SNIA PERSISTENT MEMORY + SUMMIT 2022 COMPUTATIONAL STORAGE

VIRTUAL EVENT • MAY 24-25, 2022

File System Acceleration Using Computational Storage For Efficient Data Storage

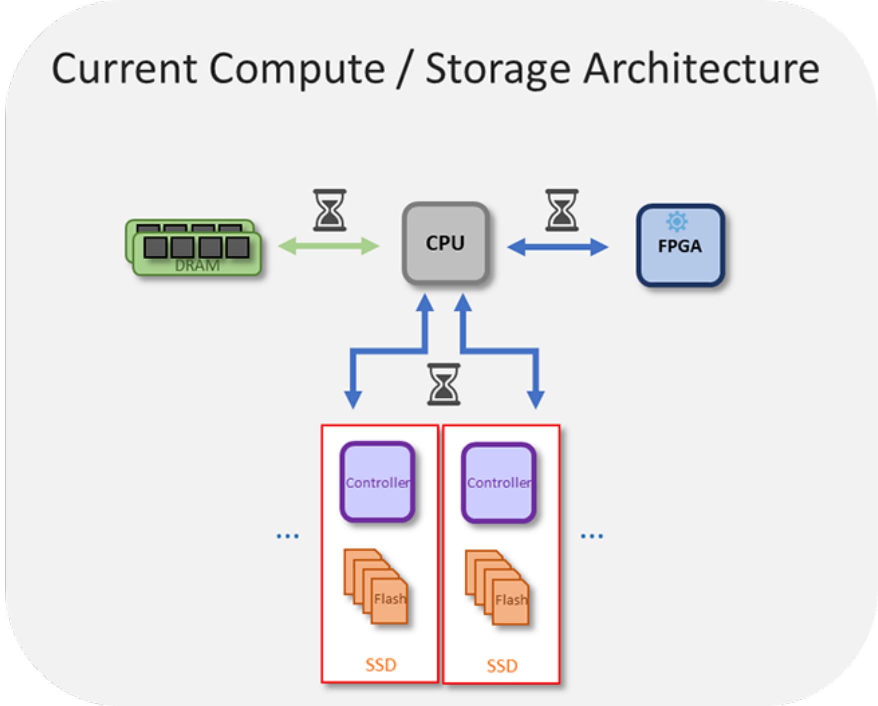
Vaishnavi S G - Software Developer - AMD
Srija Malyala - Software Developer - AMD
Amar Nagula - Senior Software Development Manager - AMD

Agenda

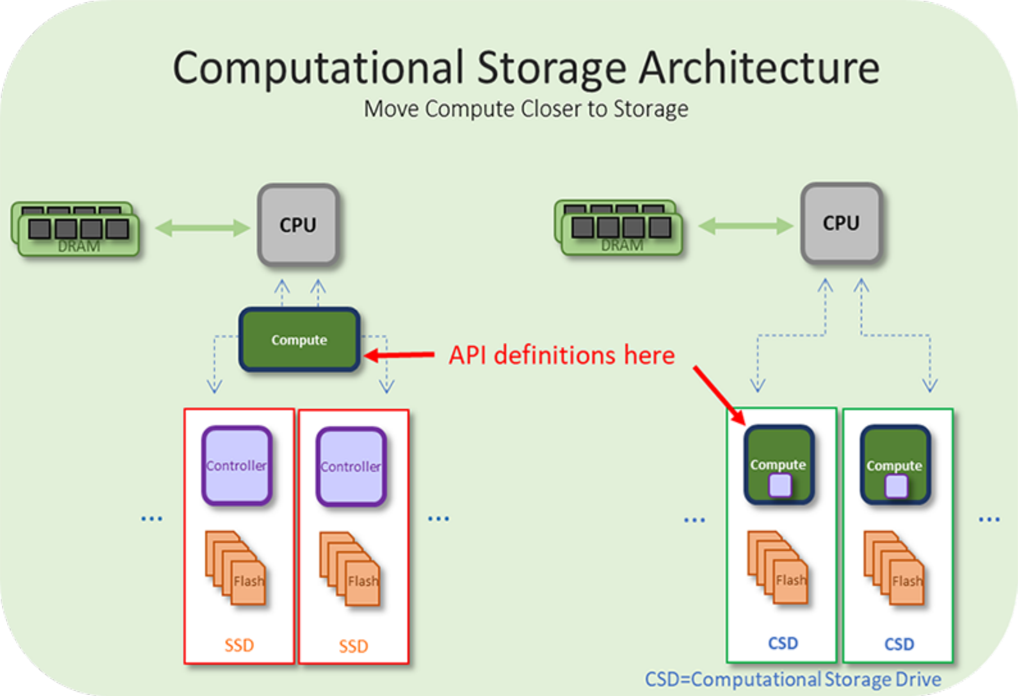
- Overview
  - What is computational storage?
  - What is a SmartSSD?
  - What is ZFS?
  - ZFS with SmartSSD

- Implementation
  - Offload design and flow
  - Results
  - Future scope

Computational Storage - Overview



Source : SNIA



Source : SNIA

- As the data grows exponentially, CPU centric processing becomes a bottleneck.

- The solution is to move compute to where data resides.

