PRODUCT WHITEPAPER

NSULATE™

Go Beyond RAID

Nyriad NSULATE™ is a software-defined alternative to RAID, designed to enable modern GPUs to function as powerful storage controllers for hyperscale storage solutions. This whitepaper is designed to help IT professionals, system integrators and researchers understand and evaluate the advantages of GPU-based storage that go beyond traditional RAID-based configurations. We also provide an overview of the major capabilities of NSULATE and compare them as closely as possible to existing RAID software. Benchmarks were performed on an Orion storage server provided by Advanced HPC Systems. The Orion uses an NVIDIA P4 GPU configured with Nyriad NSULATE for storage-processing instead of a hardware RAID controller.

For the purposes of these benchmarks, we used the ZFS file system. ZFS was chosen because it is used widely for Lustre solutions and has many software features in common with NSULATE that make direct comparisons easier. Although the XFS and EXT4 file system does not provide a comparable software RAID solution to ZFS, we also benchmarked NSULATE with XFS and EXT4. This is because we found XFS and EXT4 to generally be much faster than ZFS for storage transactions, and NSULATE adds essential resilience features to these file systems, making them interesting alternatives to ZFS for many HPC and big data applications.

Alexander St. John
Chief Technology Officer

NYRIAD™
we make every bit count

NYRIAD™
we make every bit count

Table of Contents

NYRIAD™
we make every bit count

Introduction

NSULATE is a GPU-accelerated alternative to RAID that evolved from work on the Square Kilometer Array (SKA) Telescope by the International Centre for Radio Astronomy (ICRAR). The Square Kilometer Array Telescope is the world's largest IT project supported by nearly a dozen nations and ultimately involving the deployment of hundreds of thousands of radio antennae across the Australian and African continents to map the visible Universe. The SKA needs to process over 160TB/s of astronomical data in real-time and store over 50PB of data per day at its regional supercomputing centre in Perth, Australia.The scale of these computing and storage demands generally exceed the capabilities of all modern IT solutions and have demanded extensive research into new architectures capable of reliably processing and storing data at these scales cost effectively.

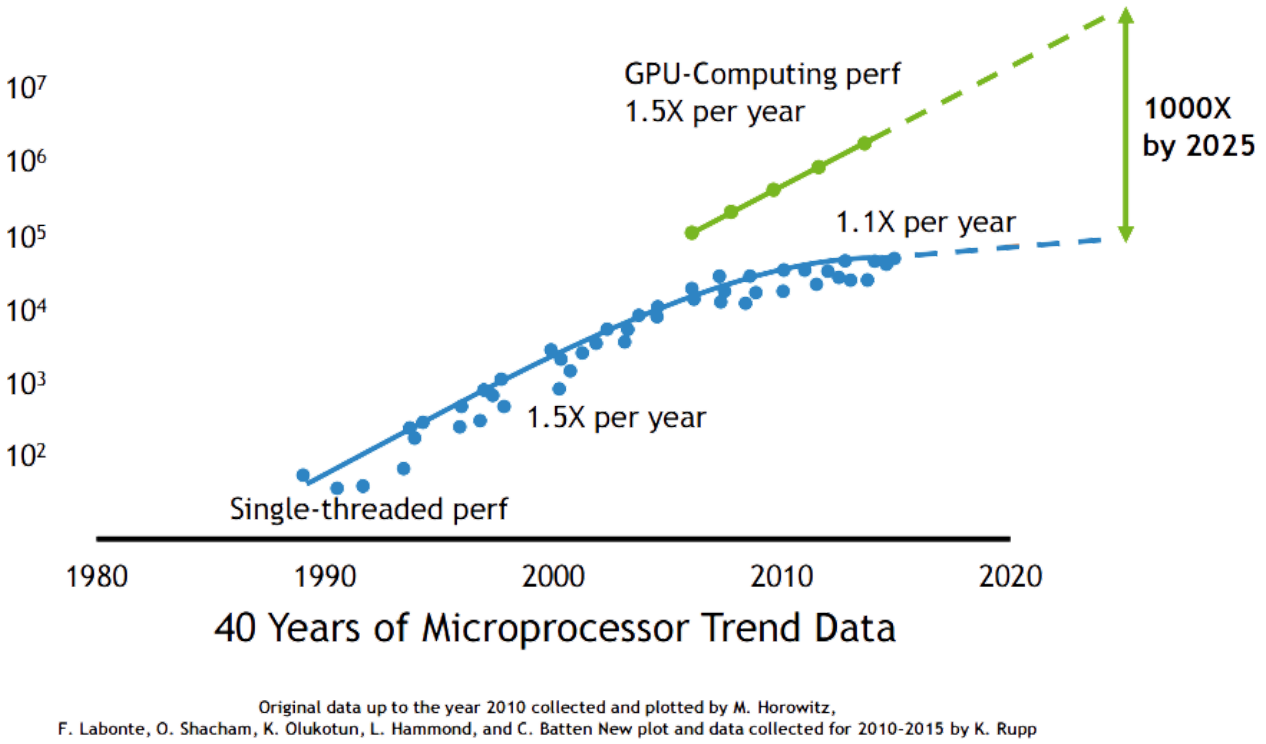
NSULATE was developed as a solution to the speed, resilience and operating cost requirements of the SKA precursor project managed by ICRAR. SKA researchers estimated that storage and networking fabric would consume as much as 70% of the power budget for the SKA supercomputers if a revolutionary leap in storage efficiency wasn’t available in time for SKA construction. Nyriad proposed the radical idea of using powerful Graphics Processing Units (GPUs), which perform the massive computations for the SKA, to also manage the storage-processing, thus replacing the separate storage system and supporting network fabric with hyperconverged compute-storage nodes. ICRAR funded a research project to develop the concept in 2014. After several successful trials, Nyriad has commercialized the solution as NSULATE for general market adoption.

NSULATE Overview

NSULATE is a Linux block device that functions as a software-defined alternative to RAID for configuring fast, reliable, larger scale storage solutions. Unlike software and hardware RAID solutions, NSULATE uses modern GPUs to perform storage-processing operations.NSULATE can be configured as an alternative to RAID in the same environments that RAID solutions are typically used, in addition to enabling many new processing solutions.

Why would you use a video card to run a filesystem?

GPUs are highly parallel processors, originally invented to perform the real-time 3D calculations essential to modern video games. Over the years GPUs have evolved to become so powerful that they are increasingly used as the core processing elements in modern supercomputers and for machine learning applications. While never intended to be used as storage-processors, the GPU's intrinsic parallelism enables it to scale performance and power savings faster than any other chip technology.



Source: NVIDIA

The rapid advances in GPU computing performance, combined with increasing adoption of GPUs for enterprise computing applications, has presented new challenges for storage technology. It is increasingly difficult to move data to and from storage fast enough to keep up with the processing speed of GPUs in media, big-data, HPC and machine learning applications.

Although storage is generally not considered to be a compute intensive application, it turns out that in the presence of the enormous low-cost processing power provided by a GPU, there are many opportunities to convert abundant compute cycles into valuable IO speed, resilience and storage efficiency.

Properties of NSULATE

NSULATE had to meet several requirements to be a practical solution for hyperscale storage.

- Highly resilient to failure
- Extremely high performance
- Extremely scalable
- Provably able to detect and recover from nearly all data corruption
- Lower power
- Efficient storage capacity utilization
- Offloads CPU workload to GPU
- Deployable using inexpensive and widely available server components
- Compatible with all existing applications and file systems

For the purposes of this whitepaper, we focus primarily on comparing NSULATE's performance and resilience to software RAID-based storage solutions. We address other features in the context of these benchmarks.

