Flash Memory Summit

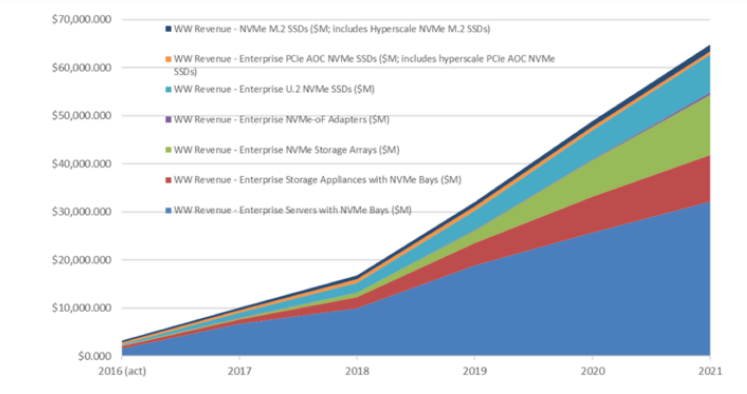
RAIN: Reinvention of RAID for the World of NVMe

Sergey Platonov
RAIDIX

NVMe Market Overview

- > 15 vendors develop NVMe-compliant servers and appliances
- > 50% of servers will have NVMe slots by 2020

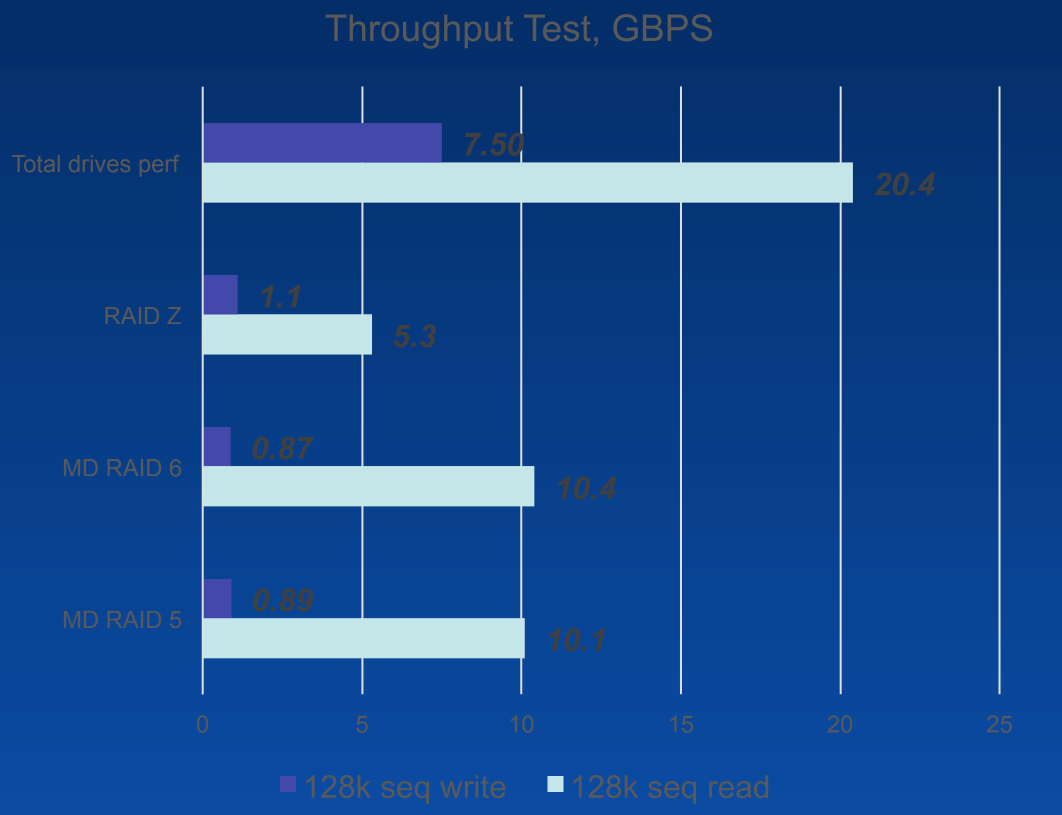
Market needs software to employ new hardware capabilities!



