fills, especially in the face of heavy write workloads.

In contrast, XDP does not require a full stripe. XDP was conversely designed to excel when dealing with partial stripe writes, which becomes the normal, steady-state operating condition when an array has been in service for some time.

When new data enters the XtremIO array, XDP will choose the emptiest stripe because it incurs the lowest overhead of parity updates and disk accesses. After the update this stripe will be full, while on average, the rest of the stripes will be slightly more than 20% free. Repeating this write/delete scenario on an array that is 80% full creates an even distribution of free space across the stripes that ranges from zero (stripe is full) to 2 x Array Free Space, which in this case is 40% (2 x 20% free).

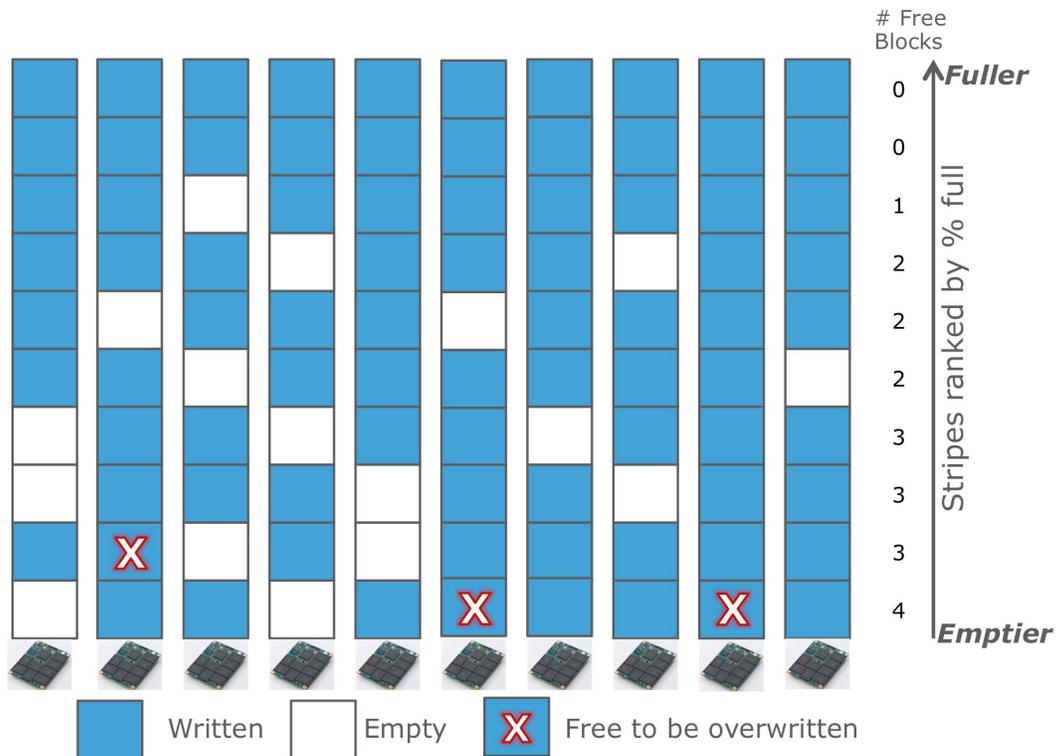


Figure 6: XDP ranks all stripes based on how full or empty they are and always writes to the emptiest stripe. In this simplified view there are ten data drives (columns) and ten stripes (rows). The emptiest stripes are ranked at the bottom. The example shows an array that is 80% full.

Since XDP uses a 23+2 stripe, and on average a stripe update is performed with 40% free space in the stripe (on an 80% full array), it means that XDP has $23 \times 40\% = 9.2$ columns in the stripe available per user write. When the stripe is updated, there will be an overhead of two writes, (row parity and diagonal parity) for a total of 11.2 write operations. 11.2 write operations to store 9.2 blocks of user data is write overhead of 1.22 I/Os. XDP's read overhead per stripe update is identical at 1.22 I/Os, reflecting the need to read existing data and parity blocks in order to perform the updated parity calculations.