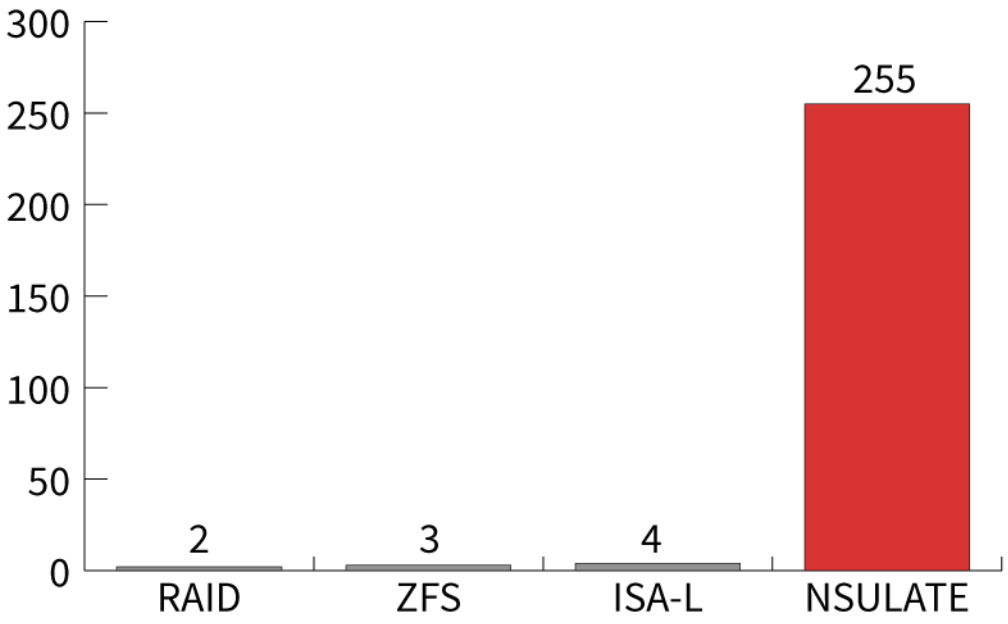
NYRIAD™
we make every bit count

Features of NSULATE

RAID controllers provide several essential core functions for storage, including managing the connectivity and SAS protocol for many SAS-based storage devices, caching, write-hole-protection and failure resilience through error correction calculations. Higher level filesystems layer many additional features on top of these capabilities, but a critical common feature is checksumming to detect data corruption.

Properties of RAID	**RAID 6 Hardware**	**ZFS-Software RAID**	**NSULATE**
SAS Connectivity	Yes	SAS-HBA	SAS-HBA
Resilience	Can recover from up to **2** device failures	Can recover from up to **3** device failures	Can recover from up to **255** device failures
Write-hole Protection	Using non-volatile memory	Via software or designated non-volatile block device	Via software or designated non-volatile block device
Corruption Detection	Relies on filesystem	Cryptographic checksum	Cryptographic checksum
Reduced CPU utilisation	Yes	CPU intensive	Yes

NSULATE excels at real-time, high parity erasure coding, adding these capabilities automatically to file systems that do not have these features.



Modern storage solutions generally rely on some form of error correction to enable them to survive the failure of some individual drives without risking data loss or overall array failure. The RAID 6 standard supports arrays with up to 2 possible device failures. The ZFS file system supports a mode called ZRAID3, which is resilient to 3 device failures. Intel provides an open source library called ISA-L, which is used by many large hyperscale data centers, including Amazon, at 4 parity. Although ISA-L can support up to 255 device failures in theory, in practice, we only find it used commercially at 4 parity. In addition, it cannot recover data from failed devices in real-time, which limits its practical utility for higher parity applications. By contrast,

NSULATE supports arrays with up to 255 device failures, and it can recover data from dozens of failed devices in real-time, allowing it to run highly degraded arrays without the immediate need to replace or repair lost drives.

For the purposes of this whitepaper, we have focused on NSULATE's unique features as they relate to performance and resilience. NSULATE has the usual features associated with RAID controllers but adapted for NSULATEs unique capabilities.

GPU Accelerated Erasure Coding

NSULATE's primary advantage is a major leap in real-time GPU-accelerated error correction technology, enabling larger, more parallel and therefore faster arrays.

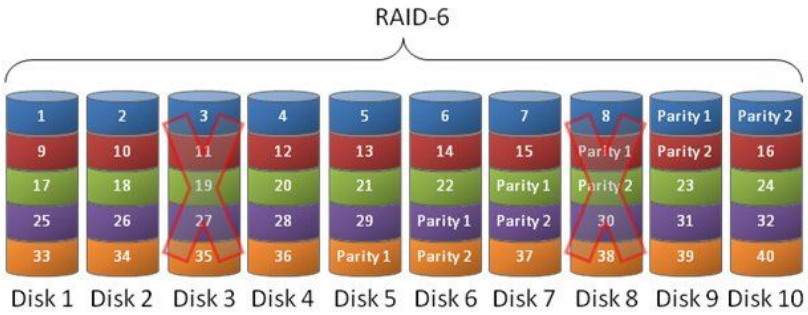
Erasure Coding Basics

Much has been written on the subject of erasure coding and storage. The following link is a useful introduction to the subject.

https://www.usenix.org/system/files/login/articles/10_plank-online.pdf

NSULATE computes Reed-Solomon erasure coding identical to Intel's ISA-L library in order to be compatible with a proven and widely adopted erasure coding equation. For the purposes of this whitepaper, we have adopted a common notation for describing erasure coding resilience in the form K:M in which K represents the number of data units in an erasure encoded stripe and M represents the number of parity units. A complete erasure encoded stripe is K+M units in size and can recover from the loss of any arbitrary M units of data. NSULATE can configure arrays with any combination of K+M <= 256. Arrays larger than 256 physical parallel drives are supported.

A typical RAID 6 configuration may be described as 8:2, meaning that the array contains 10 drives, 2(M) of which are parity overhead used to reconstruct lost drives mathematically. 10-2 = 8(K) are data units. Any 2 of the 10 drives can fail without data loss but the loss of a third drive will cause the array to lose data.



**Note that in practice the blocks of data composing parity data are actually distributed across all drives in the array in order to ensure that the failure of any randomly chosen drive has a consistent impact on overall array performance.*

The storage overhead for erasure coding is calculated as M/(K+M) or 2/10 = 20% in this example.

High Parity Erasure Coding

High parity erasure coding enables extreme resilience to data loss. While RAID arrays are constrained to surviving no more than 2 concurrent drive losses; an NSULATE array can be configured to survive the random loss of hundreds of concurrent drives. This means that exascale arrays configured this way can survive the failure of entire storage nodes, racks or remote sites.

Trade-offs

Prior to NSULATE, previous attempts to productize high parity storage solutions created some tradeoffs which impeded the practical commercialization of the technology. The two major trade-offs are:

1. Additional write amplification for high parity random writes
2. Additional read amplification for degraded random reads

Generally, these two trade-offs can impact random read and write performance under some conditions, which is mitigated by the 64GB of non-volatile cache memory allocated to NSULATE. For the purposes of this whitepaper we have chosen to focus on general file system performance comparisons and reserve more detailed analysis of random read and write performance for a subsequent paper.

Degraded Array Performance

NSULATE is designed to rely on extreme storage parallelism to run smoothly with arrays always in a degraded state (containing failed drives), requiring no admin action to mitigate the risk of array failure. This contrasts with RAID which was designed to keep an array from failing outright while an IT admin took action to replace a failed drive before more drive failures risked the loss of the array.

NSULATE can rebuild missing data from parity faster than the data can be fetched from any storage device.

This means that in practice NSULATE arrays can run in a highly degraded state with little impact on performance. It's either never necessary to rebuild a failed drive; or rebuilds can be performed at very low priority.

NSULATE enables arrays to be configured with much higher levels of parallelism because of the data security enabled by higher parity and real-time data rebuilding. The result is much greater potential performance as a result of increased parallelism at the expense of increased data amplification under some degraded conditions.

