Flash Memory Summit

# RAIN: Reinvention of RAID for the World of NVMe

Dmitrii Smirnov
Principal Software Developer
smirnov.d@raidix.com

RAIDIX LLC

Flash Memory Summit

# About the company

**RAIDIX is an innovative solution provider and developer of high-performance storage systems.**

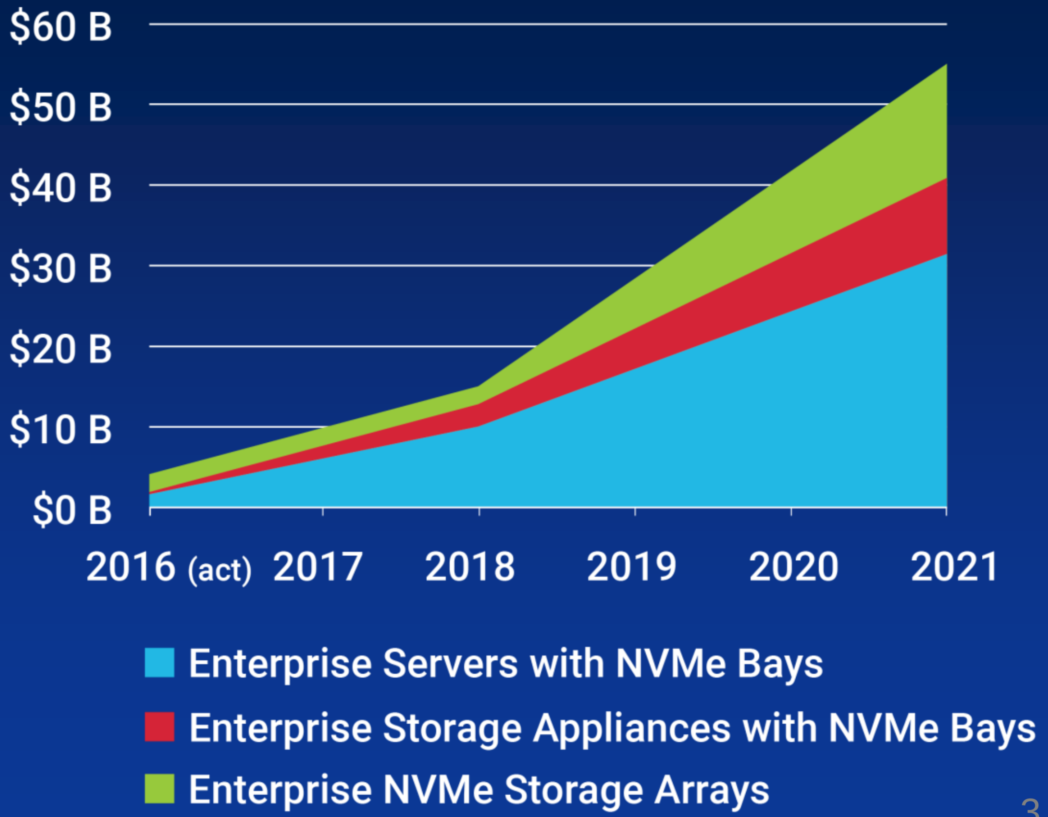
**Patented erasure coding methods and innovative technology create core value of our products.**

**Strategic partners**

# NVMe Market Overview

- The market of enterprise storages and servers with NVMe bays will grow up to $40B by 2020.
- More than 50% of servers will have NVMe bays by 2020.
- Market needs software to employ new hardware capabilities!

