FPGA Implementation of Erasure Codes in NVMe based JBOFs

Manoj Roge

Director, Data Center

Xilinx Inc.

Acknowledgement

- Shre Shah – Data Center Architect, Xilinx

Agenda

- Storage trends in Data Center
- RAID and its challenges
- Erasure coding to the rescue
- FPGA implementation of Erasure codes
- Summary

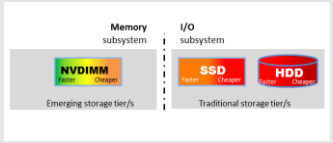
Storage Trends in Data Center

Big Data and Scale-out Storage

NVMe and Object based storage, NVMe over Fabric

Storage Intelligence and Software Defined Storage

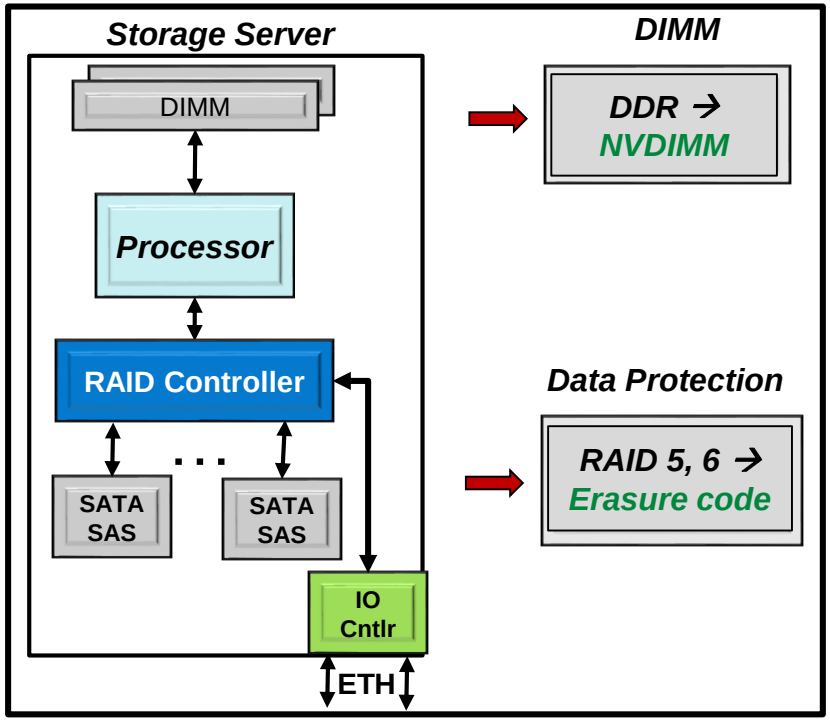
Analytics in storage arrays



Storage Class Memory

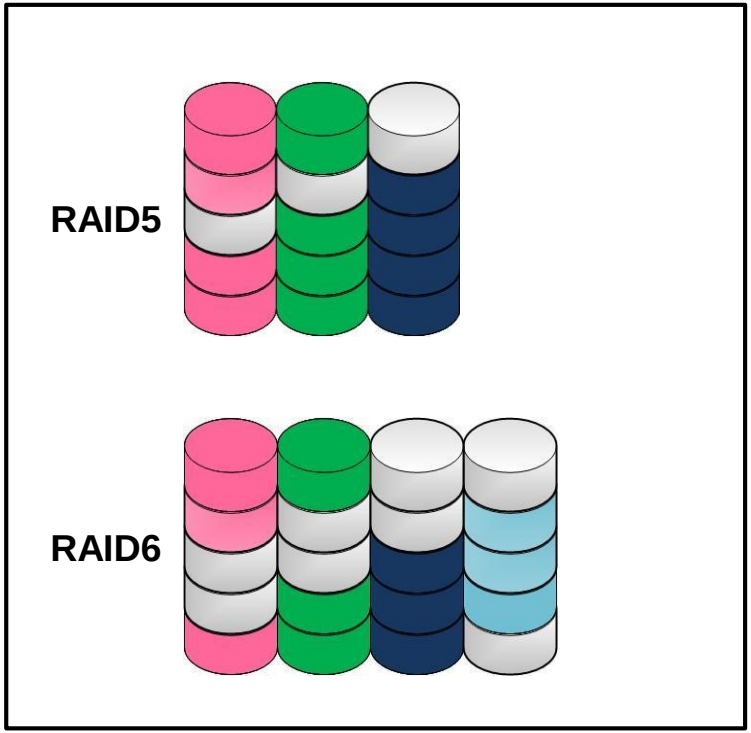
NVDIMMs

Challenges with Traditional Architecture



- SATA/SAS HDDs
  - Mechanical drives → Failure rate is higher than SSDs