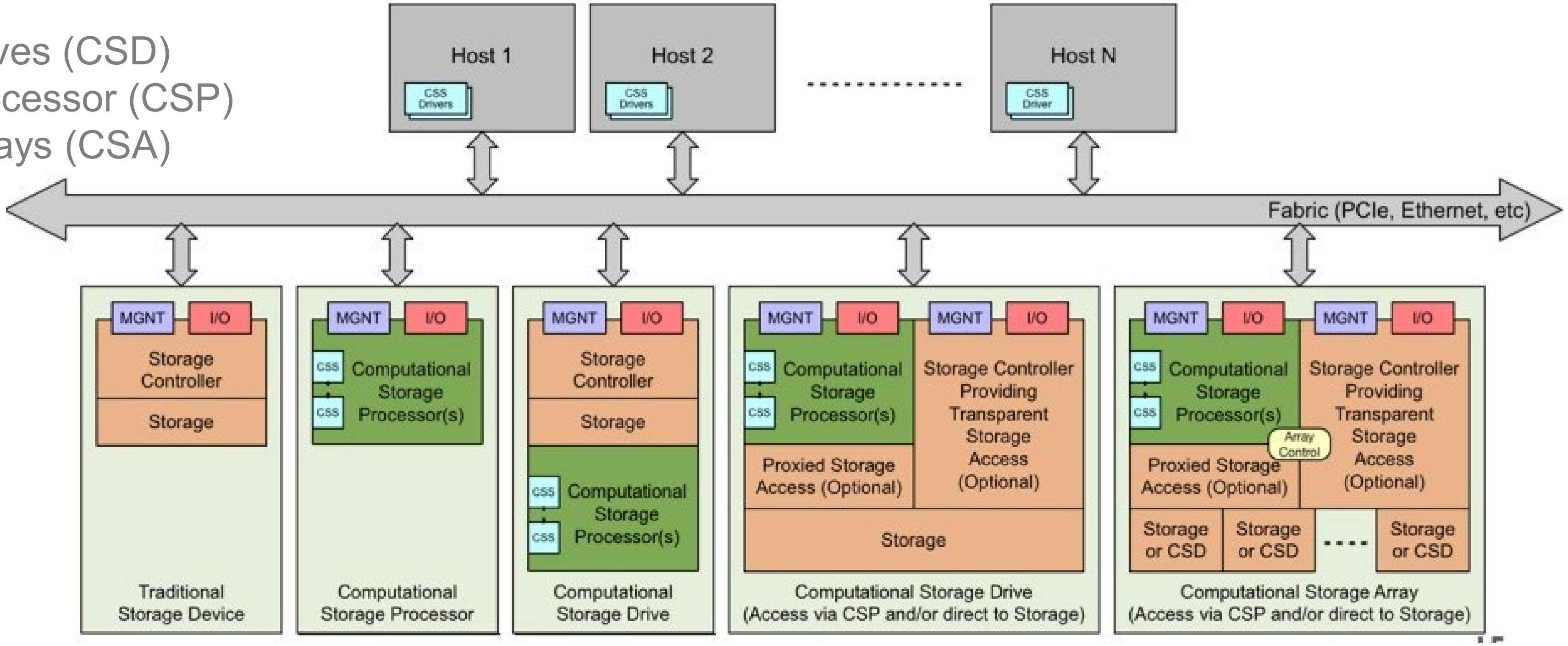
Computational Storage - Overview

- SNIA definition

“Computational Storage is defined as architectures that provide Computational Storage Functions (CSF) coupled to storage, offloading host processing or reducing data movement.”

- Types
  - Computational Storage Drives (CSD)
  - Computational Storage Processor (CSP)
  - Computational Storage Arrays (CSA)

- Protocols
  - OpenCL
  - NVMe
  - TCP over NVMe
  - More..

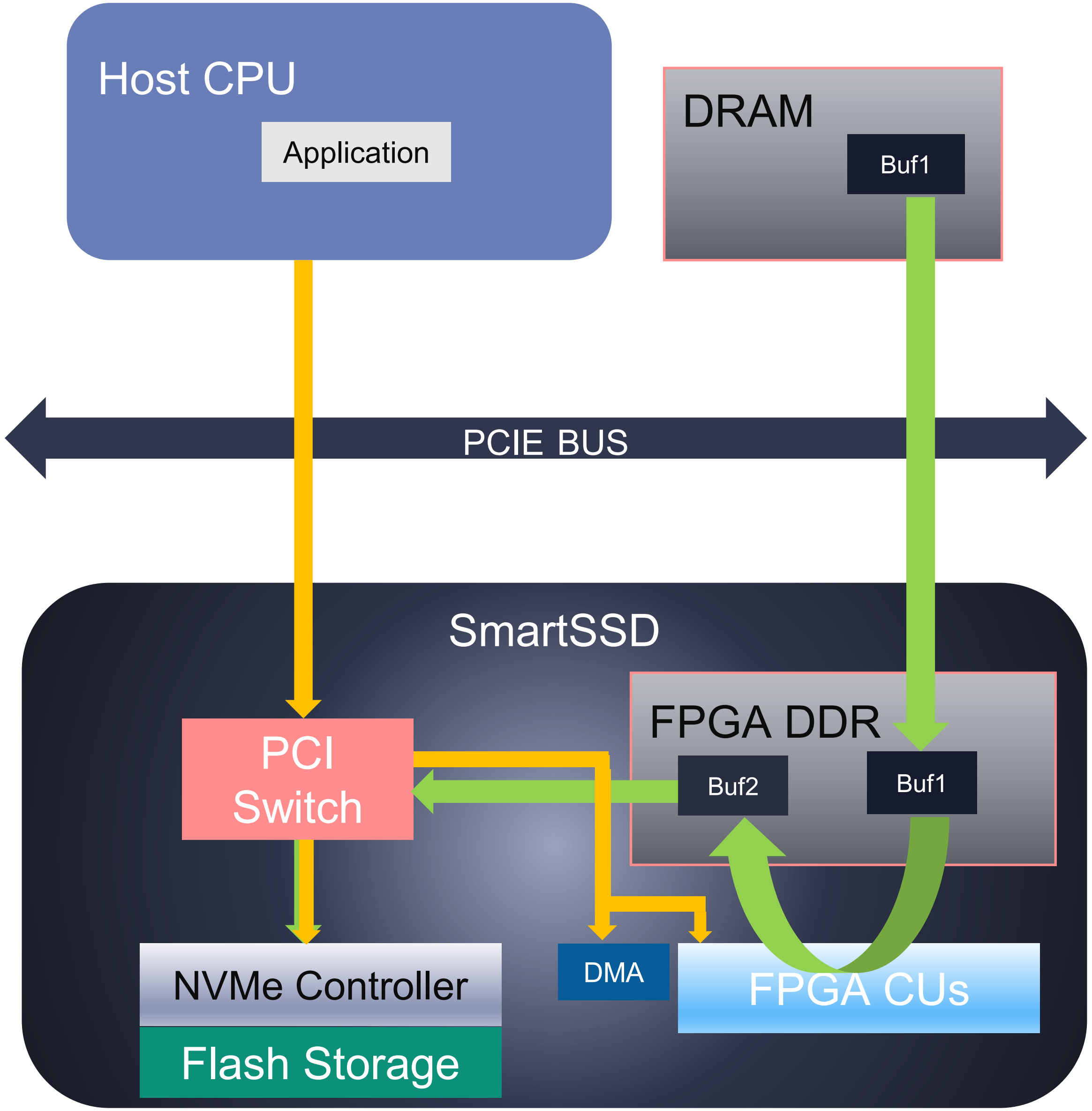


Host 1
CSS Drivers
Host 2
CSS Drivers
Host N
CSS Driver
Fabric (PCIe, Ethernet, etc)
MGNT
I/O
Storage Controller
Storage
Traditional Storage Device
CSS
Computational Storage Processor(s)
Computational Storage Processor
Computational Storage Drive
Storage Controller Providing Transparent Storage Access (Optional)
Proxied Storage Access (Optional)
Computational Storage Drive (Access via CSP and/or direct to Storage)
Array Control
Storage or CSD
Computational Storage Array (Access via CSP and/or direct to Storage)

Source : SNIA

What is SmartSSD?