*G2M report, Michael Heumann, May 10, 2018



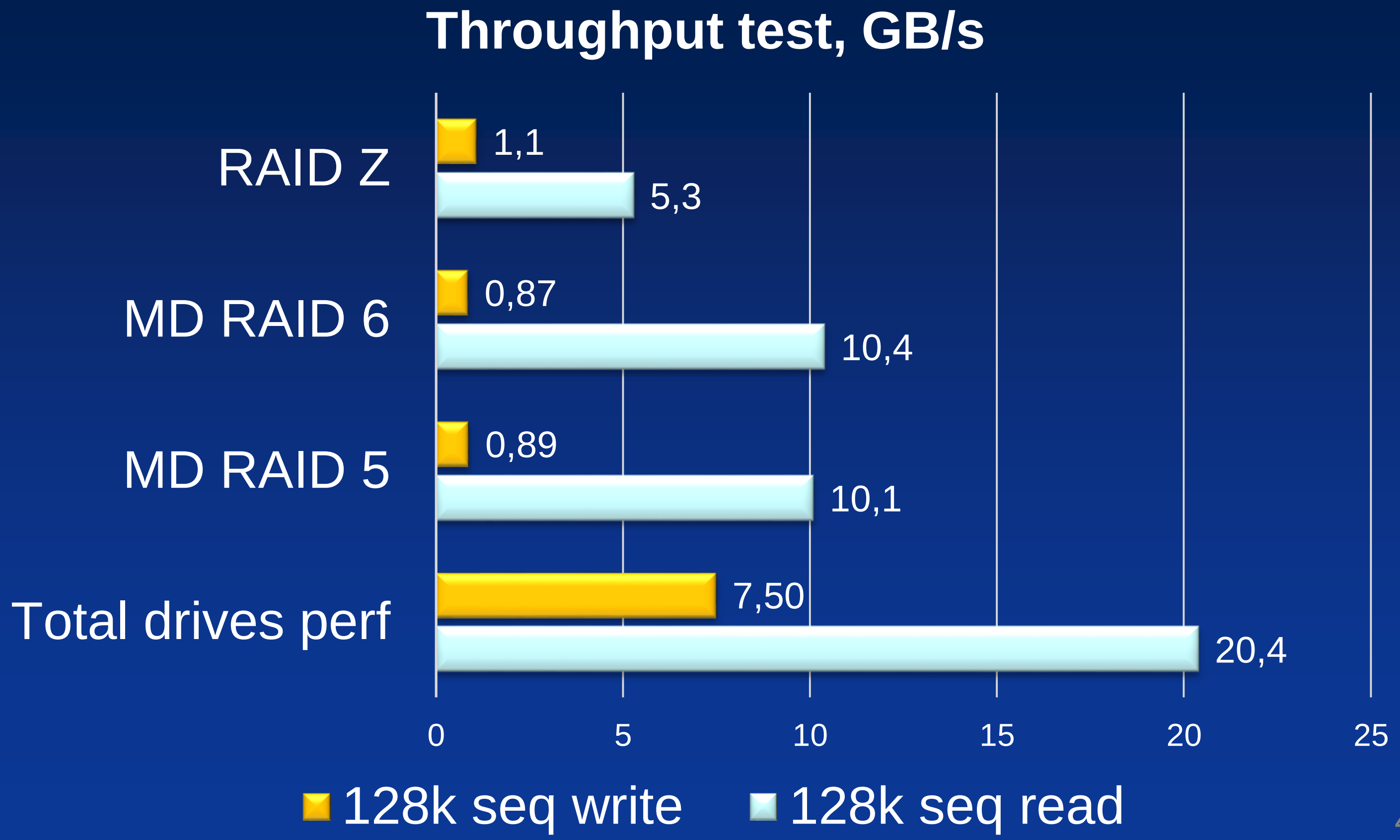
# Is existing software suitable for NVMe?