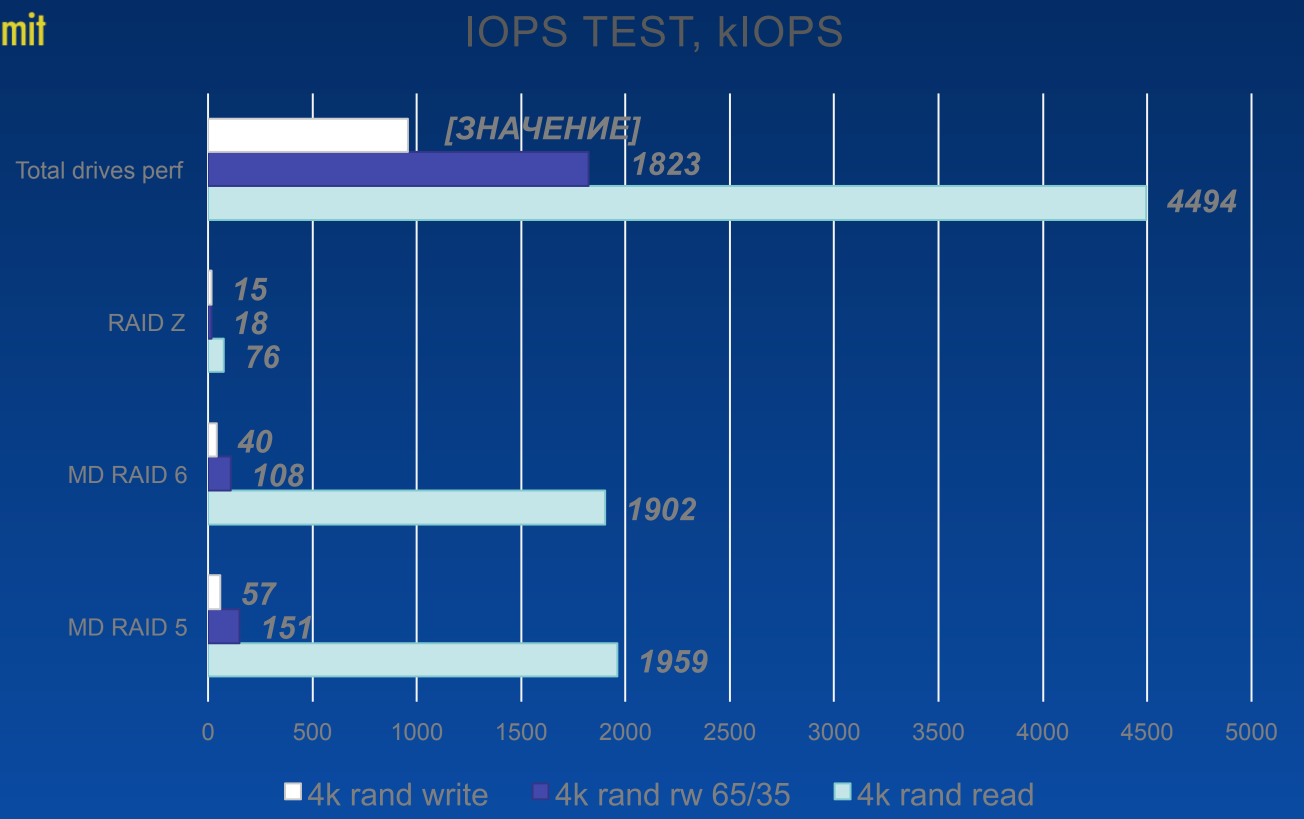
Is existing software suitable for NVMe?

We have benchmarked mdraid and zfs pools.

Tests are based on SNIA SSS PTSe.



Is existing software suitable for NVMe?