- Data Protection: RAID 5, 6
  - 40+% Overhead in storage capacity
  - Write and Read performance challenges
  - Very high re-build times
  - CPU intensive:
    - ASIC/ASSP : RAID5 & 6 Algorithms

RAID5, RAID6 Data Protection

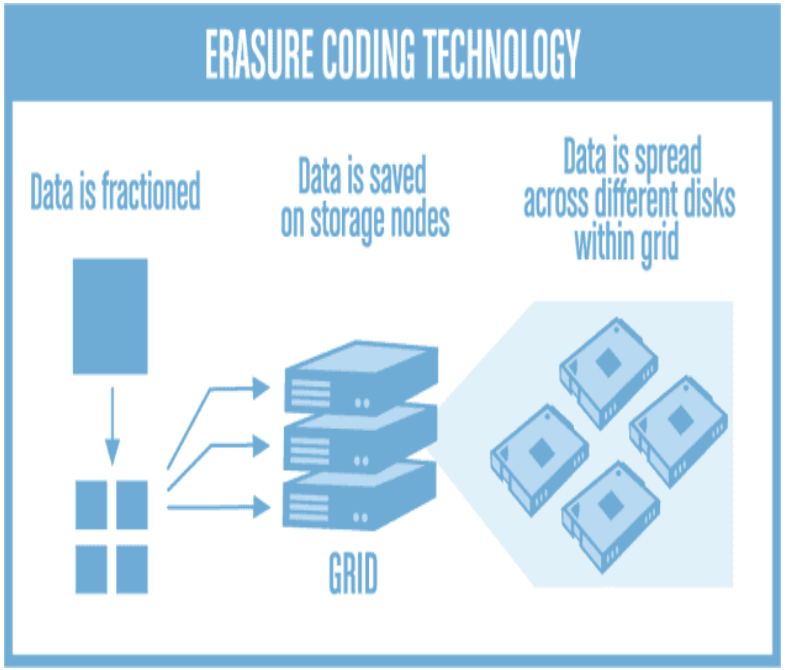


- RAID5:
  - Need minimum 3 drives to implement
  - Data striping with parity
  - Can’t survive 2 drive failures
- RAID6:
  - Need minimum 4 drives to implement
  - Data striping with double parity: Provides more robust data protection
  - Can replace any 2 drive failures

Challenges with RAID system

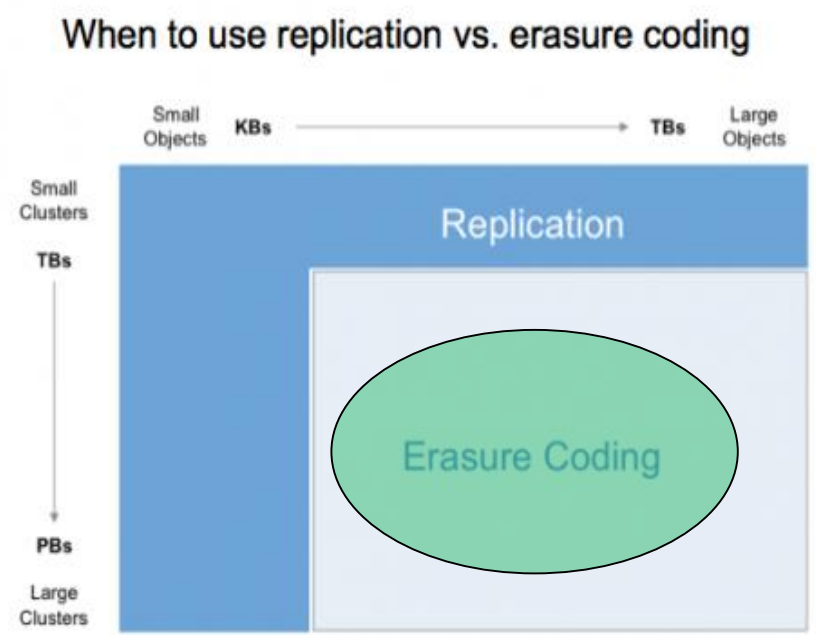
- Storage capacity increasing exponentially
- Unacceptable re-build time → Loss of data
- Continued cost reduction
- Need new method to protect data