Flash Memory Summit

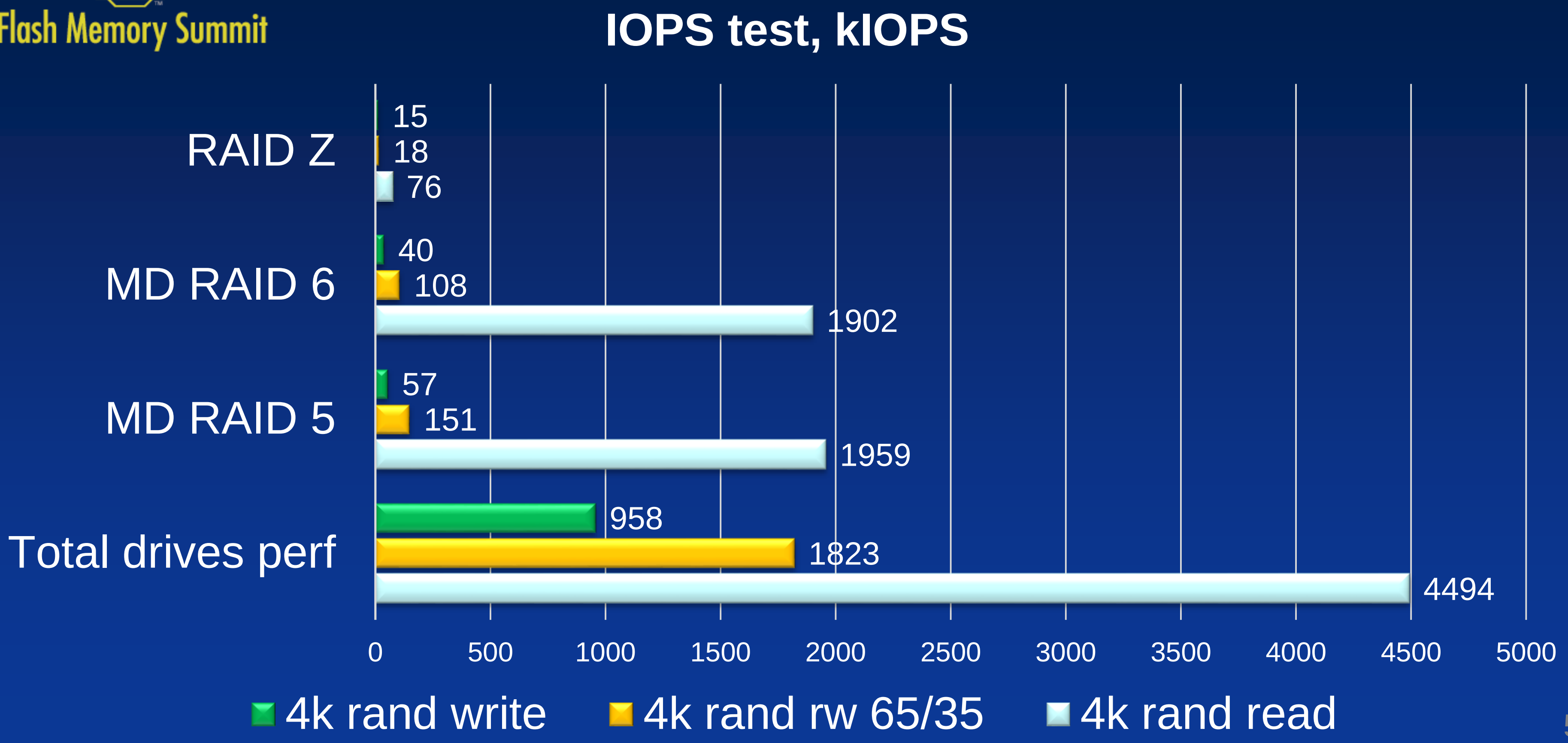
We have benchmarked mdraid and zfs pools.

12 NVMe devices.

Tests are based on SNIA SSS PTSe.



# Is existing software suitable for NVMe?



# New product vision

**Our product**

RAIDIX ERA – software RAID optimized for NVMe

**Goals**

- High performance
  For single RAID 6:
  - Up to 30 GBps
  - Up to 4 000 000 IOPS
  - Latencies < 0.5 ms
- Low CPU overhead
- Low memory usage
  - No cache
  - No data copy on datapath
- Flexibility
  - Local and network drives
  - Distributed RAID
  - Media vendor agnostic
  - POSIX API
  - Block device
- No performance loss in degraded RAID state
- Free version

# Kernel or not kernel

**User level drivers**

+ Remove system call switch overhead
+ Simplify management of block IO
+ Ensure direct access to NVMe
– Lose POSIX API: need to rewrite applications and file systems

**Linux kernel drivers**

+ Provide block device and support POSIX API: no need to rewrite applications and file systems
+ Provide higher in-kernel performance on newer 4.x kernels with system call optimizations
– Linux kernel block layer still needs to be optimized for more IOPS