- Computational Storage Drive based on OpenCL API.
- Standard NVMe interface for storage access.
- Computational storage brings FPGA and SSD together and enables the direct data transfer (DMA) from SSD to FPGA DDR avoiding the host RAM altogether.
- Peer-to-Peer (P2P) Data Transfer
  - Read
    - SSD ⇒ FPGA DDR ⇒ Compute ⇒ FPGA DDR ⇒ Host Memory
  - Write
    - Host Memory ⇒ FPGA DDR ⇒ Compute ⇒ FPGA DDR ⇒ SSD



ZFS – Overview

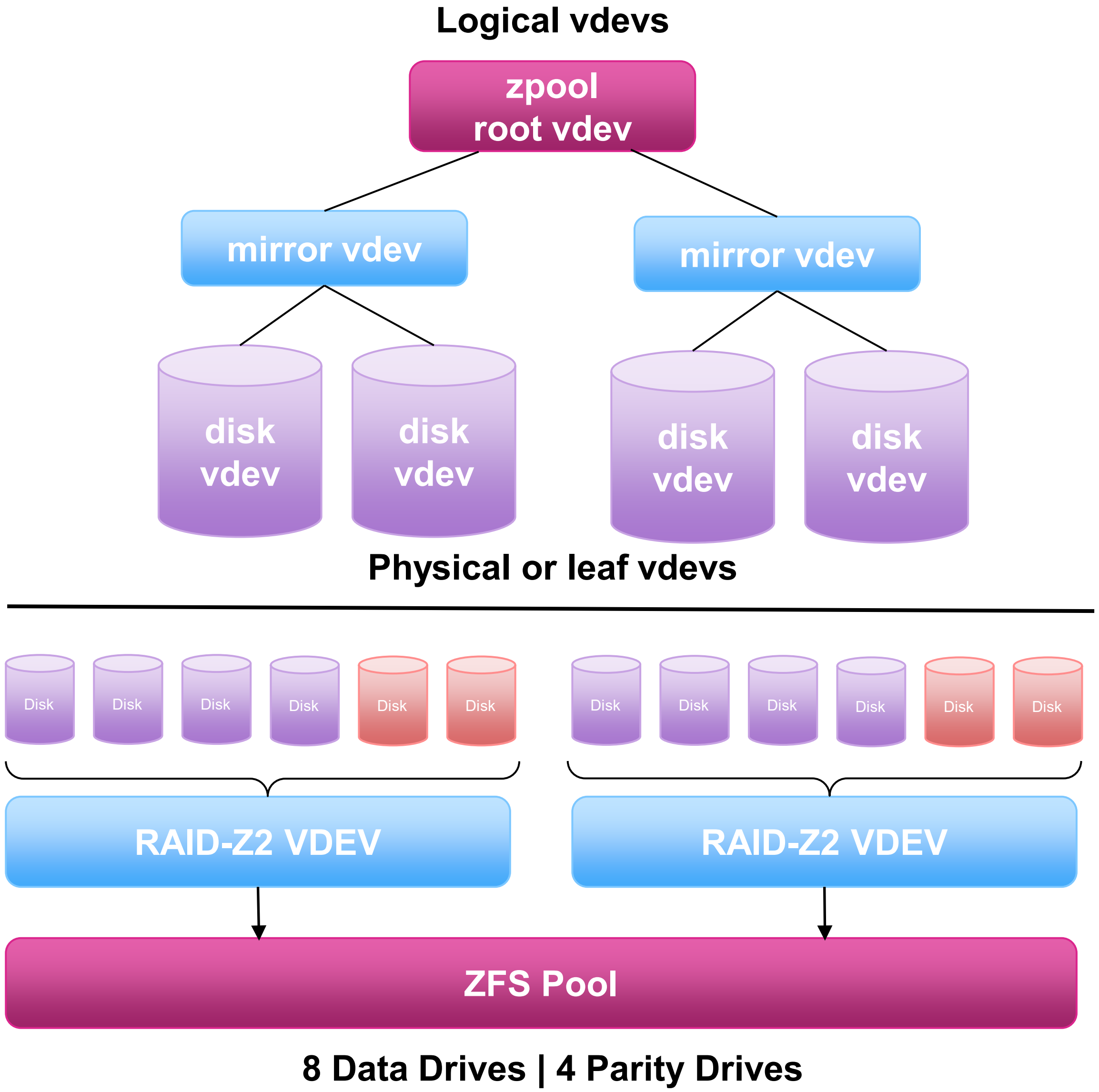
- Zettabyte File System – Next generation advanced file system.
- Designed with scalability, reliability and performance in mind, combines file system with volume manager functionality.
- Many interesting features :
  - Pooled storage
  - Copy-on-write
  - Compression
  - Snapshots
  - Data integrity verification and automatic repair
  - Redundancy with mirroring
  - RAID-Z
  - Deduplication

ZFS - Overview

- ZFS uses storage pools to combine multiple disks in various configurations (striped, mirrored, RAID-Z, RAID-Z2).

- ZFS uses layered ZIO over virtual devs with pipelines in each layer. The vdevs at the bottom are the physical disks.