Storage Efficiency

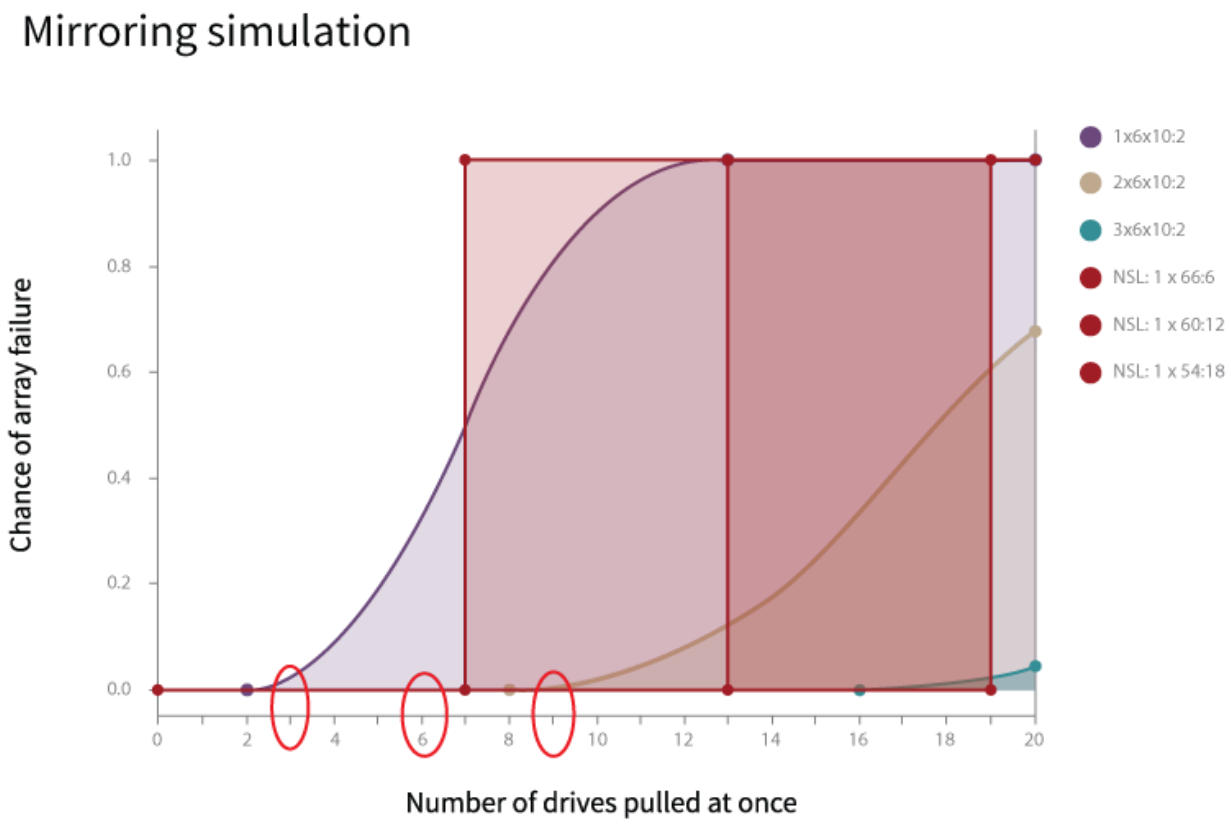
Storage efficiency increases with array scale as the ratio of K to M needed to achieve greater resilience declines. A RAID 6 array composed of 10 data drives and 2 parity drives has 20% parity overhead to mitigate the risk of any 3 random drives failing before a replacement drive can be rebuilt. An NSULATE 200:40 array also has 20% parity overhead to guard against the extremely remote probability that any randomly chosen 41 drives will fail before just one can be rebuilt. As the NSULATE resilience calculator illustrates, this is an excessive level of resilience. A 200:16 configuration is still extremely resilient to failure with only 7.4% parity overhead for a savings of 24 storage devices at scale.

Mirroring

1:M configurations of NSULATE result in automatic mirroring of blocks across devices. A 1:255 NSULATE array would store 255 parallel copies of each block stored in the same array; hence, an NSULATE array can be self-mirroring. Any random failure of up to 255 drives would be recoverable. Although this level of mirroring is extreme, it is common for larger arrays to be mirrored to additional arrays to increase their resilience.

How do we evaluate the value of resilience from mirroring against higher parity resilience? Assuming that any chance of data loss is unacceptable, a RAID 60 10x8:2 array has some chance of data loss with 3 failed drives. A mirrored RAID 60 10x8:2 array has some chance of data loss with 6 random drive failures.

By contrast a 94:6 parity NSULATE array has zero chance of data loss with six failed drives, saving 112 drives in redundant mirrored storage capacity.



Illustrated here are the failure probabilities for 1X, 2X and 3X mirrored RAID arrays (circled in red) versus 6, 12, and 18 parity NSULATE arrays. The 12 parity NSULATE array has zero probability of failure until 13 drives are lost versus a 2X mirror of an array which begins to risk data loss with the failure of 8 drives. The Orion server holds 100 drives. Configured with 90:10 parity, it is more resilient to data loss than a mirrored RAID array, conserving dozens of drives in potentially wasted storage capacity.

Summary

High parity erasure coding as implemented by NSULATE has three benefits in one solution;

Higher data resilience, higher performance due to increased parallelism and greater storage efficiencies at scale, which equates to lower cost and reduced power consumption.

Cryptographic Checksums

Unlike RAID controllers, NSULATE performs cryptographic checksums in real-time on every stored and recovered block of data. NSULATE supports a range of GPU-accelerated checksum algorithms, but all of them are equally fast for NSULATE to compute, which makes using the most mathematically sound checksums practical for all storage applications. The most advanced checksum NSULATE supports is SHA3 512 bit. The use of SHA3 checksums in concert with high parity erasure coding enables NSULATE to prove that every bit stored is perfectly recovered and that any corrupt or unrecoverable data can always be identified. Although this level of corruption detection may be excessive for most enterprise applications, it is relatively free for NSULATE to compute and necessary for mission critical and hyperscale applications where data corruption is inevitable and can be extremely costly to recompute.

RAID controllers provide no checksumming.

RAID controllers rely on the overlying filesystems to provide the feature of checksumming. If a RAID controller delivers corrupt data to a file system that performs checksumming, it is usually too late for the file system to correct the error. The ZFS file system is the exception to this rule and, like NSULATE, can detect and recover from corrupt data. This is one of the reasons we chose ZFS as the software-defined storage solution to compare NSULATE with.

Common Linux file system features	**Checksum**	**Hardware accelerated cryptographic checksum**	**Self Healing**	**Software erasure coding**	**Hardware accelerated erasure coding**	**Write hole protection**
EXT4	CRC32 on metadata only	No	No	No	RAID, max 2 parity	No
XFS	CRC32 on metadata only	No	No	No	RAID, max 2 parity	Yes
ZFS	CRC32, SHA2, Fletcher4	No	Yes	ZRAID3, max 3 parity	RAID, max 2 parity	Yes
+NSULATE adds the following features to all supported Linux file systems	CRC32C, SHA2, SHA3, SHA2 Merkle, SHA3 Merkle, on all data and metadata	**GPU accelerated** CRC32C, SHA2, SHA3, SHA2 Merkle, SHA3 Merkle, on all data and metadata	Yes	Reed-Solomon, K+M <= 256 ISA-L or Jerasure compatible	**GPU accelerated** Reed-Solomon, K+M <= 256 ISA-L or Jerasure compatible	Yes

Table of popular Linux file system resilience features and what NSULATE adds to them

Scale

Higher parity, higher quality checksums and higher performance enable larger scales and associated increases in storage efficiency. At two orders of magnitude higher parity, NSULATE enables two orders of magnitude larger contiguous array scales.

Hyperscale Resilience

To demonstrate the resilience differences between low parity RAID solutions and high parity NSULATE arrays, we developed a resilience calculator to model the probability of data loss in the event of X drives randomly failing in an array at the same time. The purpose of this analysis is to help IT professionals develop an intuition for the distinct risk differences between RAID and NSULATE arrays configured for the same number of drives, enabling us to make side-by-side comparisons between traditional RAID configurations and high parity NSULATE-based approaches. More elaborate MTTF calculations that include rebuild times are appropriate for real-world configuration considerations, but for the purposes of understanding the underlying resilience properties of NSULATE, a simpler model is used to illustrate the essential differences in resilience between low parity and high parity erasure coding.

NSULATE Resilience Calculator

The NSULATE Resilience Calculator is based on the application of a hypergeometric distribution to simulate the probability of an array failure given the instantaneous random failure of some number of drives in the array.

> "A random variable X follows the hypergeometric distribution if its probability mass function (pmf) is given by[1]
>
> $$P(X = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$
>
> where
>
> - N is the population size,
> - K is the number of success states in the population,
> - n is the number of draws,
> - k is the number of observed successes,
> - $\binom{a}{b}$ is a binomial coefficient.
>
> The pmf is positive when $max(0, n + K - N) \leq k \leq min(K, n)$ „

For N drives in a RAID array, n drives in one RAID, K failures in the RAID array and parity k

$$f(N,K,n;k) = \begin{cases} 0, & \text{if } \sum_{i=k+1}^{i=n} P(X=i) = 0 \\ 1, & \text{if } \sum_{i=k+1}^{i=n} P(X=i) = 1 \\ \sum_{i=k+1}^{i=n} P(X=i) + \sum_{j=0}^{j=k} f(N-n, K-j, n; k), & \text{otherwise} \end{cases}$$

These equations give us a mathematical answer to the question: "Given a server composed of N virtual arrays, with M parity per array, what is the probability that the array will fail if X randomly chosen drives fail at the same instant due to exceeding the parity tolerance of one of the virtual arrays?"