# Product architecture

**Components**

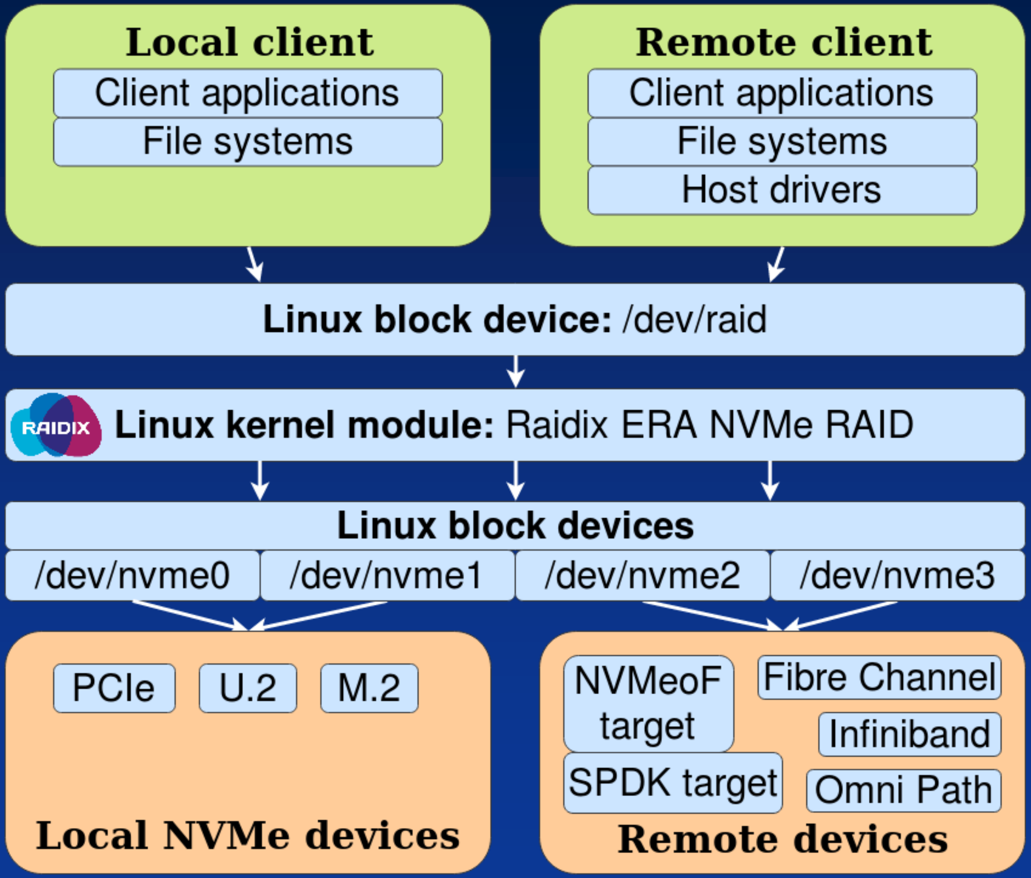
- Linux kernel driver
- RAID management utility