Flash Memory Summit

Kernel or not kernel

User level drivers

- + Remove system call switch overhead
- + Simplify management of block IO
- + Ensure direct access to NVMe
- − Lose POSIX interface and trigger obligatory application rewrite

Linux kernel drivers

- + Provide block device and support POSIX interface: no need to rewrite applications and file systems
- + Show 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

Flash Memory Summit

New product vision

Our product

Software RAID optimized for NVMe in Linux Kernel

Goals

- High performance

  For single RAID 6 :
  - Up to 30 GBps
  - Up to 4 000 000 IOps
  - Latencies < 0.5 ms
- No performance loss in degraded RAID state
- Low CPU overhead
- Memory prudent
  - No cache
  - No data copy on datapath
- Flexibility
  - Local and network drives
  - Media vendor agnostic

Product architecture

Components

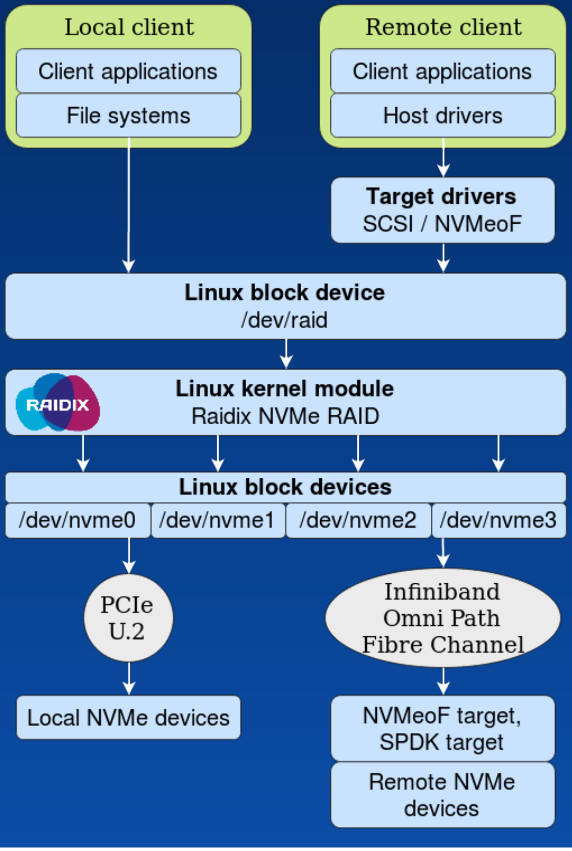
- Linux kernel driver
- RAID management utility

Installation

- Deployed using rpm or deb

Interaction

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



Flash Memory Summit

Performance principles

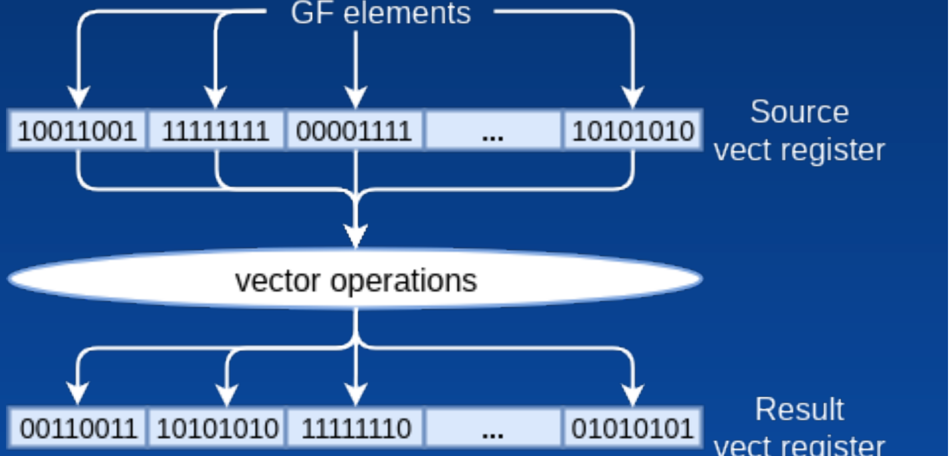
- High performance of RAID checksums calculations and data recovery
  - necessary for performance in degraded state
- Lockless datapath
- High IO handling parallelization without scheduling
- Efficient data transfer with zero-copy
- In-kernel tools:
  - per CPU cache aware efficient memory allocator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