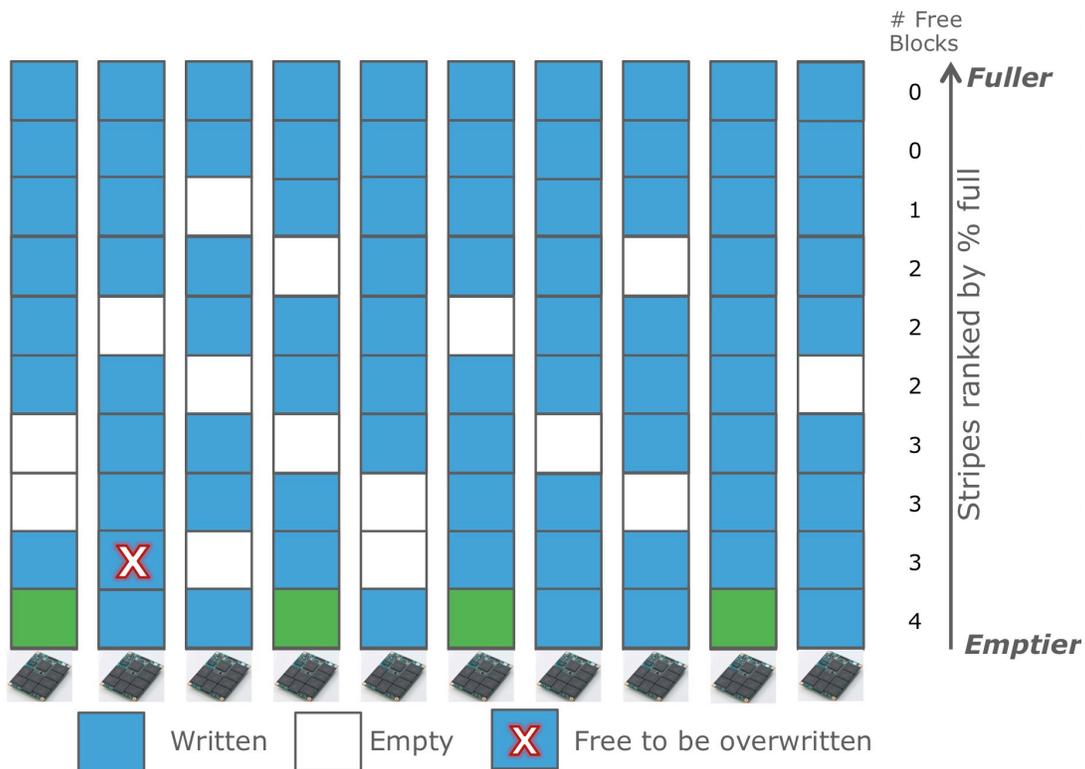


Figure 7: XDP always writes new data into the stripe with the most empty space (the one at the bottom of the stack). In this example four blocks were free (two were empty and two were available to be overwritten). Through this method, XDP amortizes the stripe update I/O overhead as efficiently as possible.



Figure 8: Every time a stripe is updated, XDP re-ranks all stripes so that the next update will occur into the emptiest stripe. As the hosts overwrite logical addresses, their old locations on SSD are marked as free to be overwritten.

The table below compares the I/O overhead of different RAID schemes against XDP.

Algorithm	Reads per Stripe Update	Traditional Algorithm Read Disadvantage	Writes per Stripe Update	Traditional Algorithm Write Disadvantage
XDP	1.22	-----	1.22	-----
RAID 1	0	-----	2	1.6x
RAID 5	2	1.6x	2	1.6x
RAID 6	3	2.4x	3	2.4x

In a flash-based system, the disadvantage of traditional RAID manifests itself not only in lower performance, but also in reduced endurance of the flash media. Since flash drives have a finite number of write cycles they can endure, XDP allows XtremIO arrays to not only perform better, but to last longer. This is a significant advantage of high value in an enterprise storage array. Consider XtremIO against another product that implements RAID 6. Both provide N+2 data protection, but XtremIO will allow the flash to last 2.4x longer while delivering 2.4x better I/O performance!

XDP's advantage holds true not only for updates, but for other important storage operations such as rebuilds upon SSD failures.

EFFICIENT REBUILDS

XDP's innovative diagonal-based parity scheme plays an important role during drive rebuilds. While rebuilding a failed drive after a single drive failure can be done naively using row-based parity information only (which requires reading K blocks per failed block), this is not the method XDP uses.

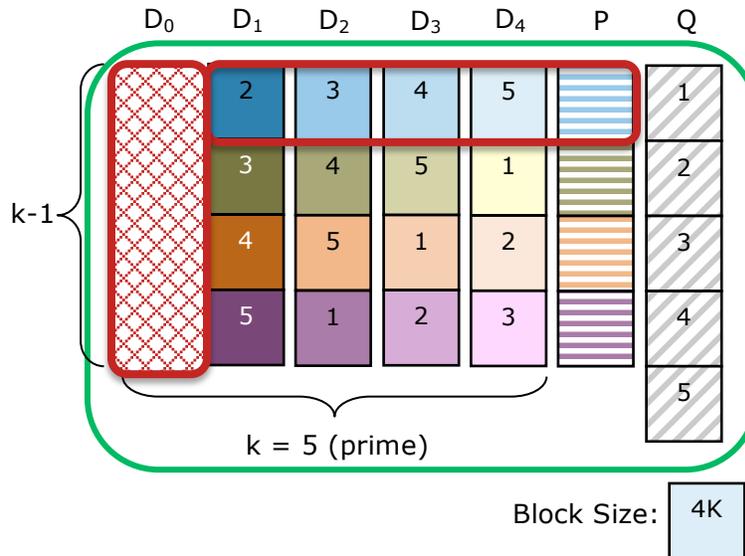


Figure 9: With RAID, a single drive failure (D₀) can be rebuilt using row-based parity (P). This requires reading K blocks (D₁, D₂, D₃, D₄, P) to rebuild each missing data block on the failed drive. XDP is far more efficient than this method.