**Installation**

- Deployed using rpm or deb

**Interaction**

- RAID works with block devices
- RAID provides a block device



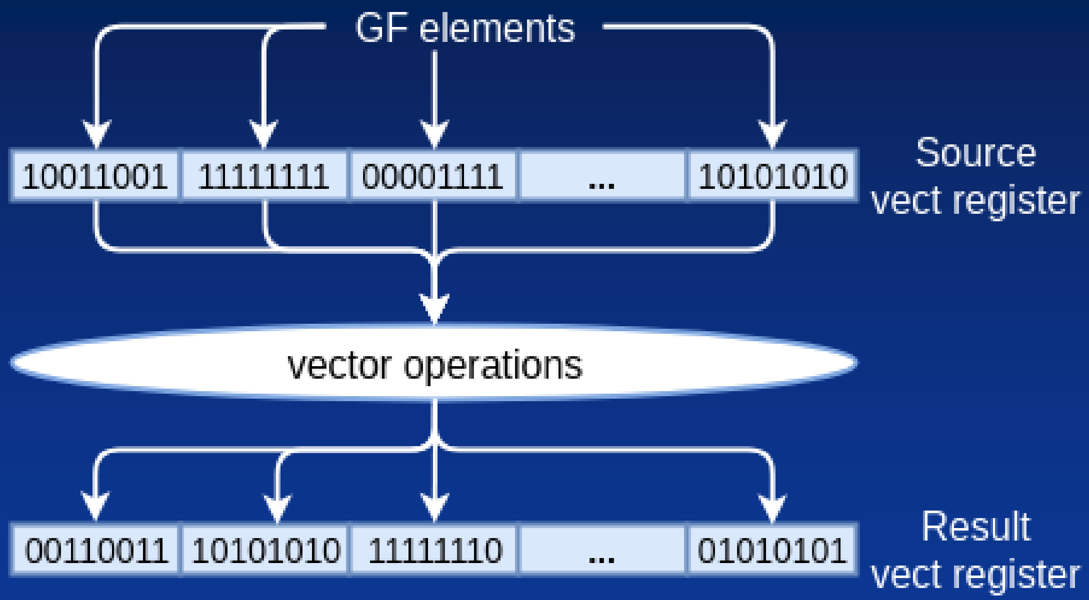
# Performance principles

- High performance of RAID checksums calculations and data recovery
  - necessary for performance in degraded state
- Lockless datapath
- High IO handling parallelization without scheduling
- Efficient data transfer with zero-copy
- In-kernel tools:
  - per CPU cache aware efficient memory allocator - kmem_cache
  - lockless list
  - stable and high performance nvmeof target and host drivers

# RAID Calculation Engine

Standard approach to calculation vectorization

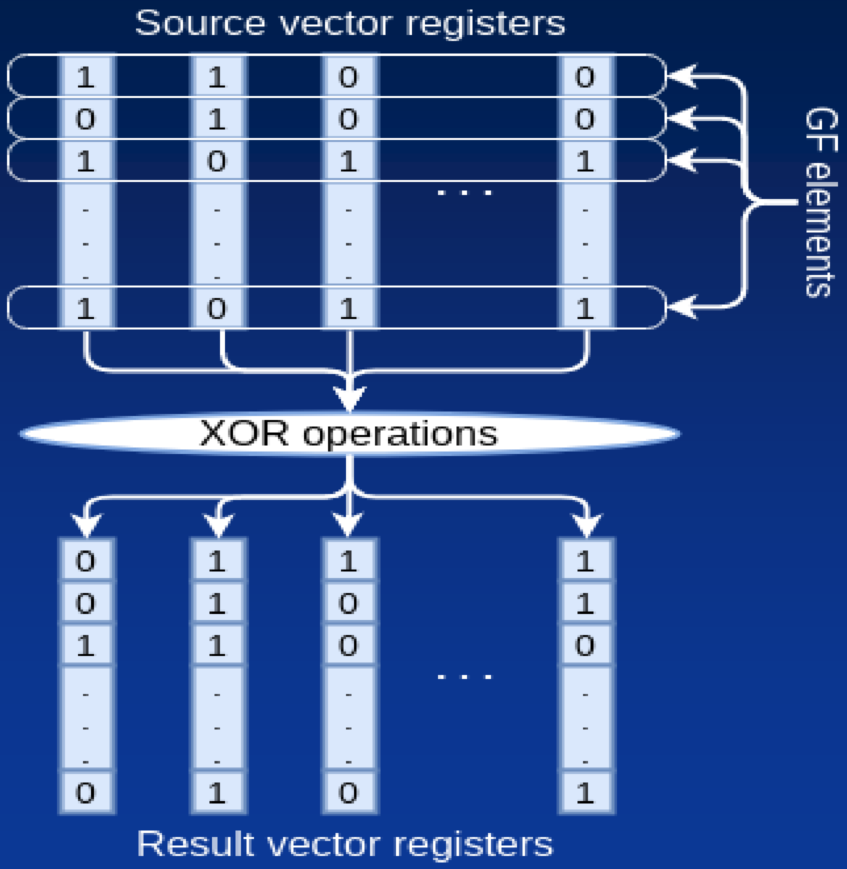
- Vector register packs Galois Field elements
- Packed shift operations
- Packed logical operations (XOR, AND)
- Shuffle operations



# RAID Calculation Engine

Our approach to calculation vectorization

- Vector contains bits of different Galois Field elements
- Only packed XORs
- Less data move operations
- Less vector operations