As an additional intuitive check that the math is correct, a Monte Carlo simulation of the same kind of array failure is included with the resilience calculator. As the two simulations show, there is a very high correlation between graphs of the same input parameter results.

Evaluated Configurations

Our reference system is a 100 drive storage server, as described further below. We chose to contrast several variations of software RAID configurations for this server compared to NSULATE on the same system.

- **ZFS (+ZRAID60) 10x8:2 with SHA256 checksums**
- **ZFS (+ZRAID30) 5x17:3 with SHA256 checksums**
- **ZFS-NSL 1x100-m where m = 8, 15, 20 with SHA256Merkle checksums**
- **XFS-NSL 1x100-m where m = 8, 15, 20 with SHA256Merkle checksums**

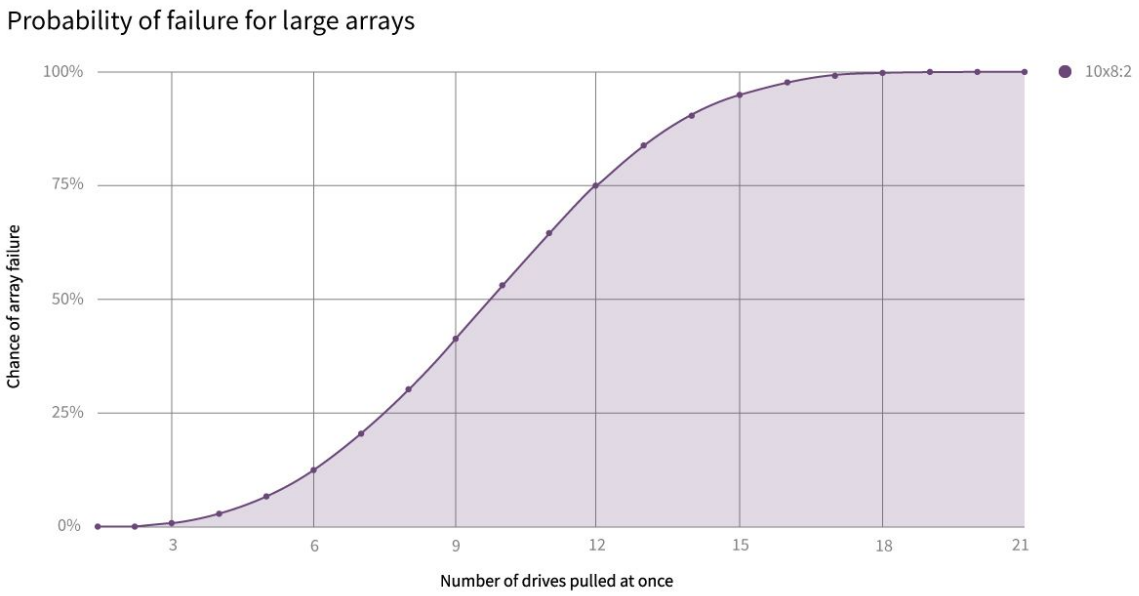
In these trials, the NSULATE resilience calculator modelled the probability of data loss given the instantaneous failure of 0...N randomly chosen drives in each array configuration. The NSULATE trial is compared with the ZFS-RAID trial in every graph. The Y-axis shows the probability of data loss while the X-axis shows the number of randomly selected failed drives.

ZFS (+ZRAID60) 10X8:2 with SHA256 checksums

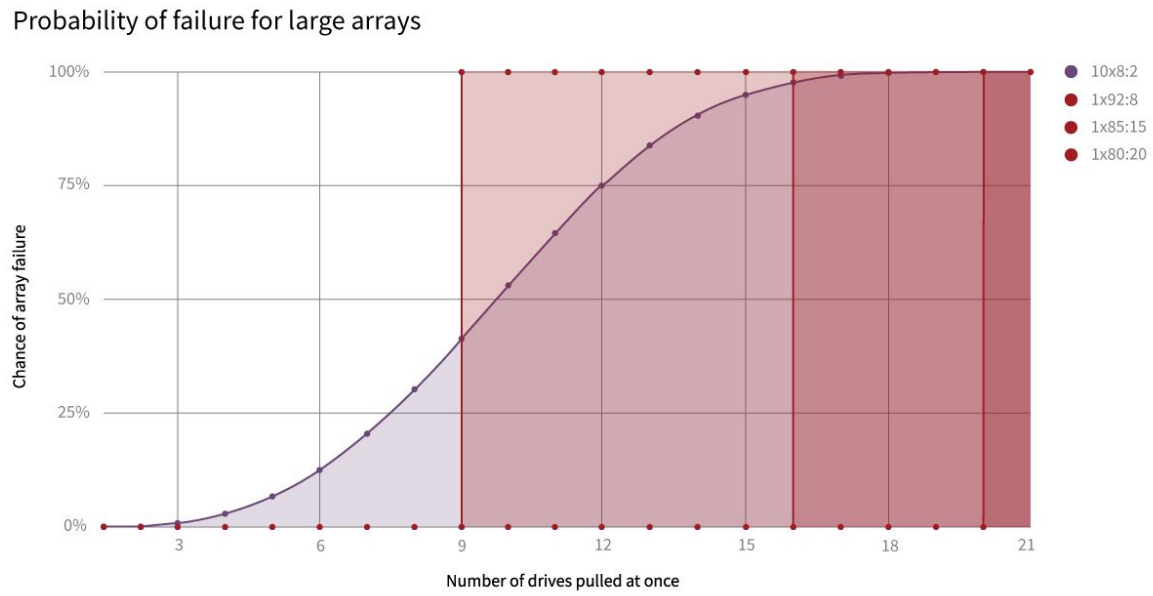
A RAID 60 configuration of 10 virtual arrays (spans) with 2 parity each, striped together would be a common way to approach configuring a 100 drive storage node. Intuitively, we can observe that the worst case scenario for this array is 3 drives coincidentally failing in the same span at the same time. There is a 100% chance of data loss in this scenario. The exact probability of 3 failures in the same span happening is relatively low, as the resilience calculator illustrates, but grows rapidly as more drives randomly fail, until array failure becomes a 100% certainty at 21 drive failures. The same array, configured with NSULATE, and 80:20 erasure coding has the same storage capacity as the RAID 60 array but zero chance of array failure with up to 20 drives randomly failing.

From this analysis we observe that the array with NSULATE would probably be more resilient in the real-world, configured as one span with just 3 parity drives, saving 17 drives worth of relatively wasted storage capacity.

The addition of just one additional parity drive has a large statistical impact on an array's overall resilience to drive failure.