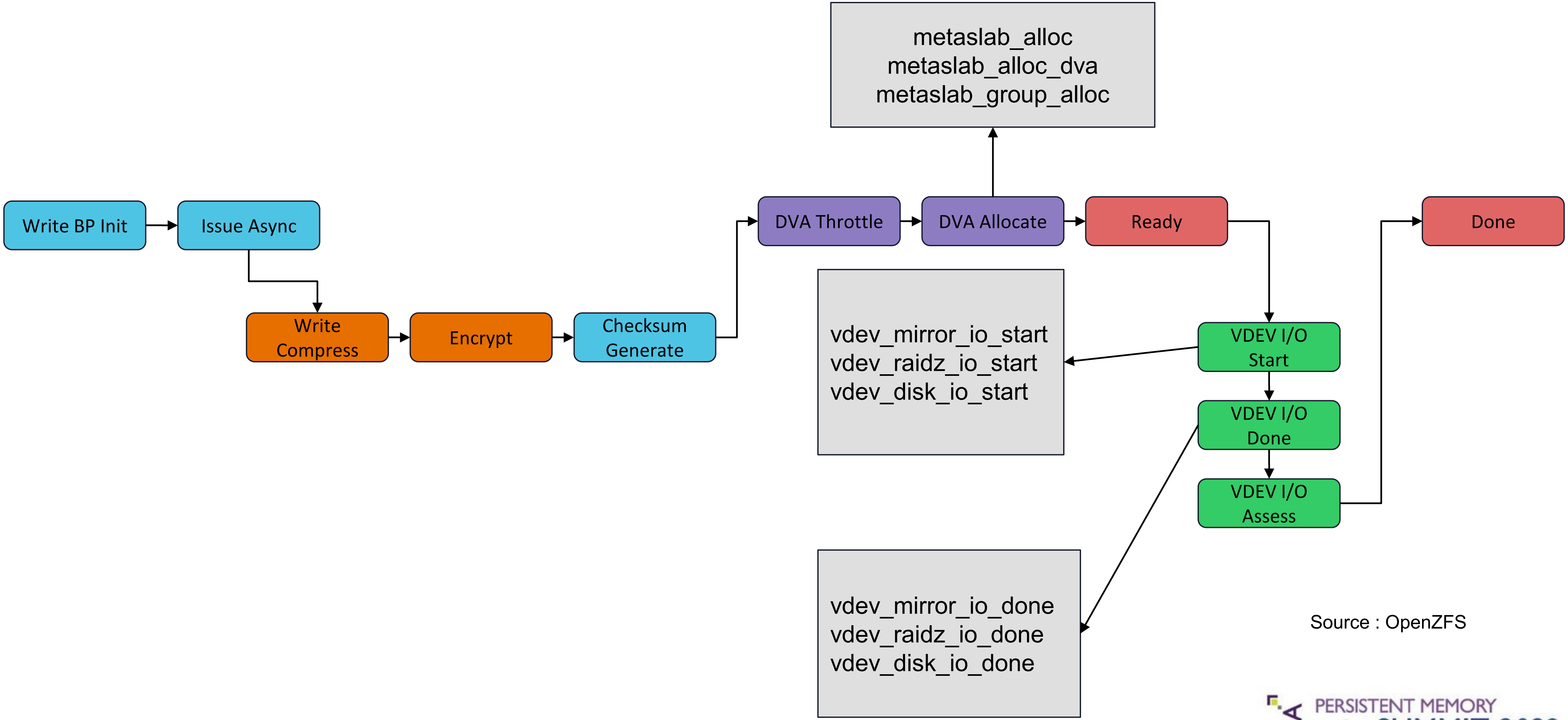
- Compression is triggered in the higher level vdevs for the ZIO.



Advantages of offloading Compression at the File System

- Offloading at the application level
  - Can achieve maximum efficacy as it can control the buffer size.
  - But, requires each application to be modified separately.
- Offloading at the block layer in the Linux stack
  - Can provide transparent compression to applications without having them to modify.
  - But a smaller buffer (block size, which is typically 4KB) available at the block layer renders the CR to be less than desirable. Also, creates the overhead issue of triggering hardware offload for smaller blocks and can bring down overall CPU efficiency.
- Offloading at File system level
  - Can achieve balance between buffer size (ZFS default block size is 128kB) and offload overhead.

ZFS – Write Pipeline



Source : OpenZFS

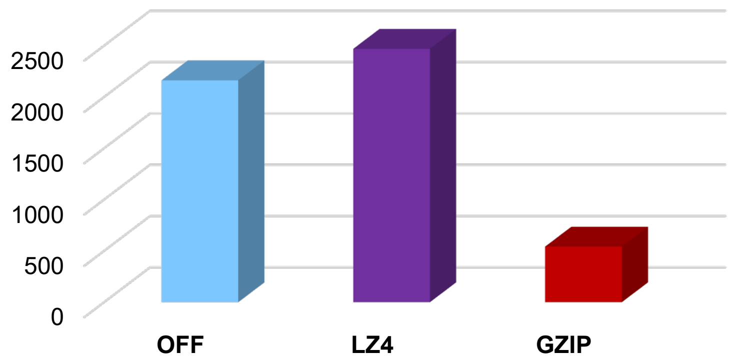
ZFS – Compression

- LZ4 vs GZIP
  - GZIP is a lossless compression algorithm with better CR than LZ4 but at the expense of CPU utilization. Hence GZIP is ideal for offload.

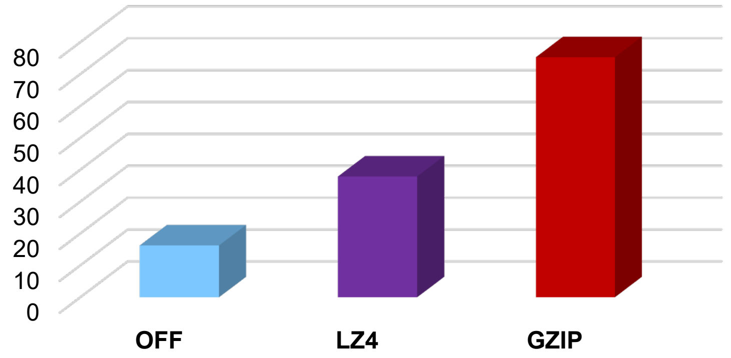
Algorithm	Compression Ratio
LZ4	2.05
GZIP	3.05

- GZIP Compression on CPU (default)
  - Less throughput
  - High CPU resource utilization
- By Offloading GZIP to CSD
  - Reduce CPU utilization
  - Improve performance and efficiency