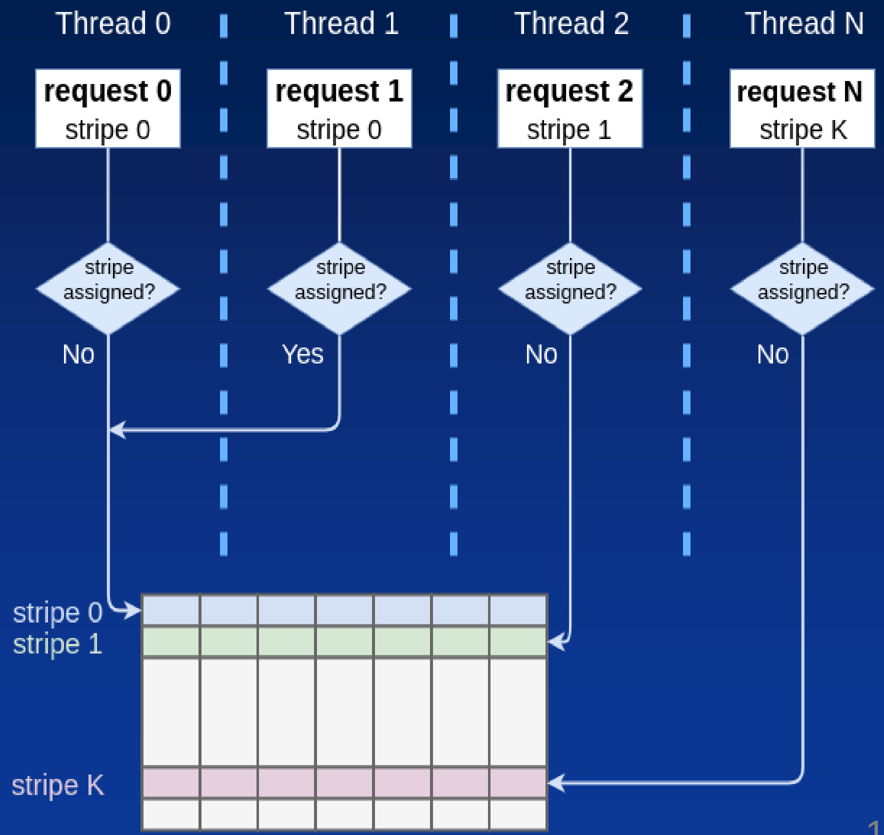


# IO Handling

**Challenge**
To update RAID checksum in multithreaded workloads

**Why**
Threads working with the same stripe can corrupt shared checksums

**Our solution**
To avoid locks by <u>dynamic</u> mapping stripes to threads responsible for its handling



# Performance test configuration

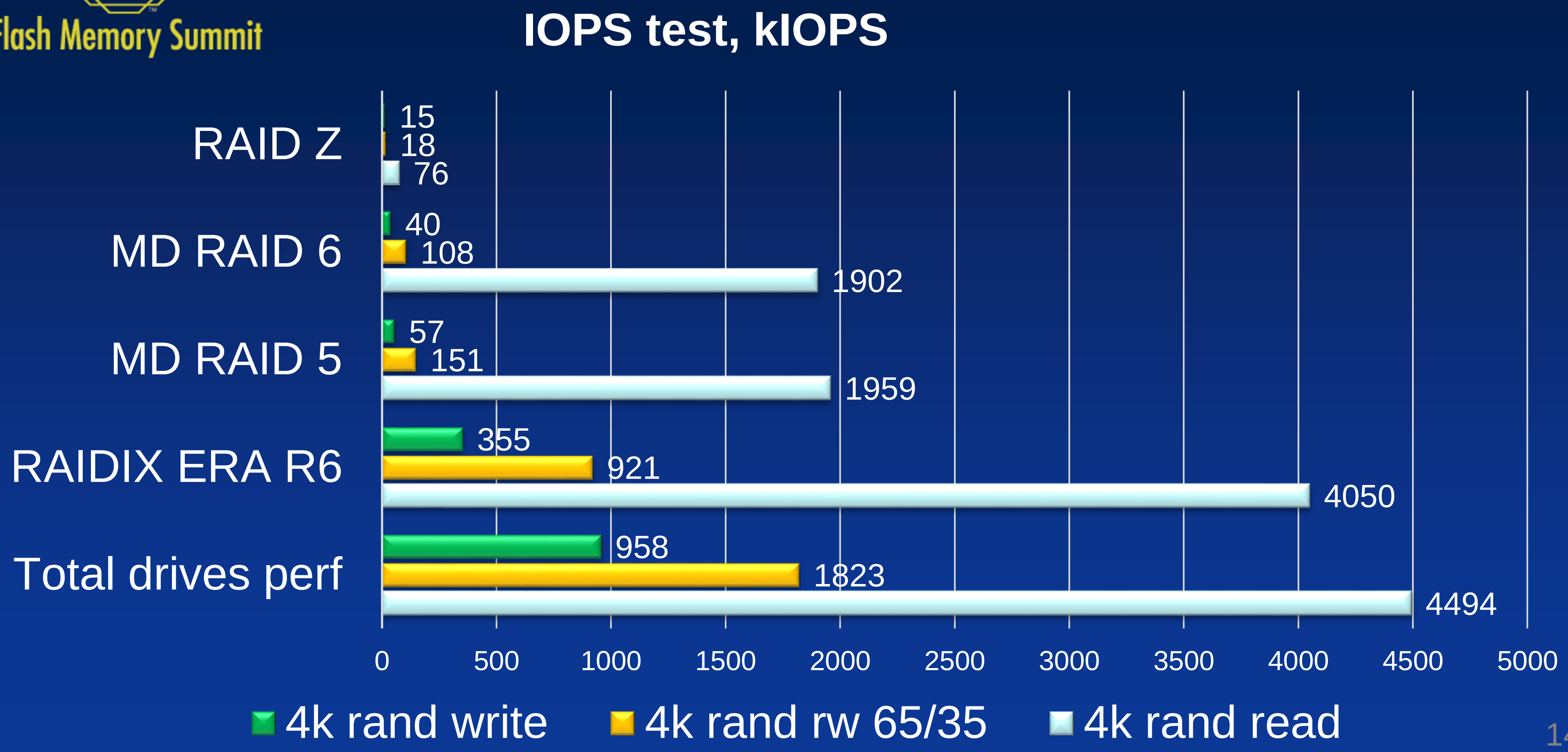
System configuration

- Intel Xeon Gold 6130 CPU @ 2.10GHz
- 12 NVMe: Intel SSD DC D3700 Series
- Hyperthreading and NUMA enabled
- Centos 7.4, Linux Kernel 4.11.6-1.el7.elrepo.x86_64
- RAID 6

Tests based on SNIA SSS PTSe

- Iodepth 32, Numjobs 64
- IOPs test
- Latency test

# IOPS test


IOPS test, kIOPS
RAID Z
15
18
76
MD RAID 6
40
108
1902
MD RAID 5
57
151
1959
RAIDIX ERA R6
355
921
4050
Total drives perf
958
1823
4494
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