XDP accelerates rebuilds by making use of both row and diagonal parity. Assuming D₀ failed, XDP will recover the first two blocks (D₀₋₁, D₀₋₂) using their row parity. With traditional RAID, the row-based parity rebuild process would continue with remaining rows.

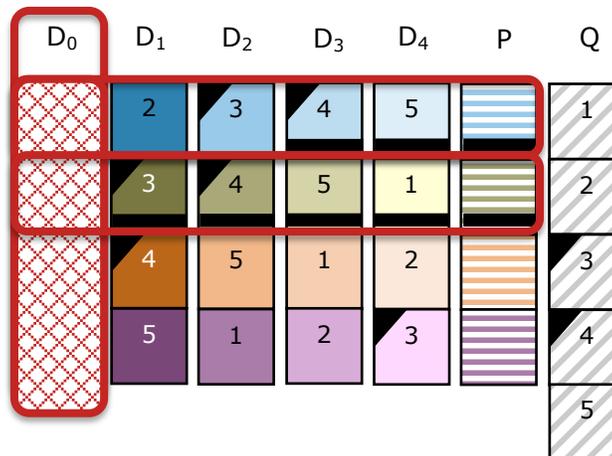


Figure 10: XDP's hyper-efficient rebuild process. In this example, the first two rows are rebuilt using conventional row-based parity.

However, with XDP, as the system reads these two rows, it keeps some of their data blocks in memory because they can be leveraged to perform diagonal-based parity rebuilds. In the diagram above, the blocks D2-3, D3-4, D1-3, and D2-4 are retained in this fashion. With this information already available, XDP need not read all the remaining rows in order to complete the rebuild because many of the diagonal parity stripes can be completed with just a few additional blocks.

For example, the diagonal stripe for the blocks numbered 4 (D0-4, which must be rebuilt, D1-4, D2-4, D3-4, and Q-4) already has blocks D2-4 and D3-4 in memory from the prior row-based rebuild operations.

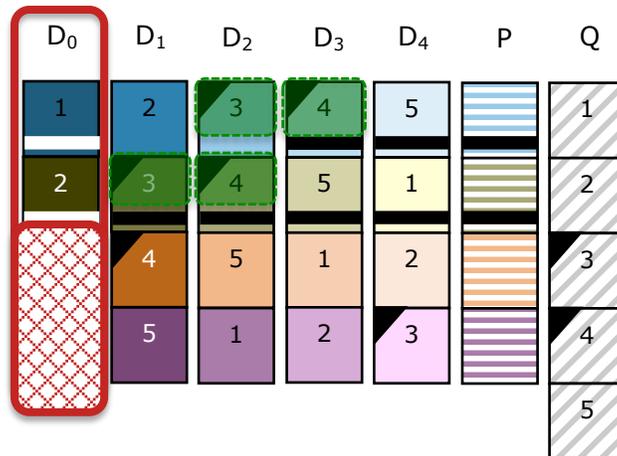


Figure 11: XDP has used row-based parity to rebuild D0-1 and D0-2. It has also retained the blocks marked in green in memory so they do not need to be re-read to complete the rebuild. This data will be used to complete the rebuild using diagonal parity (Q).

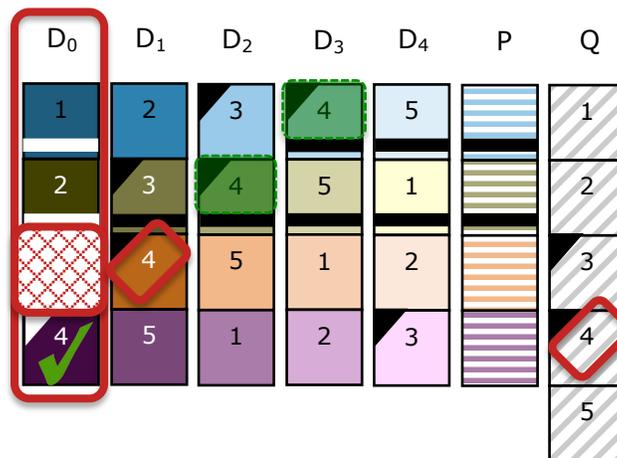


Figure 12: XDP's diagonal-based parity enables rebuilds to occur with fewer I/O operations. Here, D0-4 is rebuilt using a combination of information retained during a row-based rebuild of D0-1 and D0-2, along with diagonal information contained in D1-4 and Q-4.