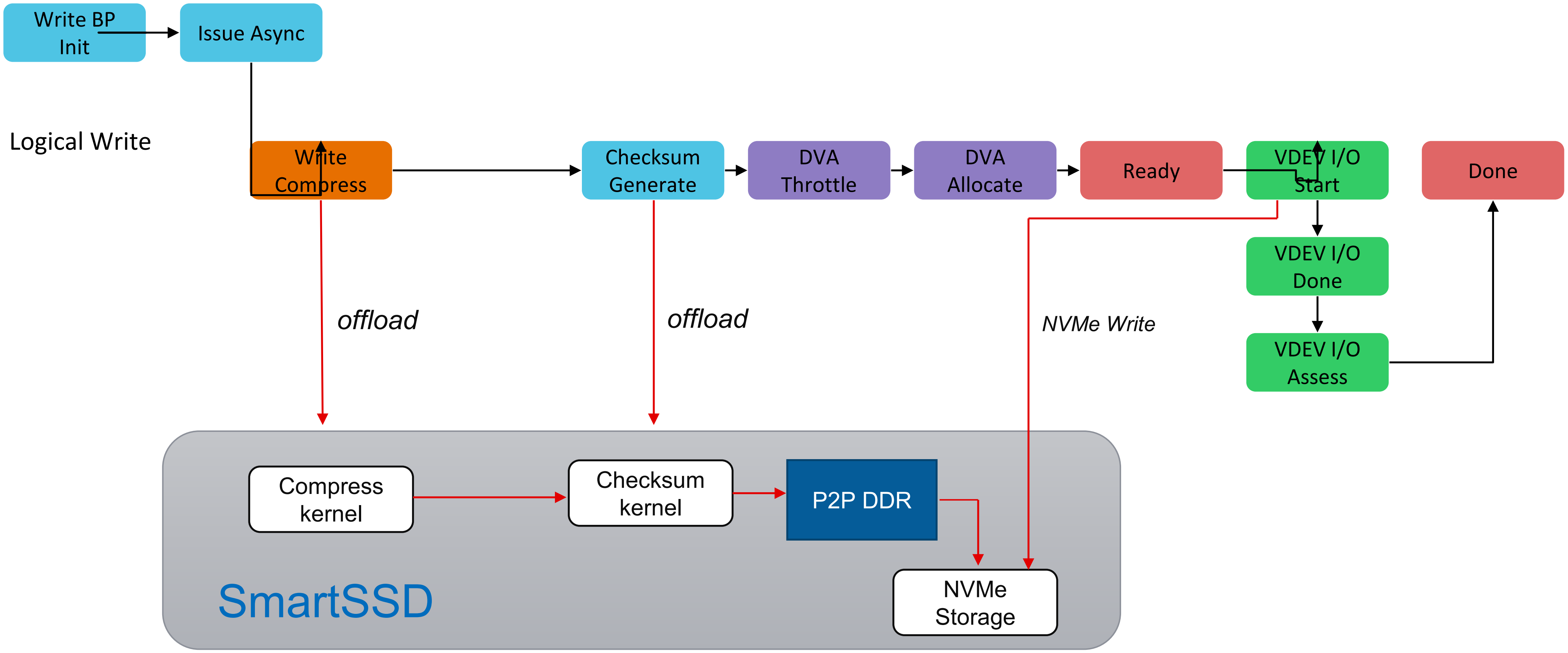
(a) Throughput (MB/s)