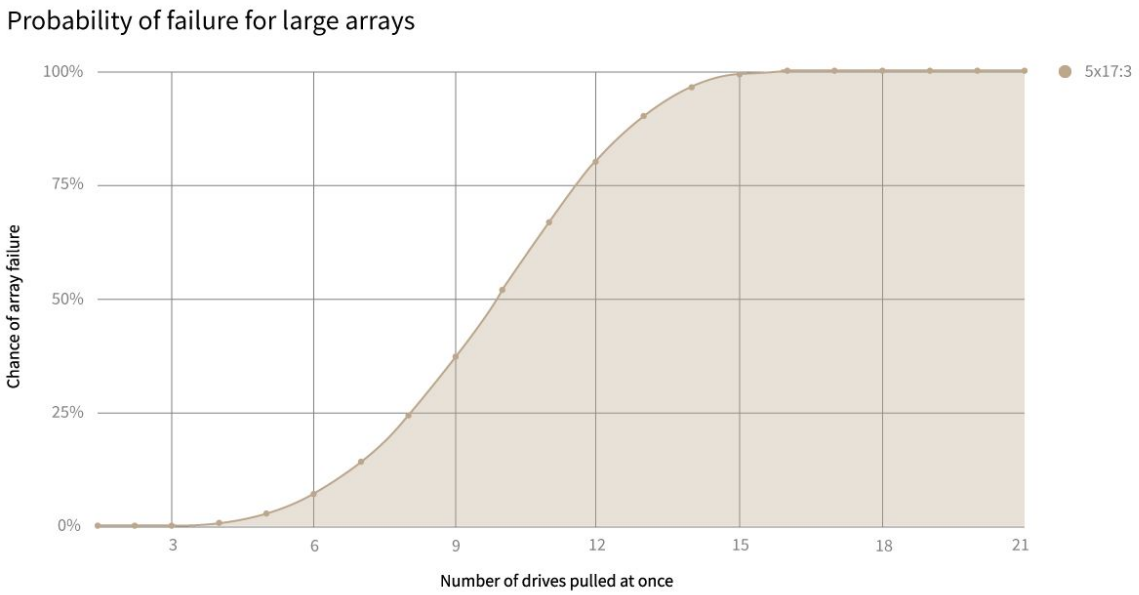


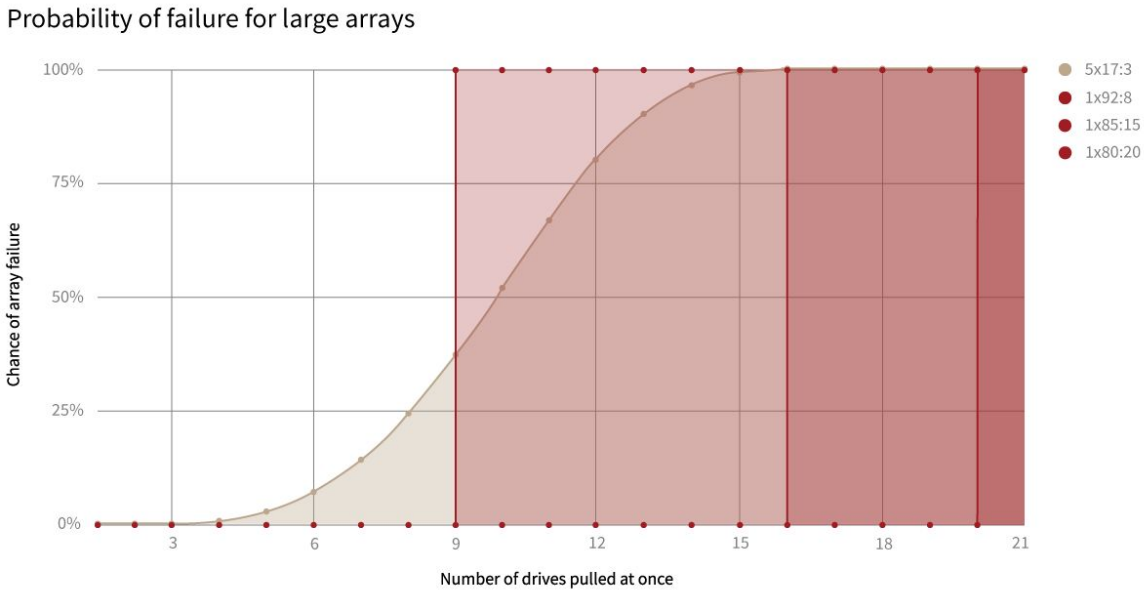
A 100 drive NSULATE array is more resilient to data loss with 3 parity drives than a 10X8:2 RAID 60 array with 20 parity drives.



ZFS (+RAID30) configured as 5X17:3

The ZFS file system has several valuable features when it comes to achieving high data resilience, including support for 1 additional parity per array and cryptographic checksumming. As with the RAID 60 array, we immediately observe that the failure of 4 drives in the same span will cause certain array failure. Because the probability of 4 drive failures is low, the resilience contribution of the additional 11 parity drives is negligible.





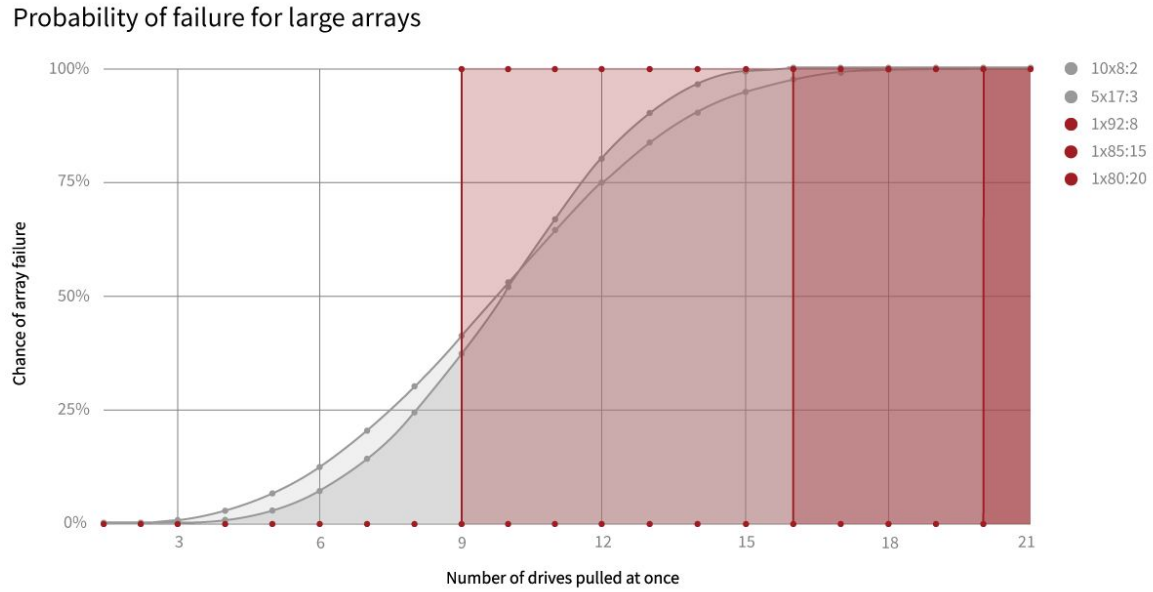
A 100 drive NSULATE array is more resilient to data loss with 4 parity drives than a 5X17:3 ZFS array with 15 parity drives.

Contrast this to a 92:8 NSULATE array, in which all 8 parity drives are contributing equally to the array's overall 100% resilience for up to 8 drives failures.

Conclusions

With cryptographic checksumming there is zero probability of data loss in any NSULATE array up to the array's parity limit. This can be as high as 255. In this context, at some scale,

a single contiguous NSULATE array with high parity is always more reliable and storage efficient than any RAID array that has to be composed of many spans to compensate for lack of parity coverage.



The NSULATE arrays have zero probability of failure until parity+1 drives fail, at which point they have an abrupt 100% chance of failure.

Performance Evaluation and Analysis

Nyriad performed a series of comparative tests in the lab with the Orion storage server, as described in detail further below. The purpose of the test framework was to compare the storage performance of the NSULATE block device to the ZFS file system. Because NSULATE is a block device, it is compared to ZFS alone followed by ZFS configured with NSULATE with the ZFS checksumming feature disabled.

Because ZFS is both a file system and a block device, while NSULATE is just a block device designed to augment existing file systems, these benchmarks are not a rigorously objective comparison. NSULATE adds additional hidden metadata to all file system transactions and can amplify reads and writes under certain conditions, as discussed later in this document. NSULATE has its own internal non-volatile cache, which is roughly analogous to the non-volatile cache on RAID controllers but behaves very differently. In the case of the sequential degraded read tests, it can appear to go impossibly fast because it is caching the large reconstructed stripes in memory. ZFS is also a redirect-on-write (ROW) file system, which means its on-disk format is very different from RAID hardware, resulting in broad differences in performance behavior compared to XFS+NSULATE or EXT4+NSULATE.

The ZFS file system is an extremely CPU intensive file system. It consumed the full resources of the two Xeon Gold CPUs (28 cores, 56 threads) during many of the benchmark tests. NSULATE is pinned to a single CPU and consumes the partial resources of a single CPU thread to operate. NSULATE offloads the compute intensive work to the GPU. Disabling ZFS checksumming and error correction also significantly reduces the CPU burden for ZFS when it is paired with NSULATE.

Over the course of performing these benchmarks, we concluded that the most important objective of this whitepaper is to reproducibly illustrate how NSULATE would perform compared to RAID-based storage configurations under common file system benchmarking conditions.