Flash Memory Summit

RAID Calculation Engine

Standard approach to calculation vectorization

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



GF elements

10011001	11111111	00001111	...	10101010

Source vect register

vector operations

00110011	10101010	11111110	...	01010101

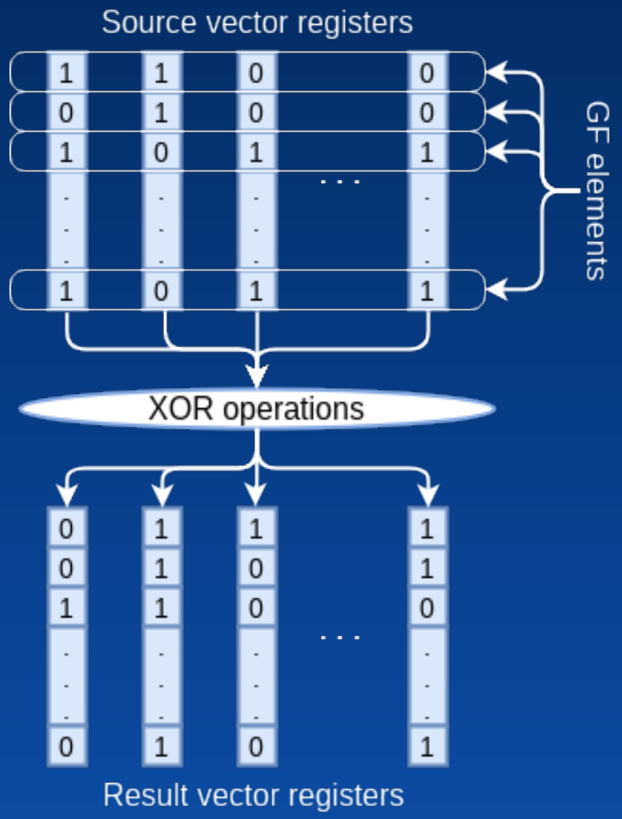
Result vect register

Flash Memory Summit

RAID Calculation Engine

Our approach to calculation vectorization

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

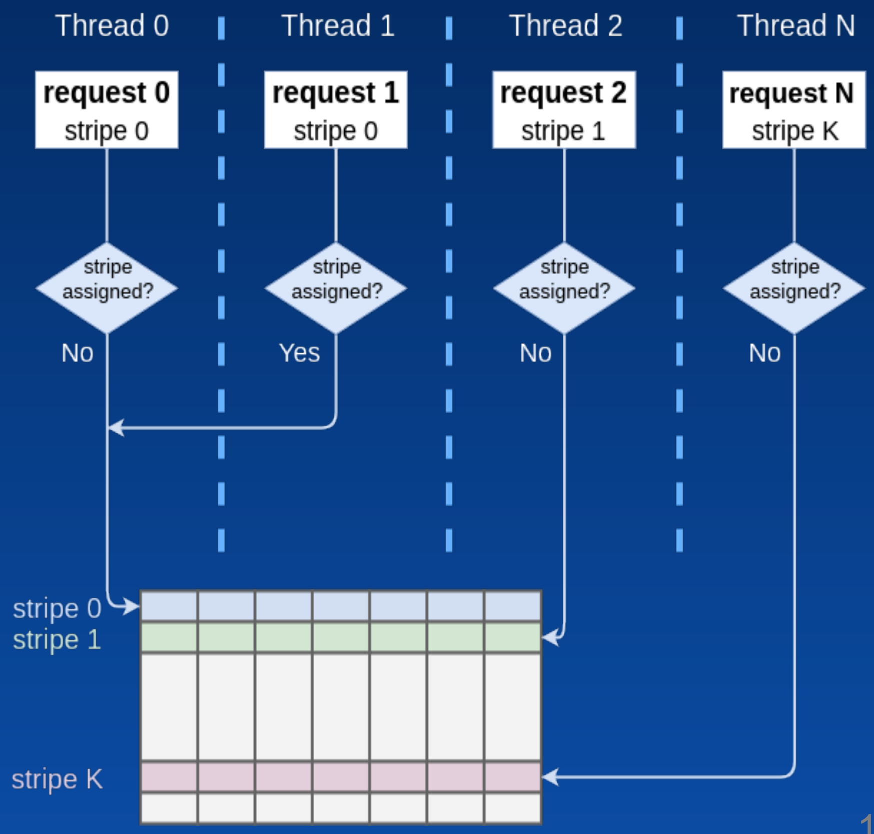


IO Handling

Challenge
Update of RAID checksum
in multithreaded workloads

Why
Threads working with the same
stripe can corrupt shared checksums

Our solution
To use lockless algorithms for
calculation of checksum in the thread