(b) CPU Utilization %

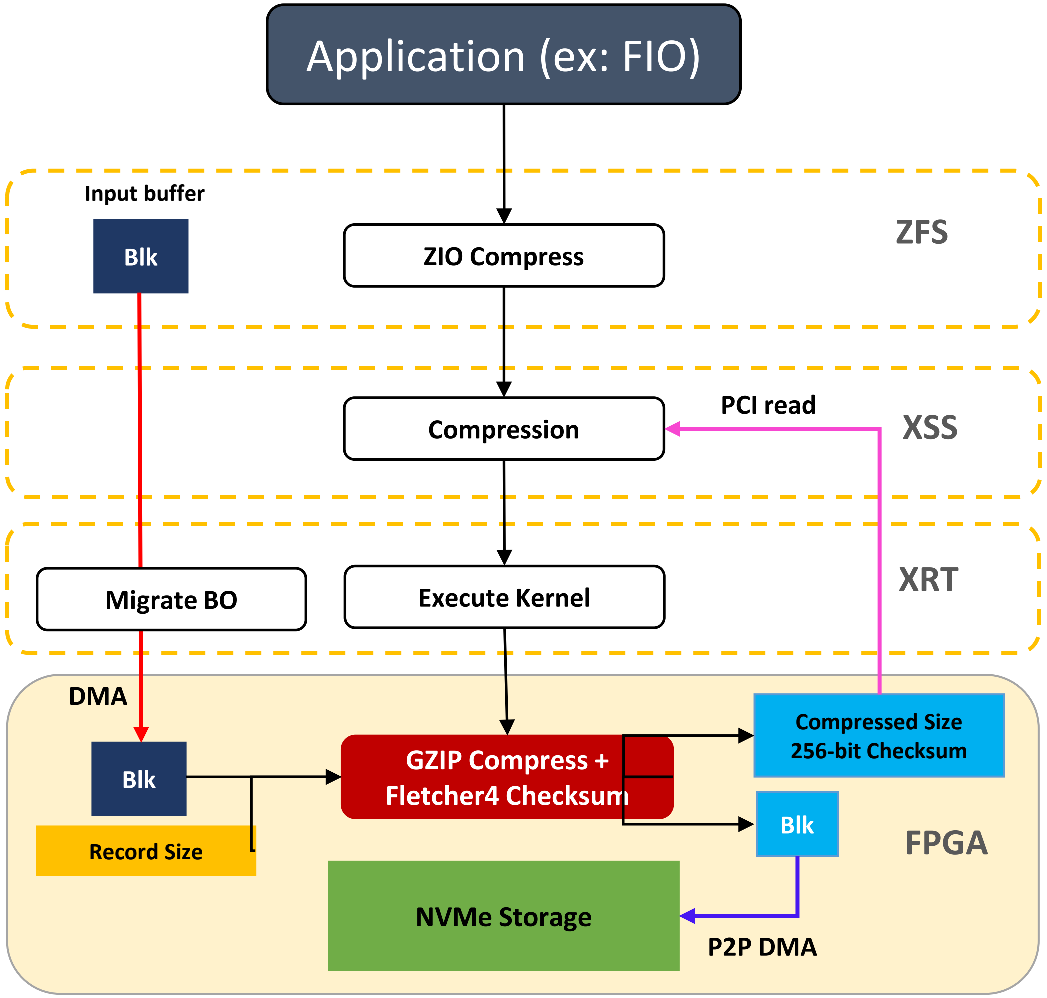


Write Pipeline with offload



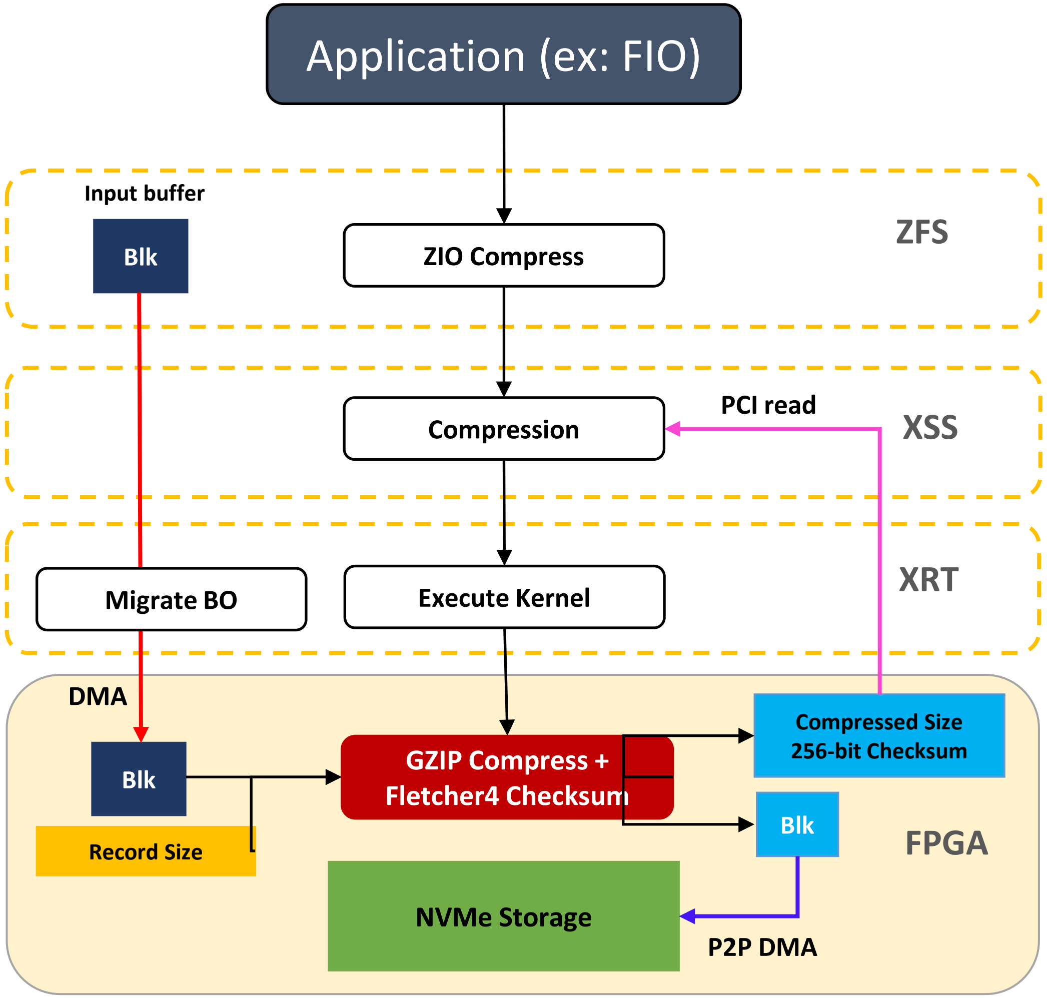
ZFS on SmartSSD

- ZFS is modified to send compress requests to XSS - a kernel space library that provides higher level storage acceleration APIs to applications.
- XSS uses XRT - a runtime library that communicates with FPGA.
- XSS moves input data to FPGA DDR, triggers compression and checksum calculation.
- The compressed data from FPGA DDR is moved to SSD(P2P) during the disk *write* call of ZFS.

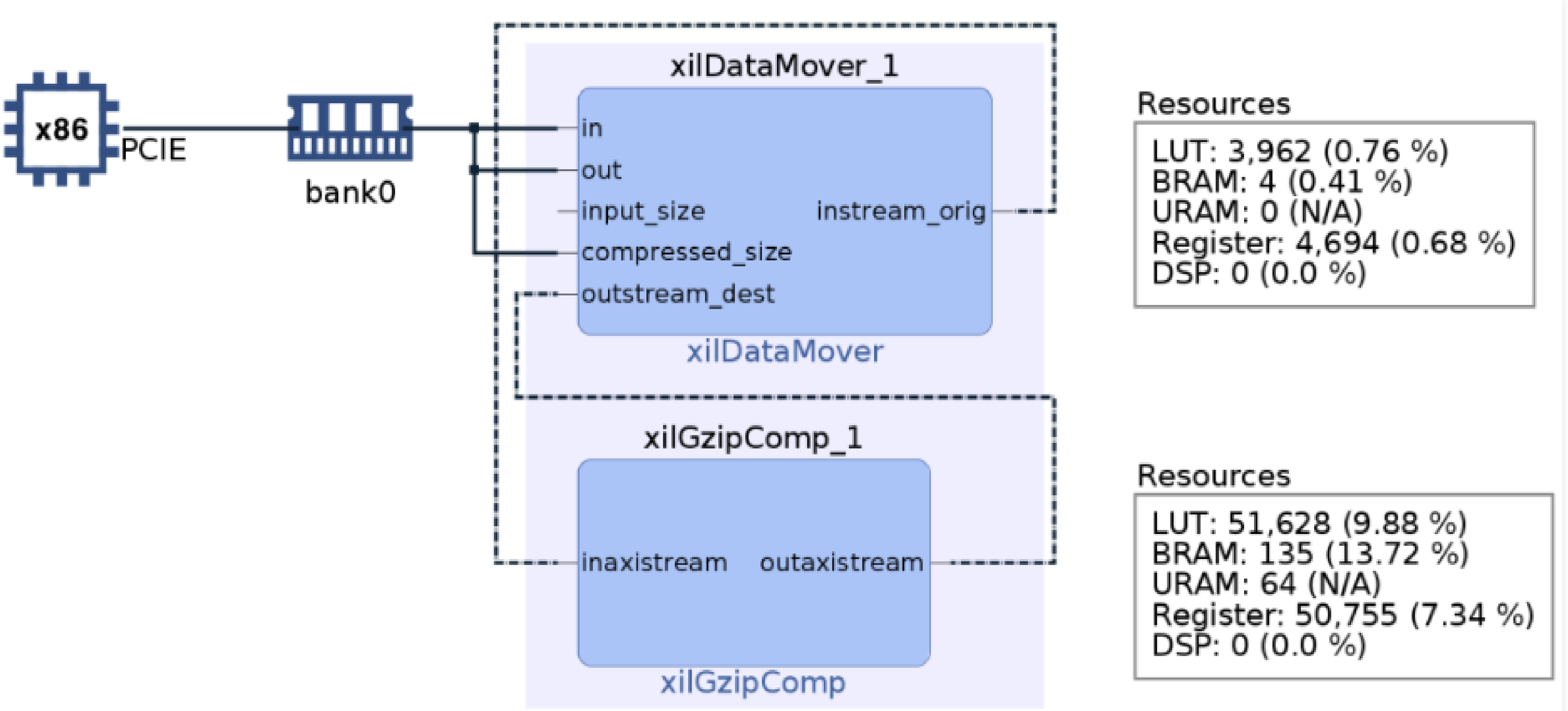


ZFS on SmartSSD

- The default Fletcher-4 Checksum is also offloaded to avoid data transfer to the host from FPGA DDR after compression.
- To support ZFS ARC (Adaptive Replacement Cache), which is enabled in ZFS by default, compressed data is moved to host DRAM cache through DMA.