4k rand write
4k rand rw 65/35
4k rand read

# IOPS and latency tests

For single RAID 6:

- ✓ Up to 4 000 000 IOPS
- ✓ Latencies < 0.5 ms

| Average IOPS (kIOPS) | | | | | |
|---|---|---|---|---|---|
| Block Size (KiB) | Random Read / Write Mix % | | | | |
| | 0/100 | 35/65 | 50/50 | 65/35 | 100/0 |
| 4 | 355 | 486 | 619 | 921 | 4050 |
| 8 | 180 | 249 | 320 | 520 | 2510 |
| 32 | 59 | 83 | 116 | 167 | 640 |
| 128 | 14 | 20 | 30 | 40 | 160 |
| 1024 | 3 | 4 | 6 | 7 | 19 |

| Average Response Time (ms) | | | |
|---|---|---|---|
| Block Size (KiB) | Random Read / Write Mix % | | |
| | 0/100 | 65/35 | 100/0 |
| 4 | 0.16 | 0.13 | 0.10 |
| 8 | 0.20 | 0.16 | 0.13 |
| 16 | 0.31 | 0.22 | 0.18 |

# Challenges

**Performance challenge #1**

Initial architecture idea was to avoid locks by <u>permanent</u> mapping stripes to threads responsible for its handling.