Erasure Code Overview



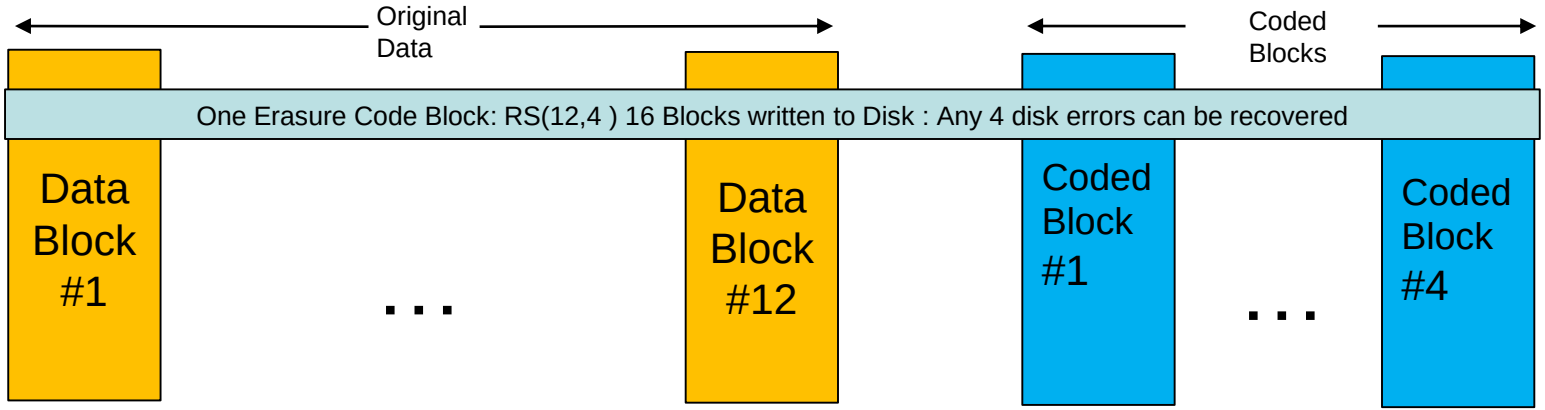
- Data striped: Small chunks
- Reed Solomon Codes
  - FEC calculated
- DATA + Parity checks spread across multiple storage devices
- Rebuild: Read D + P
  - Reconstruct D: P when disks fail