GZIP HLS Kernel



Resources

LUT: 3,962 (0.76 %)
BRAM: 4 (0.41 %)
URAM: 0 (N/A)
Register: 4,694 (0.68 %)
DSP: 0 (0.0 %)

Resources

LUT: 51,628 (9.88 %)
BRAM: 135 (13.72 %)
URAM: 64 (N/A)
Register: 50,755 (7.34 %)
DSP: 0 (0.0 %)

Multicore GZIP Compress Streaming kernel attached to a Data Mover kernel

GZIP HLS Kernel

- The GZIP compression and Fletcher-4 checksum kernels are developed in High-Level Synthesis (HLS) language using the Vitis software development platform.
- The data mover kernel reads the source buffer from FPGA DDR and writes to the input stream of the GZIP compress kernel.
- Similarly, the data mover kernel reads the compressed data from the output stream of GZIP compress kernel and writes to the P2P destination buffer present in FPGA DDR.
- There are 8 parallel engines of LZ77, Huffman Encoder, etc. inside the GZIP compress kernel which run simultaneously to improve performance.
- These kernels are designed in such a way that they can handle multiple compress requests at once in pipelined fashion using a chaining interface.

Validation - Test Configuration

Server Details:

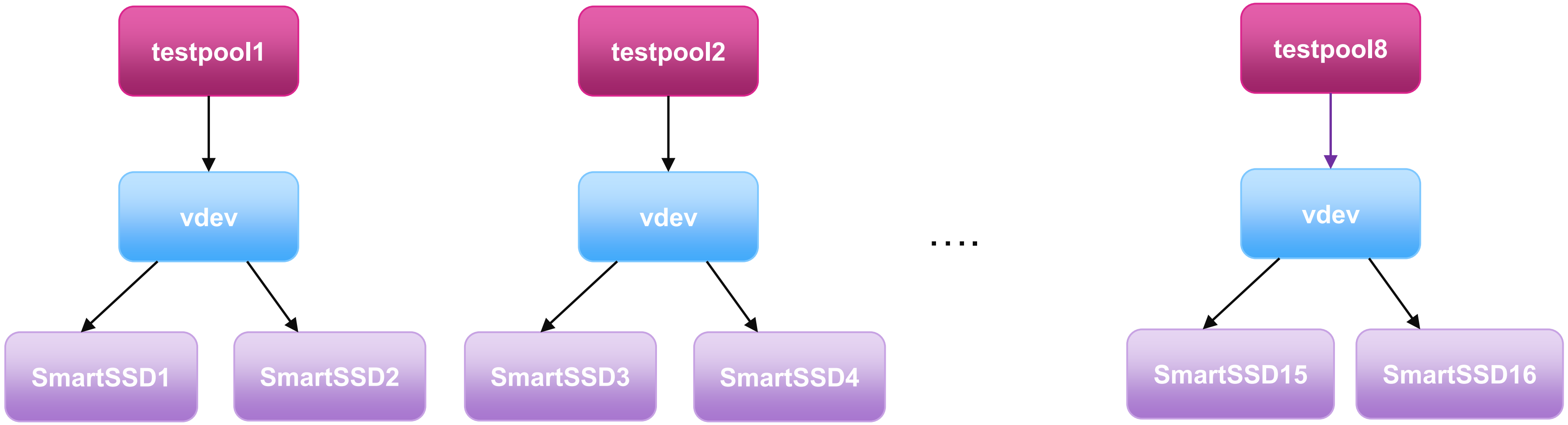
Dell 7525 AMD EPYC 7282, 2 socket
64 CPU, Freq: 2187MHz

OS: Ubuntu 16.04.06

Linux kernel: 4.15.0-142-generic

Multiple pools and volumes

```
$zpool create testpool1 mirror /dev/nvme0n1 /dev/nvme1n1
$zpool create testpool2 mirror /dev/nvme2n1 /dev/nvme3n1 ……
$zfs create testpool1/vol1
$zfs create testpool2/vol1
…
```



- FIO is run on all the pools simultaneously and cumulative IOPs are noted down

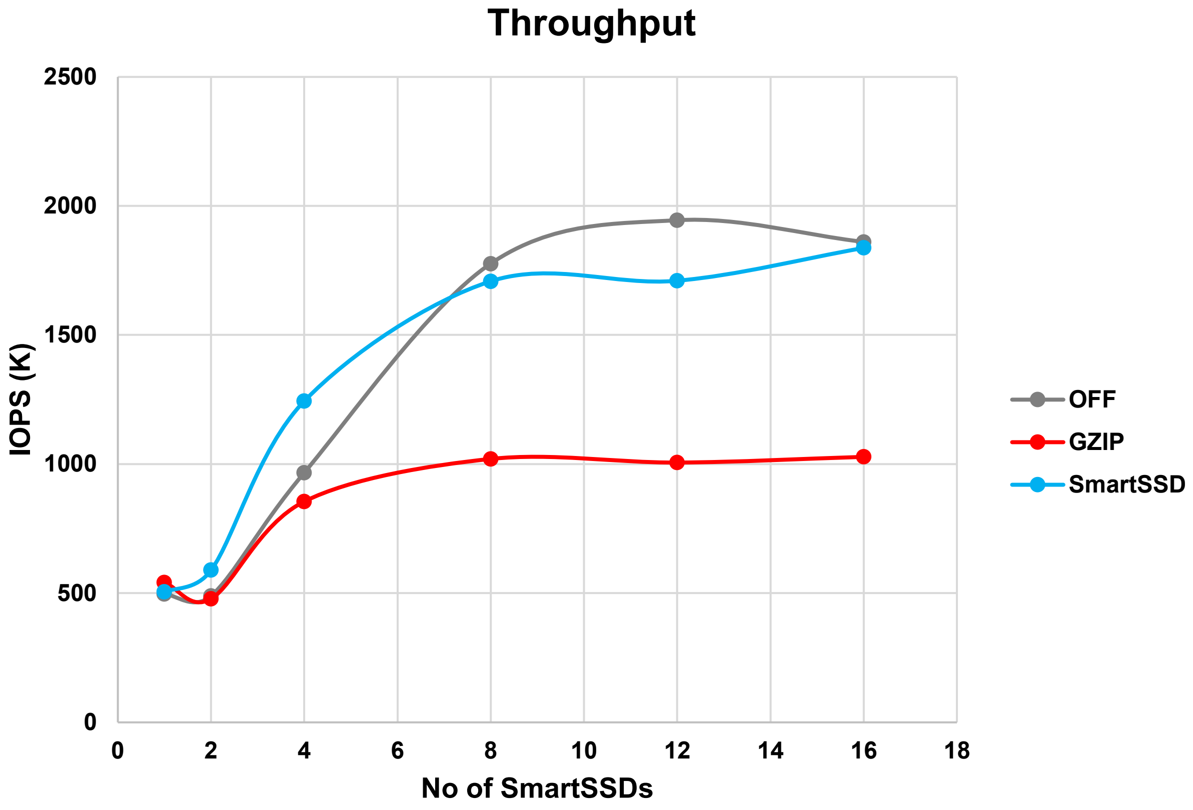
```
$fio --name=test --ioengine=libaio --iodepth=32 --rw=write --bs=4k --direct=1 --size=20G --numjobs=12 --group_reporting --buffer_compress_percentage=75 --directory=/testpool1/vol1/
```

How much of advantage?