System Configurations

While NSULATE is a dramatic departure from familiar approaches to storage, it is productized in a form that is easy to compare with RAID hardware based solutions. NSULATE is implemented as a Linux block device and can be configured and managed in the same ways as a traditional RAID controller. NSULATE is compatible with the same file systems and applications as any RAID controller. RAID controllers are hardware components that integrate several functions in a single solution, incorporating SAS and RAID control logic and a non-volatile cache on a single board. NSULATE relies on a GPU for storage-processing computations, a separate HBA card to communicate with SAS storage devices and NVDIMMs configured on the host motherboard for non-volatile memory caching and write-hole-protection.

The Orion storage server is configured with 64GB of ECC memory and an additional 64GB of Netlist NVDIMM memory. The NVDIMM memory is used by NSULATE for write-hole-protection and caching, giving NSULATE over eight times the non-volatile cache of any modern RAID card and enabling NSULATE to sustain several seconds of maximum random 4K IOPs. Because of the large disparity between NSULATE and RAID non-volatile cache sizes, we chose to disable the NVDIMM cache for sequential benchmarking to expose the underlying GPU-accelerated storage performance. The NVDIMM cache itself would absorb over 60GB of fast random 4K IOPs before overflowing into physical storage, making benchmarking prohibitively time consuming and uninformative.

Since the Orion server does not require a RAID card to manage its storage-processing, we have chosen to compare a single HBA + GPU versus ZFS software RAID. The Orion storage server is configured as a single 4U storage server with a 100 drive SAS compatible backplane capable of supporting 2.5” SAS SSDs or 3.5” HDDs as our reference test configuration.

The table below describes the system used for performance evaluation. The system is located at Advanced HPC’s Test Lab.

Chassis	Thunder SX FA100-B7118 4U 100 HDD Storage Server
Motherboard	Tyan S7118GMR
Processor	2X Intel® Xeon® Gold 6132 Processor 19.25M Cache, 2.60 GHz
GPU	NVIDIA 900-2G414-0000-000 Tesla P4 8GB GDDR5
HBA	Broadcom SAS3008
RAM	12X 16GB DDR4 RDIMM M393A2K40BB2-CTD 2666 MHz
	4X 16GB Netlist NVvault DDR4 (NV4) 2666 MHz
Resilience	Nyriad NSULATE
Disks	100X Toshiba MGO4ACA 2TB SATA
Network	Intel® Ethernet Converged Network Adapter XL710 10/40 GbE

Array and File System Configuration

In each of the tests, the system was configured to enable the greatest supported resilience and parallelism from the storage array by building a single large storage array out of 100 storage devices, composed of one or more spans.

ZFS-SHA2-512 vs NSULATE-SHA3-512-Merkle Algorithms

Although ZFS supports a SHA2-512 bit mode, it does not store an actual 512 bit checksum, instead truncating the 512 bit checksum to 256 bits. For the purposes of our benchmarks we decided to compare ZFS with SHA2-256 bit checksums to NSULATE's SHA2-256 bit Merkle checksum. Although NSULATE implements a cryptographically more advanced SHA3-512-Merkle hash algorithm, the SHA2-256 comparison is a more balanced benchmark comparison with ZFS.

While XFS does not support real-time checksums with error correction, we benchmarked XFS with NSULATE cryptographic checksums and error correction enabled.

Testing Tools

IOZONE is a respected and widely adopted tool for general file system benchmarking applications. We performed these tests with IOR, FIO and our own internally developed testing tools and concluded that IOZONE produced the most broadly comprehensible results. Because it is impractical to disable all of the layers of caching that take place within modern storage solutions, we chose to try to configure all hardware and software as optimally as possible on the reference Orion server and then run each test at twice the systems available memory. This produces an objective representation of how we would expect the Orion server to perform in the real-world given the tested workloads.

All devices were freshly provisioned and, after being worn in, benchmarked for sequential read and write access patterns. We chose 64K blocks as our reference block size for sequential reads and writes across 8GB test files. All sequential tests were run with 32 threads. For the purposes of this whitepaper we chose to emphasize comparisons of overall file system performance allowing each file system to do it's best to perform optimally.

Tests Performed

RAID Configurations

- **ZFS (+ZRAID60) 10x8:2 with SHA256 checksums**
- **ZFS (+ZRAID30) 5x17:3 with SHA256 checksums**

NSULATE Configurations

- **ZFS-NSL 1x100-m where m = 8, 15, 20 with SHA256 checksums**
- **XFS-NSL 1x100-m where m = 8, 15, 20 with SHA256 checksums**

Degraded Array Benchmarks (M devices)

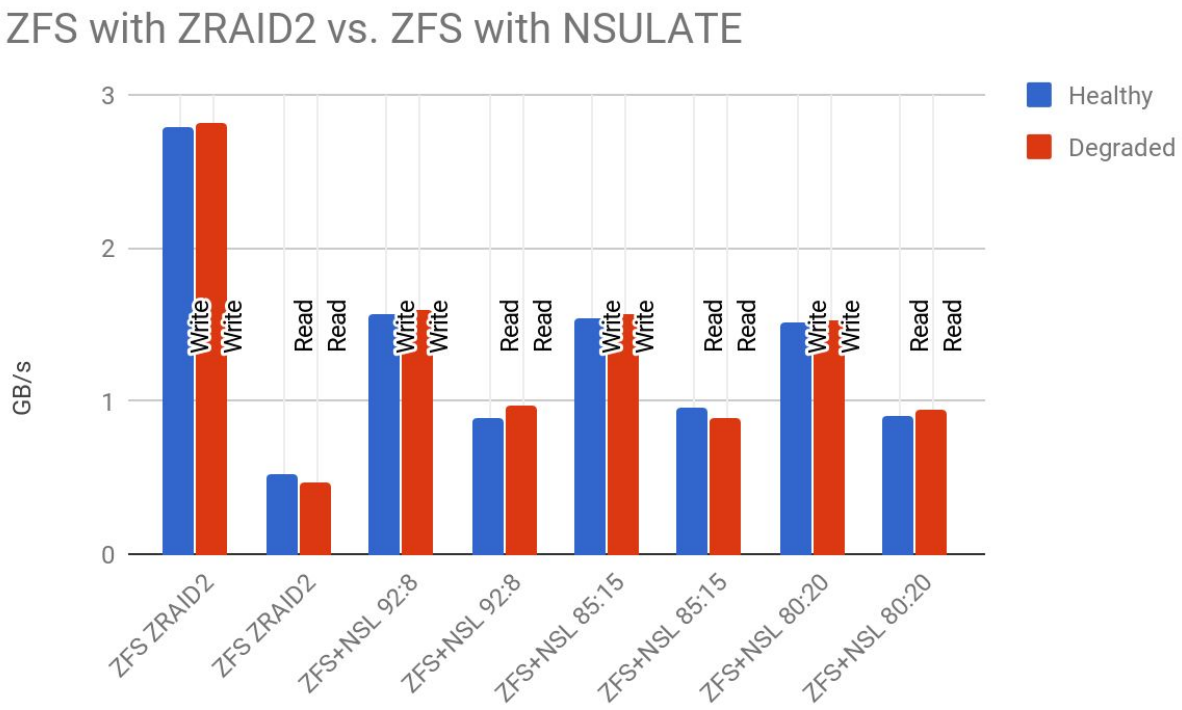
Degraded performance tests are an uncommon benchmark because RAID-based arrays with 2 parity are so precariously close to failure that nobody would reasonably consider operating them for long in that condition. NSULATE supports such high parity levels that leaving an array running in a degraded state becomes a practical consideration. For these tests, we removed the maximum number of drives from each span that each test array could possibly tolerate without failure (M). This configuration forces the CPU or NSULATE (GPU) to perform the maximum amount of computation to correctly recover data from an array with many missing storage devices. Recovering missing data also forces ZFS and NSULATE to read every block in a stripe in order to recover a single missing block.

A ZRAID 60 10x8:2 configuration would have to read 11 blocks of data to recover a single missing block. By comparison, NSULATE configured in a 92:8 array would have to read 92 blocks to recover a single missing block. This would seem to put NSULATE at a significant performance disadvantage; however, as we see from the benchmarks, NSULATE performs well. The reason for this is that NSULATE's large stripes behave like a read-ahead cache in these conditions, such that subsequent sequential reads are in memory after a single block in a stripe has been recovered.