Thus, rebuilding D0-4 only requires two reads (D1-4 and Q-4) whereas in traditional RAID it would require four reads (D1-5, D2-1, D3-2, D4-3, and P). Similarly, rebuilding D0-3 only requires reading D4-3 and Q-3.

This method does not provide any advantage when rebuilding lost parity columns, and thus it requires a little more than $3K/4$ reads on average (recall that each disk contains both data and parity columns in a rotating parity scheme). It is easy to see that such a scheme also balances the rebuild reads evenly across the surviving disks.

Algorithm	Reads to rebuild a failed disk stripe of width K	Traditional Algorithm Disadvantage
XDP	$3K/4$	
RAID 1	1	None
RAID 5	K	33%
RAID 6	K	33%

XDP's efficiency advantage leads both to faster rebuild times as well as better array performance during rebuilds since fewer back-end I/Os are required to complete the rebuild, leaving more front-end I/Os available for user data.

XDP ADMINISTRATIVE BENEFITS

XDP's benefits extend beyond superior data protection, performance, capacity utilization, and flash endurance. The algorithm is completely implemented in software (rather than in a hardware controller such as an ASIC or FPGA), allowing tremendous flexibility as future X-Brick designs are rolled out. This flexibility is demonstrated in how XDP responds dynamically to common fault conditions.

A standard XtremIO X-Brick contains 25 SSDs, with 23 for data and two for parity. When one of the 25 SSDs in an X-brick fails, XDP will quickly rebuild the failed drive, while dynamically reconfiguring incoming new writes into a 22+2 stripe size to maintain N+2 double failure protection for all new data written to the array. Once the rebuild completes and the failed drive is replaced, incoming writes will once again be written with the standard 23+2 stripe. This adaptability allows XDP to tolerate a sequence of SSD failures without having to rush to the data center to replace failed SSDs.

XDP always maintains a reserve capacity to perform a drive rebuild (this reserve is factored out of XtremIO's usable capacity) and can continue to rebuild failed drives again and again while serving user I/Os, as long as there is still available capacity in the array. The array will have less available capacity (until the faulty drives are replaced) and may be a little slower because of fewer working drives, but will remain healthy and protected. This is a very unique feature that allows administrators to defer drive replacements until a convenient time, which is especially important in remote, secure, and 'lights out' data centers. A side benefit of this is that the array doesn't require hot spares, so every slot in the system contains an SSD that is actively storing data and adding to the array's performance.

A final administrative benefit of note is that XDP is tolerant of the most common mistake made with respect to storage arrays – the accidental removal of a drive. This happens when an administrator goes to replace a failed drive and accidentally ejects the wrong slot. In traditional RAID this results in either data loss (RAID 5) or a 2nd level rebuild being initiated (RAID 6). With XtremIO, the drive should just be re-inserted. The array's content-based engine will recognize the drive and its contents and XDP will simply abort the 2nd level rebuild that was initiated and bring the drive back into service.

CONCLUSION

XtremIO's Data Protection scheme represents a leap forward in storage array technology. By designing for and taking advantage of the unique properties of flash storage, XDP enables XtremIO arrays to perform better and last longer while being easier to use and costing less. A summary of XDP's benefits are:

- N+2 data protection like RAID 6.
- Incredibly low capacity overhead of 8%
- Performance superior to any RAID algorithm
- Flash endurance superior to any RAID algorithm
- Faster rebuild times than traditional parity-based RAID algorithms
- Superior robustness with adaptive algorithms that fully protect incoming data even when failed drives exist in the system
- Administrative ease through fail-in-place support and accidental drive removal handling

With these advantages, the XtremIO all-flash array gives storage administrators the best of all worlds. There is no longer any need to guess which RAID scheme to use for a particular data set, and no trade-offs to make. With XtremIO data gets the best protection and the best performance at the same time.

HOW TO LEARN MORE

For a detailed presentation explaining XtremIO's storage array capabilities and how it substantially improves performance, operational efficiency, ease-of-use, and total cost of ownership, please contact XtremIO at info@xtremio.com. We will schedule a private briefing in person or via web meeting. XtremIO has benefits in many environments, but is particularly effective for virtual server, virtual desktop, and database applications.

CONTACT US

To learn more about how EMC products, services, and solutions can help solve your business and IT challenges, [contact](#) your local representative or authorized reseller—or visit us at www.EMC.com.

EMC², EMC, the EMC logo, XtremIO and the XtremIO logo are registered trademarks or trademarks of EMC Corporation in the United States and other countries. VMware is a registered trademark of VMware, Inc., in the United States and other jurisdictions. © Copyright 2013 EMC Corporation. All rights reserved. Published in the USA. 02/13 EMC White Paper

EMC believes the information in this document is accurate as of its publication date. The information is subject to change without notice.