Throughput Improvement

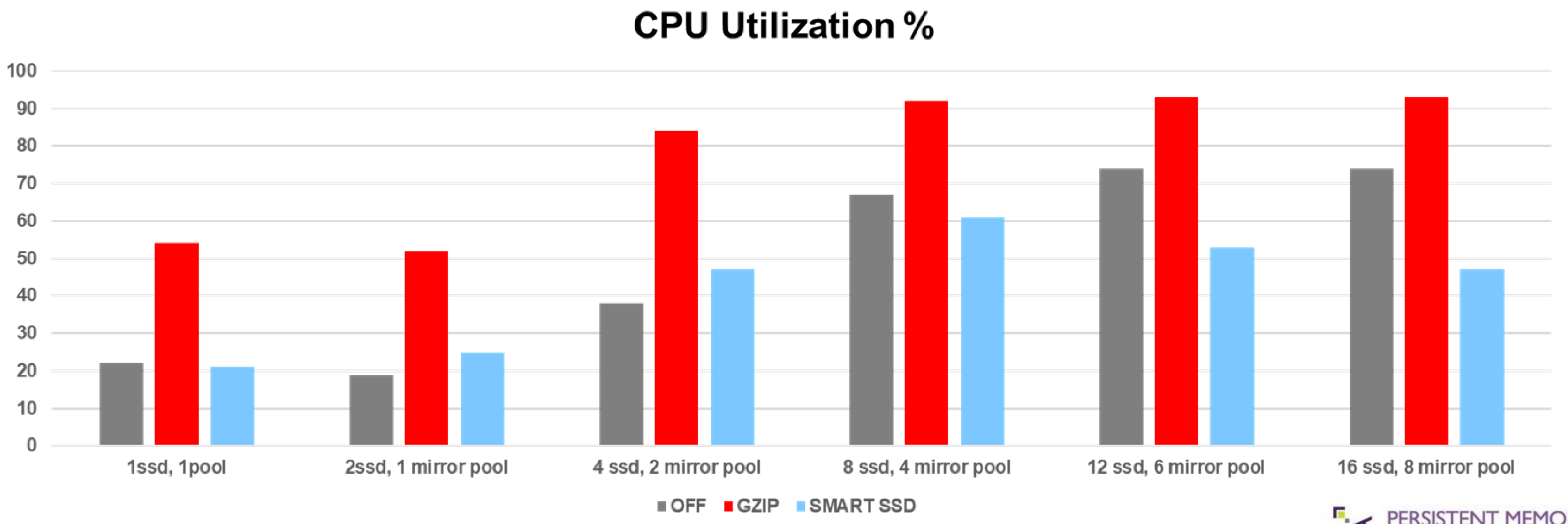
- Industry standard benchmark IO tool FIO is run on the ZFS volume and the combined IOPS is recorded.
- CPU vs SmartSSD
  - ***2X improvement***

Mode	Compression Ratio
OFF	1
SW GZIP	3.81
HW OFFLOAD	3.76



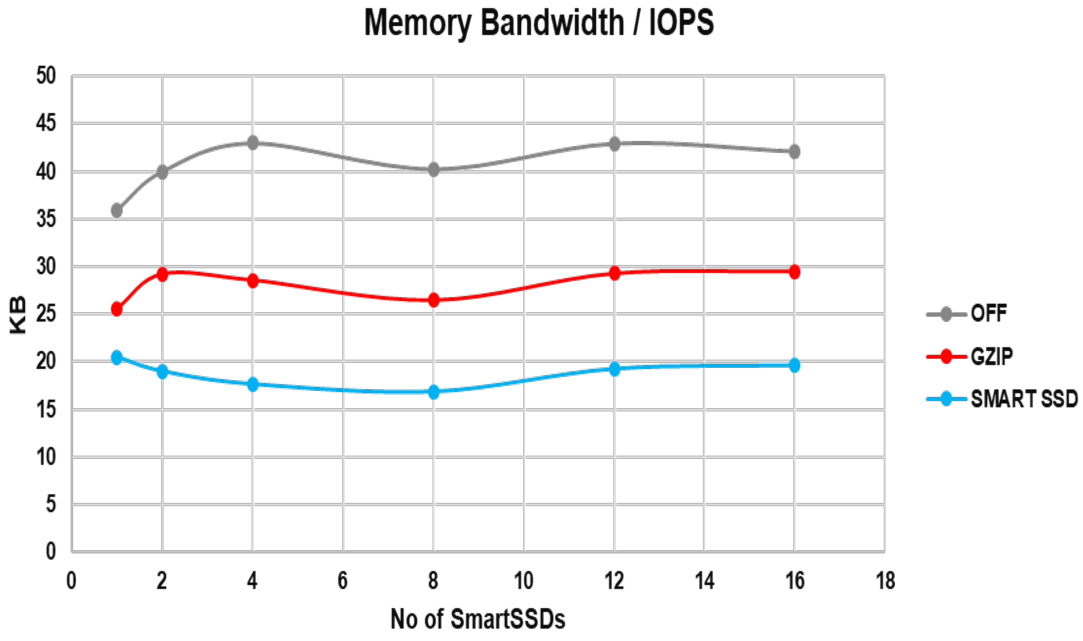
CPU Resource Utilization Reduction

- While the FIO is being run, CPU utilization is recorded using Linux mpstat.
- CPU vs SmartSSD
  - ***Around 55 % decrease in CPU resource utilization***



System Memory Efficiency Improvement

- While the FIO is being run, Memory Bandwidth for the system is recorded using AMD uProf tool.
- Memory bandwidth increases in proportion to increased IOPS.
- Memory BW to IOPs ratio is lowest for the offload case even with the high IOPS. This is the advantage of Computational Storage Drives like SmartSSD.



Future scope

- GZIP kernel with higher compression levels
  - Field replaceable due to FPGA nature
- Offload decompression
- Support other checksum algorithms like SHA 256
- Offload other features like RAIDZ, deduplication
- Integrate with application-level acceleration functions

SNIA PERSISTENT MEMORY + COMPUTATIONAL STORAGE SUMMIT 2022

VIRTUAL EVENT • MAY 24-25, 2022

Please take a moment to rate this session.

Your feedback is important to us.