that is responsible for the stripe
calculations at the moment



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

Flash Memory Summit

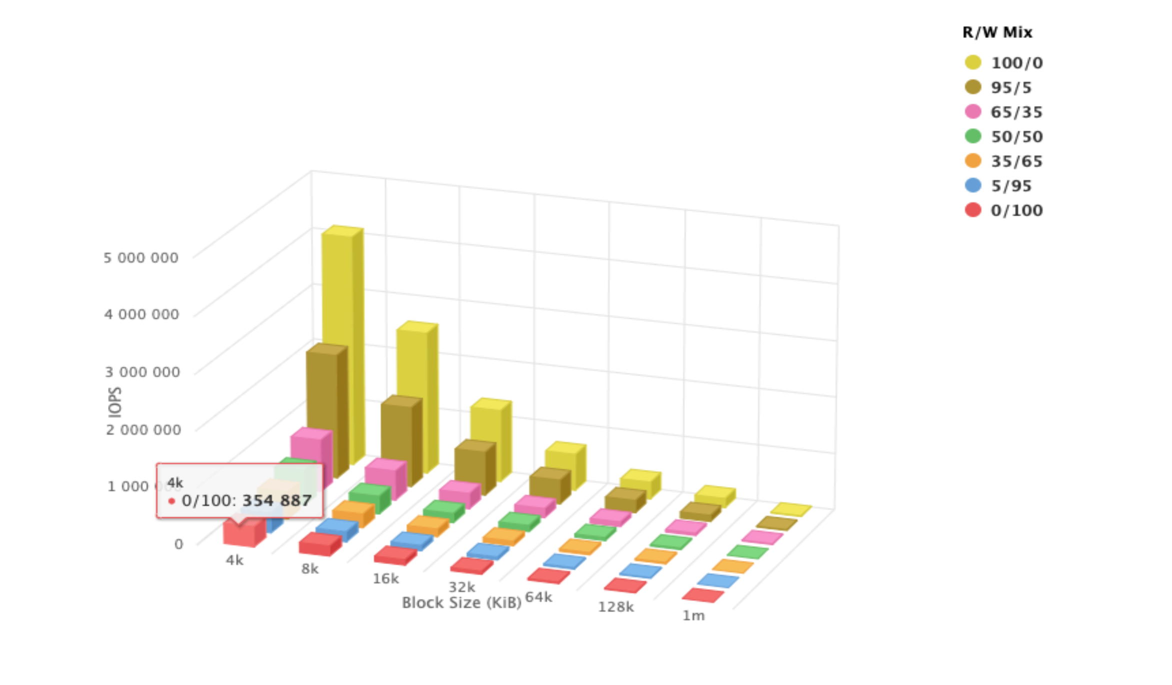
Performance testing results

Flash Memory Summit

IOPs test

Block Size (KiB)	Read / Write Mix %						
	0/100	5/95	35/65	50/50	65/35	95/5	100/0
4k	354887	363830	486865.6	619349.4	921403.6	2202384.8	4073187.8
8k	180914.8	185371	249927.2	320438.8	520188.4	1413096.4	2510729
16k	92115.8	96327.2	130661.2	169247.4	275446.6	763307.4	1278465
32k	59994.2	61765.2	83512.8	116562.2	167028.8	420216.4	640418.8
64k	27660.4	28229.8	38687.6	56603.8	76976	214958.8	299137.8
128k	14475.8	14730	20674.2	30358.8	40259	109258.2	160141.8
1m	2892.8	3031.8	4032.8	6331.6	7514.8	15871	19078

