RAIN: Reinvention of RAID for the World of NVMe

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

RAIDIX LLC
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

- Intel
- HGST (a Western Digital brand)
- AIC
- GIGABYTE™
- Mellanox Technologies
- Panasonic
- QLOGIC™
- EchoStreams
- Broadcom
- ATTO
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?

We have benchmarked mdraid and zfs pools.

12 NVMe devices.

Tests are based on SNIA SSS PTSe.

- RAID Z: Throughput test, GB/s
  - 128k seq write: 1.1
  - 128k seq read: 5.3

- MD RAID 6: Throughput test, GB/s
  - 128k seq write: 0.87
  - 128k seq read: 10.4

- MD RAID 5: Throughput test, GB/s
  - 128k seq write: 0.89
  - 128k seq read: 10.1

- Total drives perf: Throughput test, GB/s
  - 128k seq write: 7.50
  - 128k seq read: 20.4
Is existing software suitable for NVMe?

IOPS test, kIOPS

RAID Z
- 15
- 18
- 76

MD RAID 6
- 40
- 108

MD RAID 5
- 57
- 151

Total drives perf
- 958
- 1823
- 1959
- 1902

4k rand write
4k rand rw 65/35
4k rand read
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
Components

- Linux kernel driver
- RAID management utility

Installation

- Deployed using rpm or deb

Interaction

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

Local client

- Client applications
- File systems

Remote client

- Client applications
- File systems
- Host drivers

Linux block device: /dev/raid

Linux kernel module: Raidix ERA NVMe RAID

Linux block devices

- /dev/nvme0
- /dev/nvme1
- /dev/nvme2
- /dev/nvme3

Local NVMe devices

- PCIe
- U.2
- M.2

Remote devices

- NVMeoF target
- Fibre Channel
- Infiniband
- SPDK target
- Omni Path
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
Our approach to calculation vectorization

- Vector contains bits of different Galois Field elements
- Only packed XORs
- Less data move operations
- Less vector operations
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 dynamic mapping stripes to threads responsible for its handling
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 and latency tests

For single RAID 6:
✓ Up to 4 000 000 IOPS
✓ Latencies < 0.5 ms

<table>
<thead>
<tr>
<th>Block Size (KiB)</th>
<th>Random Read / Write Mix %</th>
<th>Average IOPS (kIOPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0/100</td>
<td>35/65</td>
</tr>
<tr>
<td>4</td>
<td>355</td>
<td>486</td>
</tr>
<tr>
<td>8</td>
<td>180</td>
<td>249</td>
</tr>
<tr>
<td>32</td>
<td>59</td>
<td>83</td>
</tr>
<tr>
<td>128</td>
<td>14</td>
<td>20</td>
</tr>
<tr>
<td>1024</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Average Response Time (ms)

<table>
<thead>
<tr>
<th>Block Size (KiB)</th>
<th>Random Read / Write Mix %</th>
<th>Average Response Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0/100</td>
<td>65/35</td>
</tr>
<tr>
<td>4</td>
<td>0.16</td>
<td>0.13</td>
</tr>
<tr>
<td>8</td>
<td>0.20</td>
<td>0.16</td>
</tr>
<tr>
<td>16</td>
<td>0.31</td>
<td>0.22</td>
</tr>
</tbody>
</table>
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
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