Erasure Coding to the Rescue



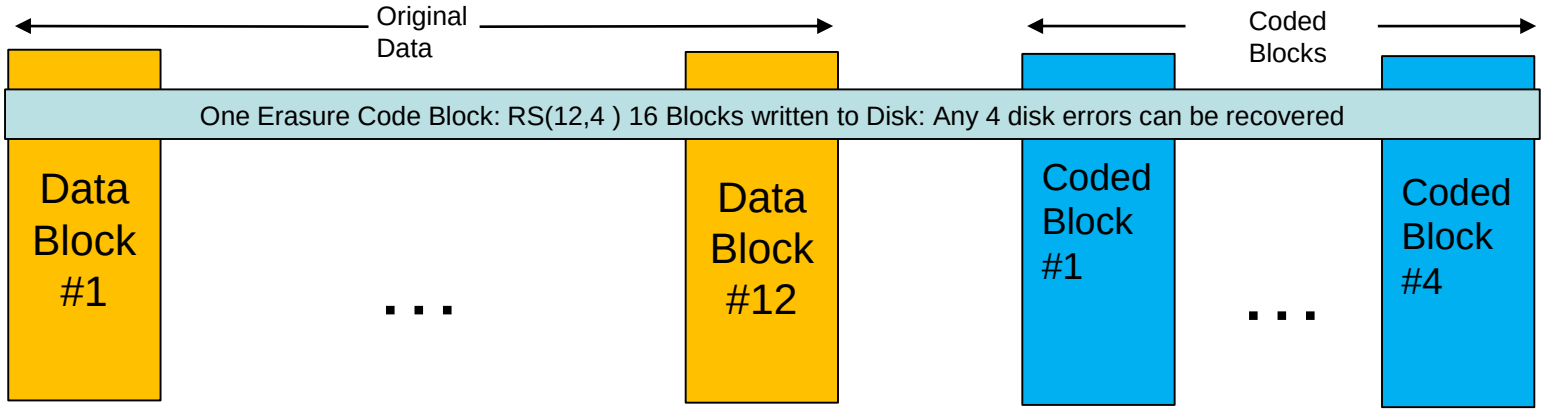
- Higher storage efficiency
- Fast re-build with FPGA
  - No loss of data
- Software stack available
  - Ceph, Hadoop

FPGA Implementation of Erasure Codes



- Assume Erasure code RS(12,4)
  - 12 Data blocks and 4 parity blocks, block size: 1KB – 128MB
  - {Data, Parity} = G (G is a generator matrix) * D (D is Data Matrix)

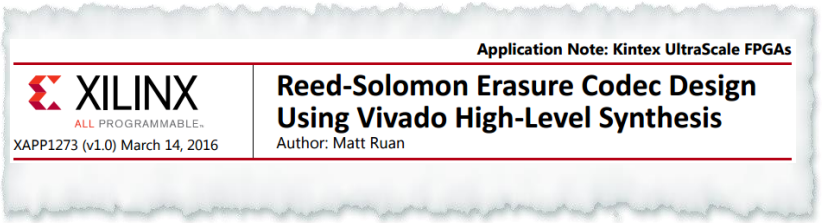
FPGA Implementation of Erasure codes



- Data Recovery : Erasure code RS(12,4)
  - 12 Data blocks and 4 parity coded blocks, Assume 4 failures
  - {Erased Data} = Erased Rows * G (Survived Rows) * Survived Data
  - Original Data = {Data- Erased Data, Erased Data}

Xilinx Implementation of Erasure Codes

- Implemented in C/C++ with directives
  - Easy to maintain and customize
  - Compiled with HLS tools
- App Note and Source code Available
  - Code runs at > 300 MHz
  - Encoder/Decoder : 28.8Gbps (Data), 9.6 Gbps (parity)
  - Multiple instances : Endec to achieve higher rate : 100Gbps
  - Latency : < 90 ns
- Performance:
  - Over 10x higher performance compared to CPU


Application Note: Kintex UltraScale FPGAs

XILINX
ALL PROGRAMMABLE.

XAPP1273 (v1.0) March 14, 2016

Reed-Solomon Erasure Codec Design Using Vivado High-Level Synthesis

Author: Matt Ruan

Summary

- Erasure codes solve RAID inefficiencies
  - Less overhead to save storage (e.g. RS(30,3) is 10% overhead)
  - Faster rebuild time
  - CPU offloaded from parity calculation
- FPGAs are good fit for Erasure code implementation
- Xilinx solutions for emerging storage architectures
  - Scale-out storage, Database acceleration, Storage class memories
  - Please visit Xilinx Booth 721 to check out solutions listed above

Follow Xilinx

Thank you!

facebook.com/XilinxInc

twitter.com/#!/XilinxInc

youtube.com/XilinxInc

linkedin.com/company/xilinx

plus.google.com/+Xilinx

xilinx.com/about/app-download.html