IOPs test

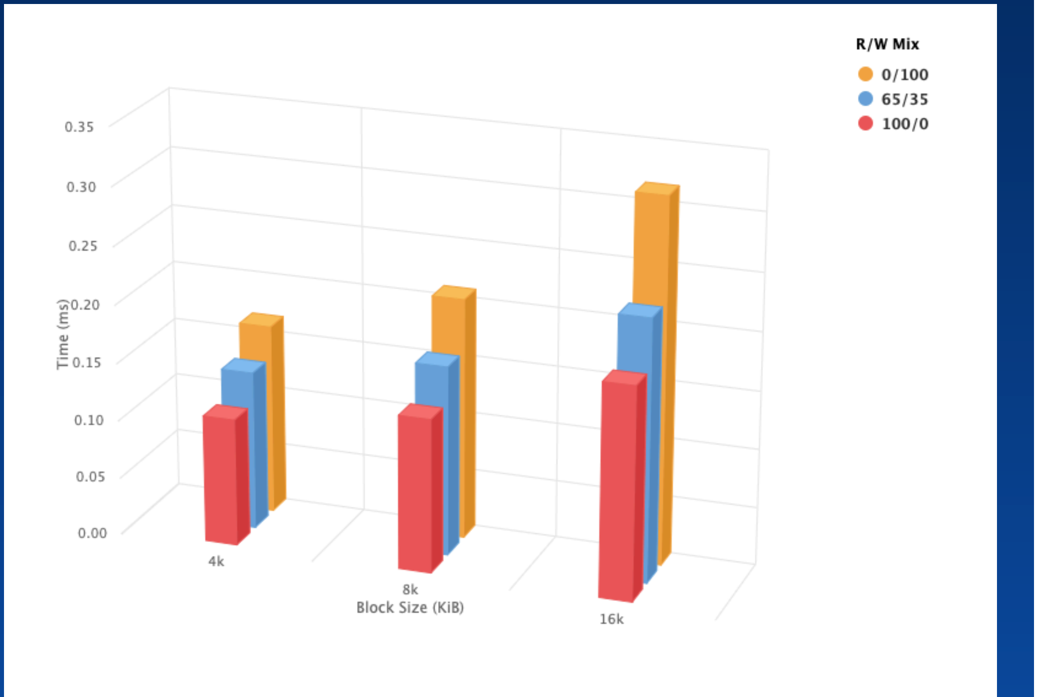


Plot A.6 - IOPS -All RW Mix & BS - 3D Columns

Flash Memory Summit

Latency test

Average Response Time (ms)			
Block Size (KiB)	Read / Write Mix %		
	0/100	65/35	100/0
4k	0.16334	0.136397	0.10958
8k	0.207056	0.163325	0.132586
16k	0.313774	0.225767	0.182928