Results

ZFS (+ZRAID60) 10X8:2 vs ZFS+NSULATE

In the first chart, we compare the performance of a ZFS RAID 60 array with software checksumming versus comparable ZFS+NSULATE configurations with NSULATE's GPU- accelerated SHA2-256-Merkle checksumming and high parity erasure coding enabled. NSULATE adds a small amount of additional metadata to every storage transaction to accommodate the higher parity and cryptographic checksums.

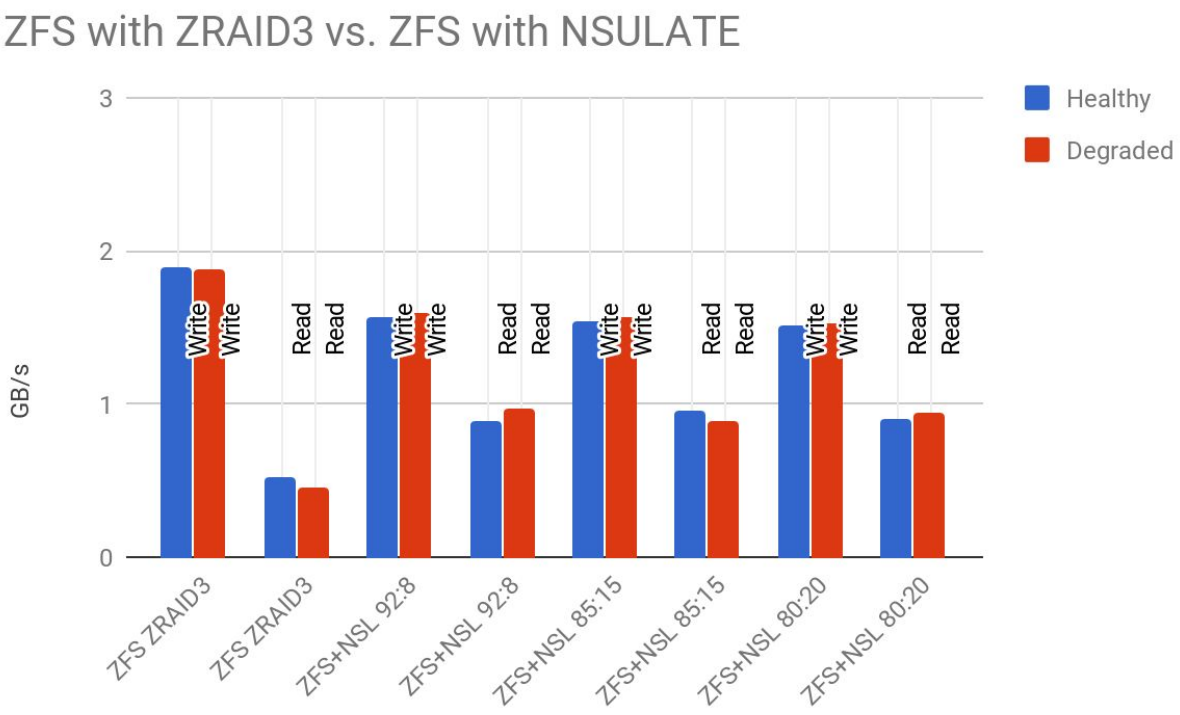


Write speed appears to be impacted by the additional metadata storage associated with adding checksums to every block, but ZFS read speeds are accelerated. While the GPU compute time to produce or check the SHA2-256-Merkle hash is negligible, writing it to disk incurs some additional write overhead. In high parity configurations, read performance benefits from better parallelism, greater information density per stripe and a read-ahead advantage when degraded reads have to be recovered from an entire stripe. This leaves them cached for subsequent sequential reads.

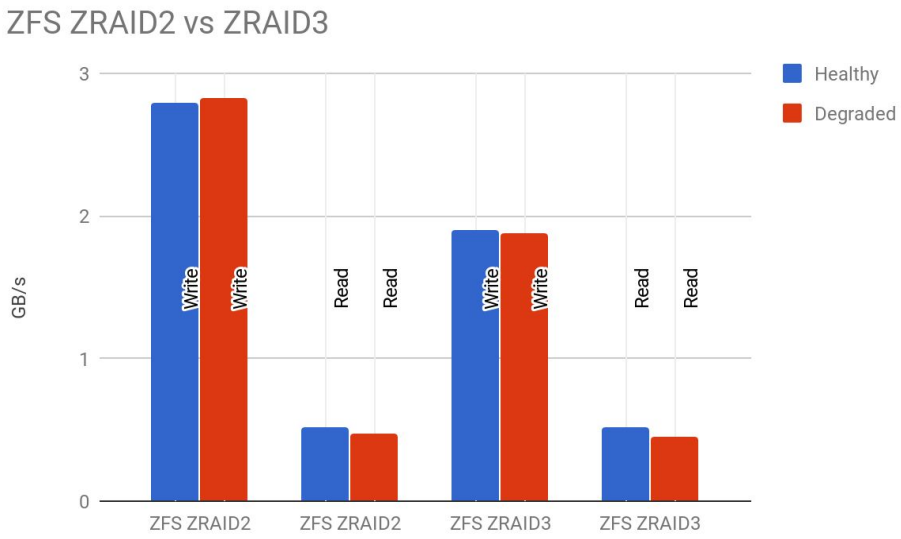
Even with extremely high parity, a degraded NSULATE array with 20 randomly disabled drives maintained performance. Although the ZFS ZRAID2 array performed better at writes, it consumed most of the compute resources of two XEON Gold processors with 14 cores each to maintain its degraded performance, or over 10 times the CPU resources of NSULATE for the same benchmarks.

ZFS (+ZRAID30) 5X17:3 vs. ZFS+NSULATE

For this benchmark, we compared ZFS in a software ZRAID 30 configuration with software SHA2-256 checksums versus ZFS+NSULATE with ZFS checksumming disabled and substituted for NSULATE SHA2-256 Merkle checksumming and higher parity error correction.



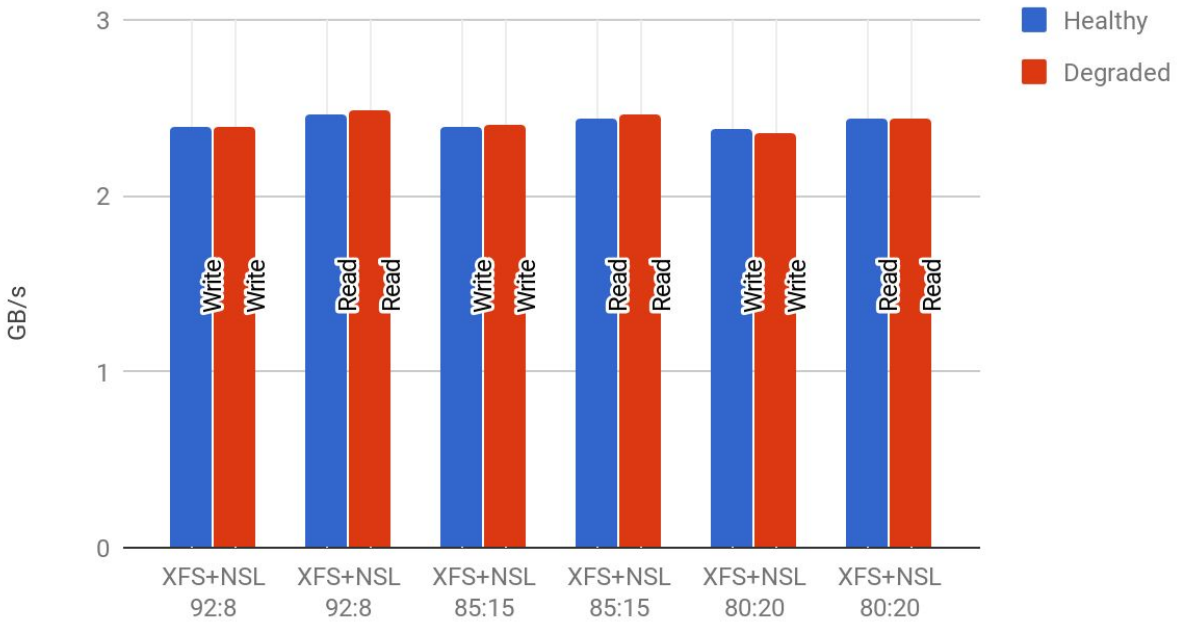
While NSULATE's performance remains consistent across high parity and degraded conditions, the additional parity calculation for ZFS ZRAID3 in software slows its writes down dramatically and slows its reads down noticeably in degraded conditions compared to its ZRAID2 performance.