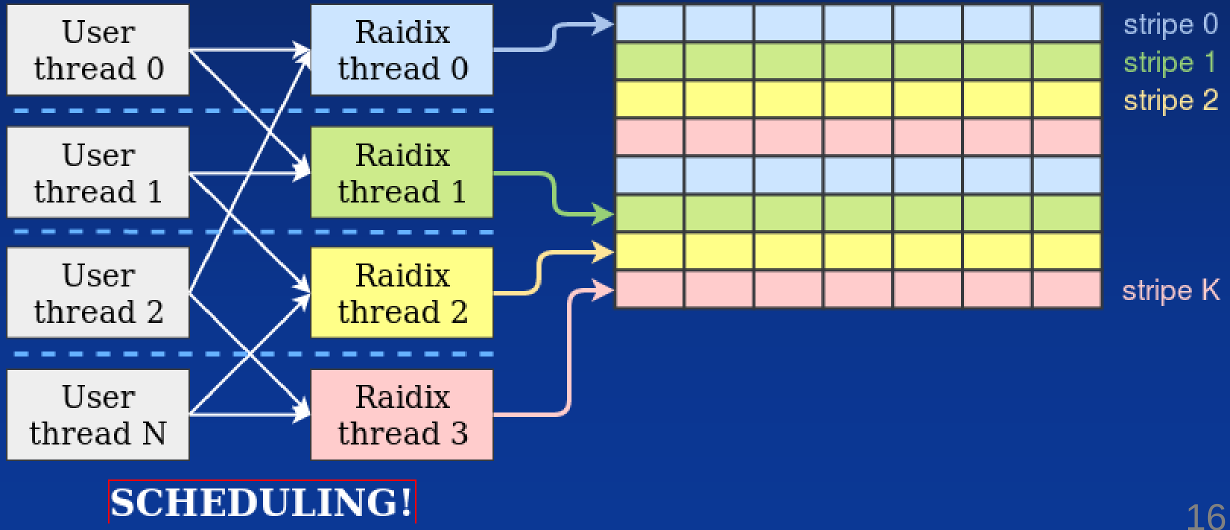
It resulted in two times less performance than our goals.

**Problem**

Scheduling on datapath

**Solution**

Architecture without scheduling



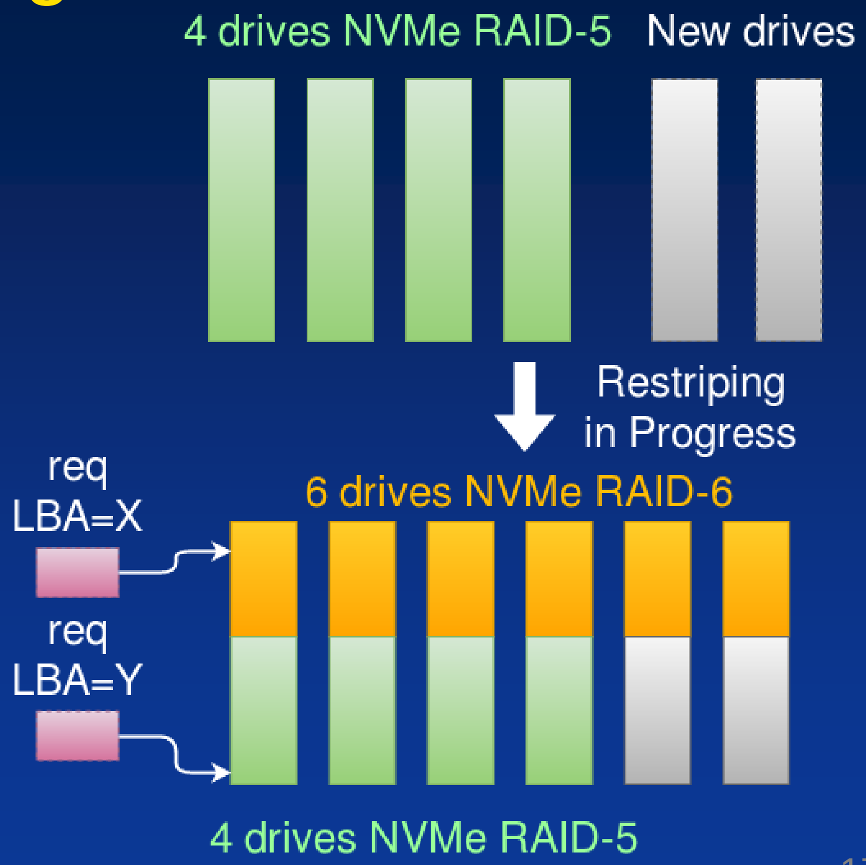
Flash Memory Summit

# Challenges

**Performance challenge #2**
Keep high IO performance while scaling RAID to new devices

**Problem**
RAID in 2 configurations should handle IO in both parts without latency degradation

**Solution**
Background restriping with non-blocking restriping window



# What is next?

- Add LRC and Regeneration codes for distributed RAID
  - Reduce number of reads for faster single failure recovery
- Integrate with existing volume managers or create a new one
  - Linux volume manager (LVM), SPDK lvol, ZFS vol, etc.
- Optimize performance for 3.x kernels

# Thank you

RAIDIX LLC

request@raidix.com