Plot A.11 - Average Latency vs. BS and R/W Mix - 3D Plot

Flash Memory Summit

Challenges

Performance challenge #1

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

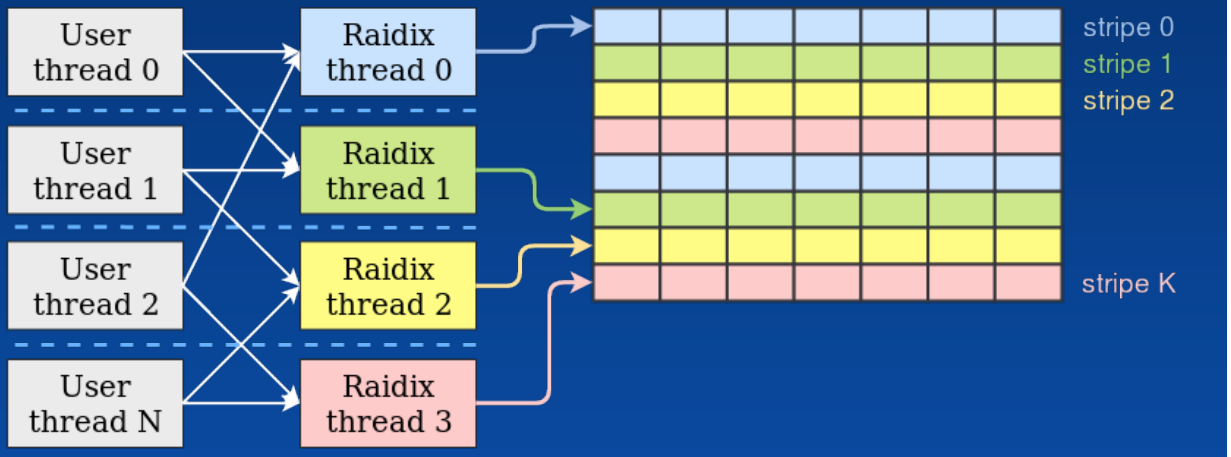
It resulted in two times less performance than our goals.

Problem

Scheduling on datapath

Solution

Architecture without scheduling



User thread 0

User thread 1

User thread 2

User thread N

Raidix thread 0

Raidix thread 1

Raidix thread 2

Raidix thread 3

stripe 0

stripe 1

stripe 2

stripe K

SCHEDULING!

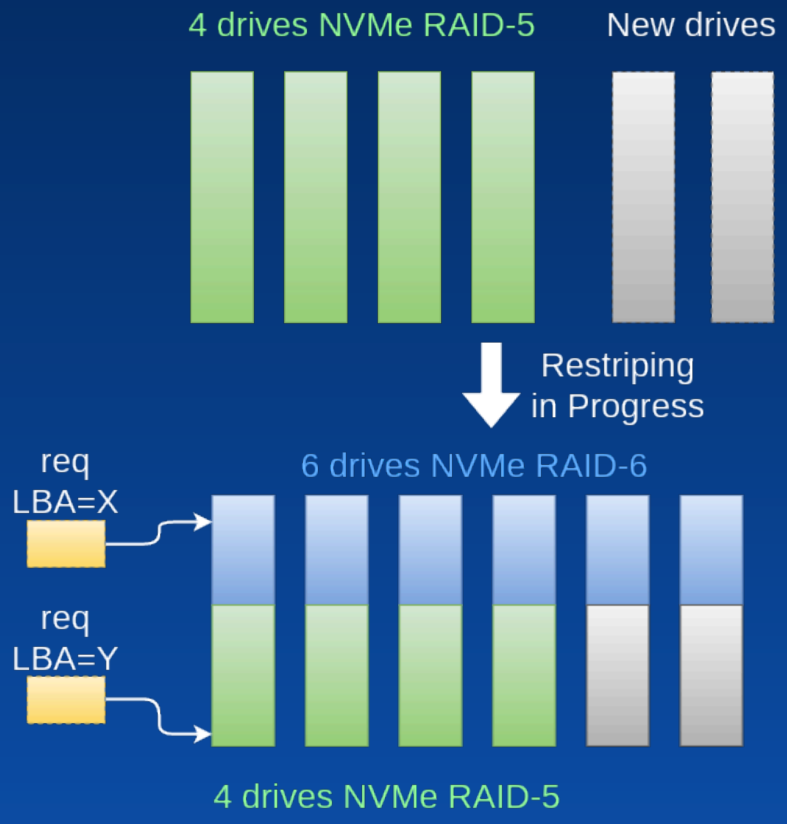
Flash Memory Summit

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



Flash Memory Summit

What is next?

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