NSULATE with EXT4, XFS and ZFS file systems

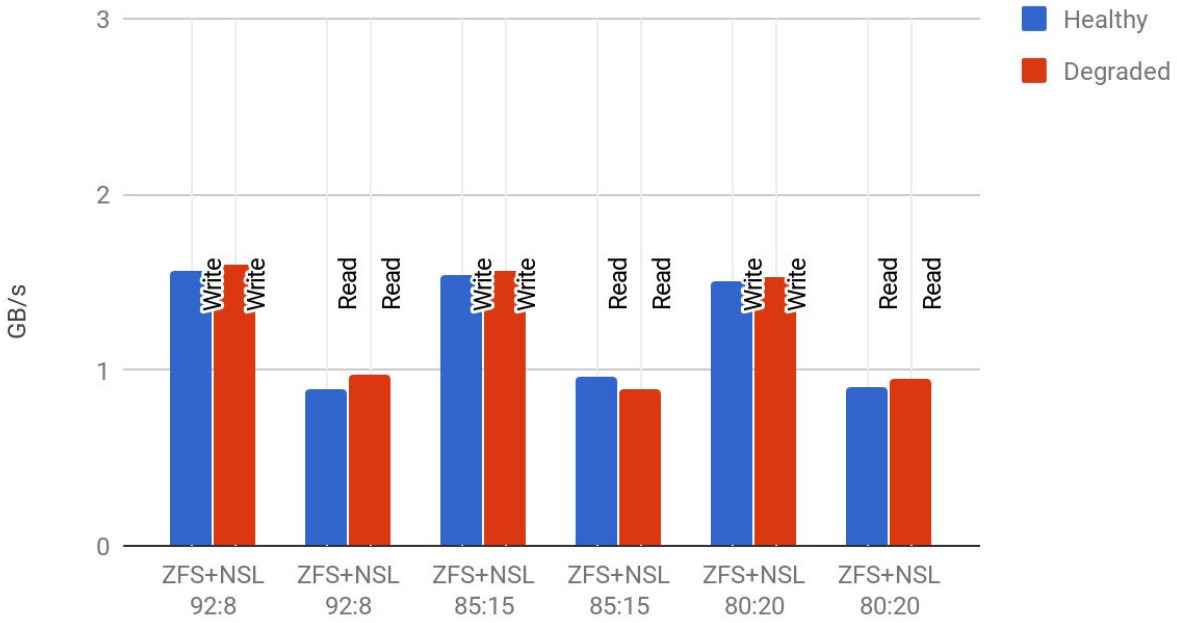
Here we compare the performance of three widely used file systems with NSULATE providing GPU-accelerated checksumming and high parity error correction. Each file system is popular for certain specialized applications because of the differences in their relevant feature sets and performance characteristics. Configuring them with NSULATE, dramatically alters their properties. EXT4 & XFS inherit cryptographic checksums and real-time error correction from NSULATE with little or no performance compromises, while NSULATE dramatically reduces ZFS CPU overhead and increases its resilience capabilities.

NSULATE + XFS

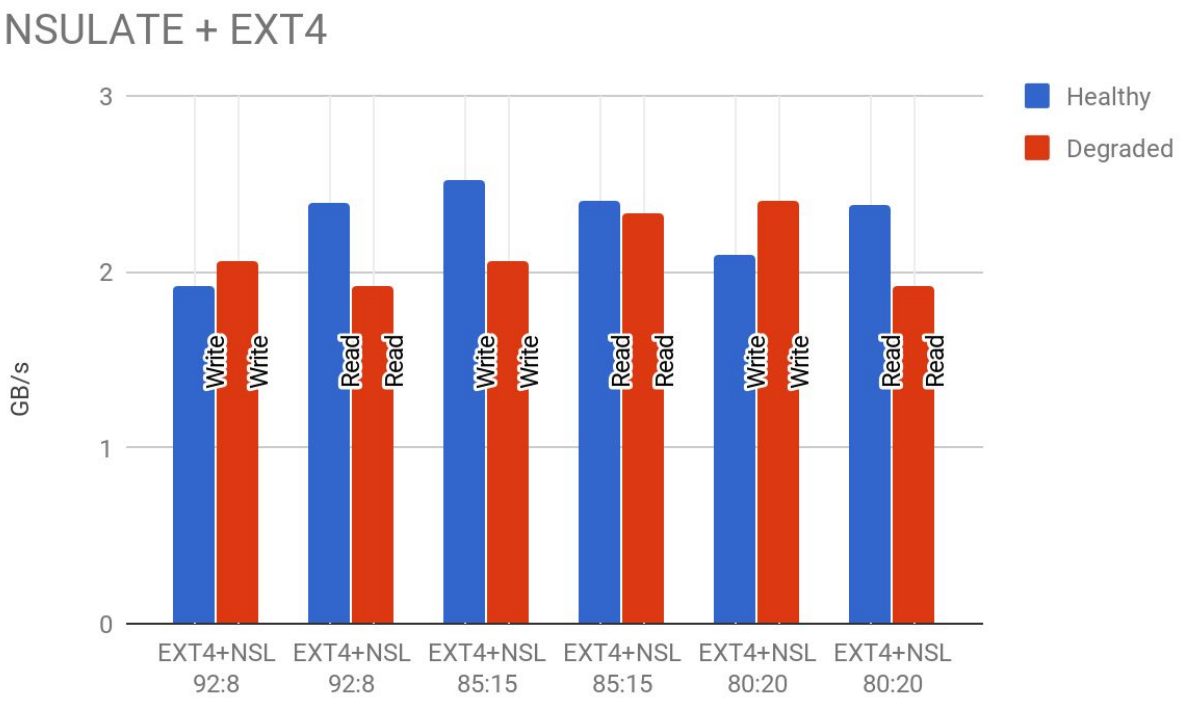


XFS performs consistently under all NSULATE parity configurations, even in maximum degraded states. Degraded performance is often a little faster because NSULATE has to write less data to a degraded array and degraded reads are cached when they have to be recomputed from parity in real-time.

NSULATE + ZFS



While ZFS write performance is a little lower with NSULATE, it's CPU utilization is reduced by an order of magnitude. It's likely that NSULATE adding metadata to the ZFS data writes adds some overhead to ZFS data placement. ZFS read performance is nearly twice as fast due to NSULATE's large stripes resulting in more efficient read-ahead caching.



Conclusions

GPU-accelerated NSULATE arrays generally exhibit performance comparable to traditional hardware and software RAID solutions while enabling much higher levels of data resilience, array scalability and storage efficiency. Write performance can be compromised in some circumstances due to the addition of additional checksum metadata to all data blocks. Read performance can often be accelerated due to NSULATE's use of larger stripes and more non-volatile cache memory resulting in better read-ahead performance. For compute intensive file systems like ZFS, NSULATE can dramatically reduce CPU overhead, freeing the CPU for other tasks, while adding powerful error correction capabilities to lean, fast file systems like EXT4 and XFS without adversely impacting their performance.

Cost Analysis

We avoided detailed cost analysis for this whitepaper because NSULATE is designed to go beyond RAID in capabilities making direct comparisons difficult. NSULATE makes minimal use of CPU resources and generally performs best pinned to a single NUMA node. Compute intensive filesystems like ZFS need a great deal of CPU resources and memory to perform computing for all the features they support, hence ZFS made intensive use of the CPU during the benchmarks. NSULATE, on the other hand, needs a larger non-volatile memory cache compared to RAID hardware for write-hole-protection and stripe assembly. The most important cost distinction between an ordinary RAID hardware-based or software RAID solution and an NSULATE accelerated one with high parity is the leap in physical storage efficiency. As illustrated earlier, a 92:8 array is more resilient to drive failure than a 10X8:2 RAID array saving 12 drives in otherwise wasted storage space.

In general we found that a little more parity can increase array resilience significantly with minimal impact on performance. Higher parity configurations, although relatively free to configure, may have an impact on random performance, which is alleviated by the large non-volatile memory cache configured with the Orion storage server.

NYRIAD™
we make every bit count

References

https://storagemojo.com/2010/02/27/does-raid-6-stops-working-in-2019/
https://queue.acm.org/detail.cfm?id=1670144
http://www.iozone.org/