SDC 18
September 24-27, 2018
Santa Clara, CA
www.storagedeveloper.org

FPGA-Based ZLIB/GZIP Compression as an NVMe Namespace

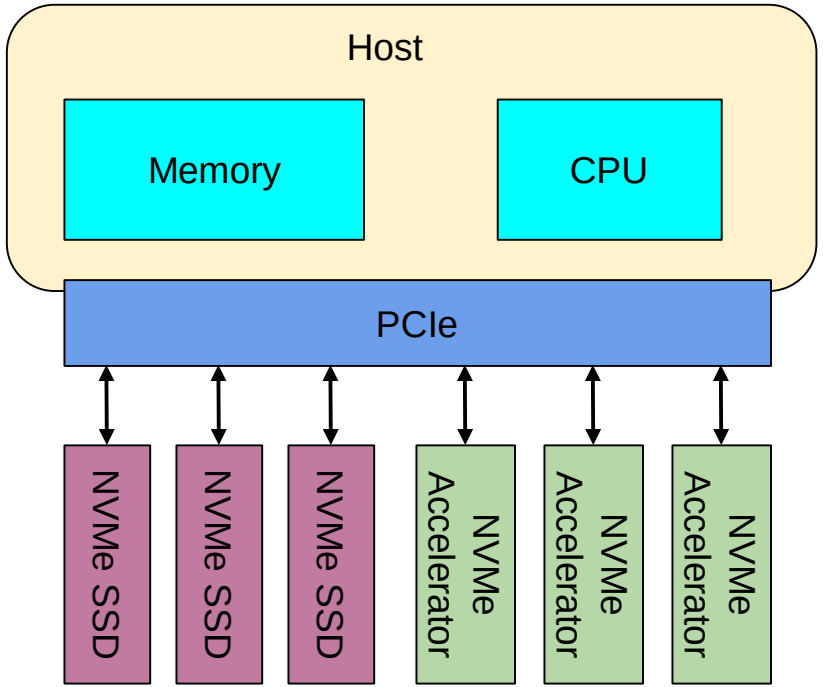
Saeed Fouladi Fard
Eideticom

Why NVMe

- NVMe: A standard specification for accessing non-volatile media over PCIe
- High-speed and CPU efficient
- In-box drivers available for major OSes
- Allows peer-to-peer data transfers
- Reduces system memory access
- Frees CPU time

Why NVMe, Cont'd

- NVMe can be used as a high speed platform for using and sharing accelerators with low overhead
- Easy to use Accels

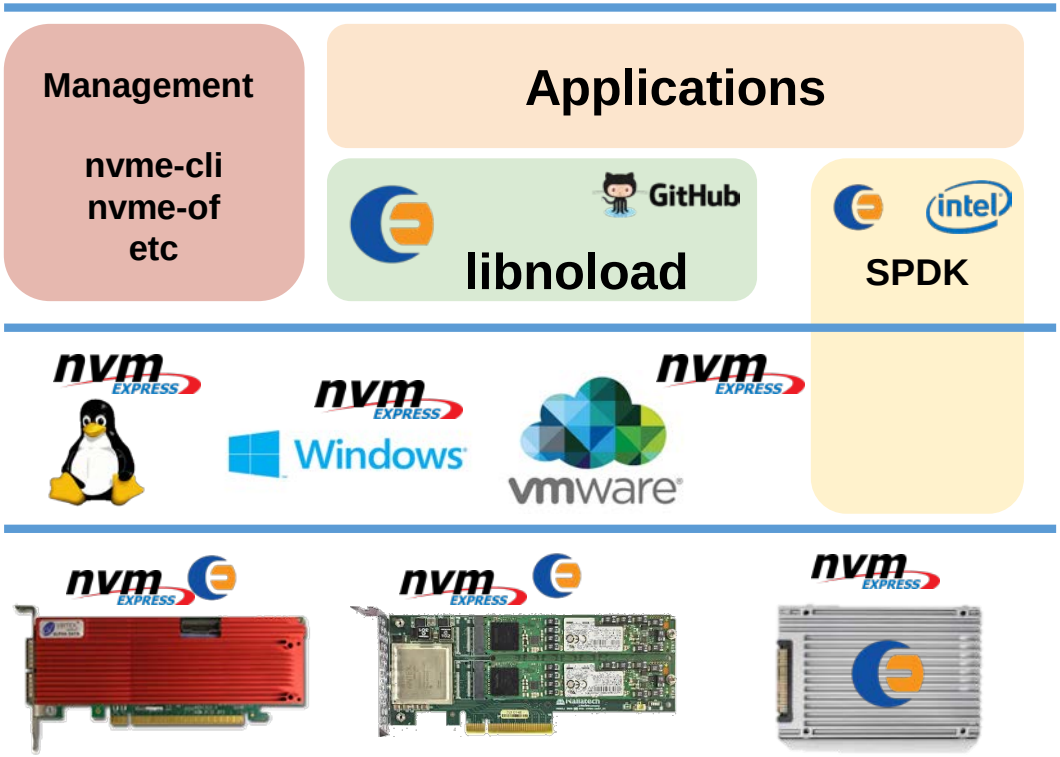


Hardware Platform

- Called **NoLoad™** = NVMe Offload
- NoLoad™ can present FPGA accelerators as NVMe namespaces to the host computer or peers
- Accelerator integration, verification, and discovery is mostly automated
- Host software can be added to use the accelerator

Streamlined Accelerator Integration

NoLoad™ Software



- **Userspace:** both kernel & userspace frameworks supported
- **OS:** use inbox NVMe driver (no changes)
- **Hardware:** NoLoad™ Hardware Eval Kits

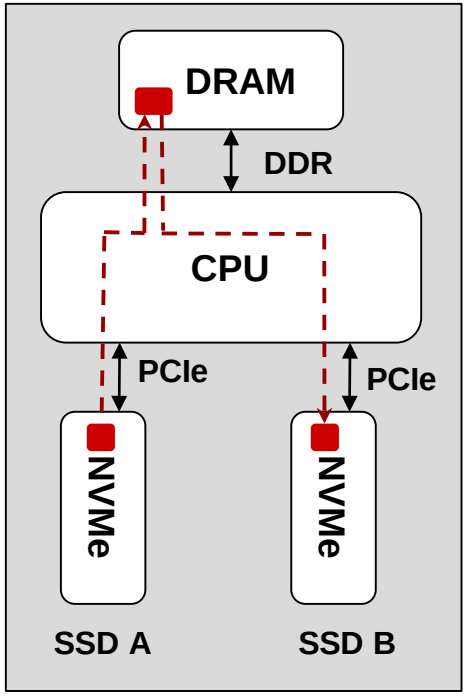
Accelerators as NVMe devices

```
amaier@styx: ~
File  Edit  View  Search  Terminal  Help
amaier@styx:~$ sudo nvme list
Node             SN                   Model                                    Namespace Usage                      Format           FW Rev
---------------- -------------------- ---------------------------------------- --------- -------------------------- ---------------- --------
/dev/nvme0n1     PHKE7504002U750BGN   INTEL SSDPE21K750GA                      1         750.16  GB / 750.16  GB    512   B +  0 B   E2010325
/dev/nvme1n1     2D402445ARZK65QAIA00 Eideticom NoLoad Accelerator Alpha       1           0.00   B /   2.15  GB    512   B +  0 B   1.180911
/dev/nvme1n2     2D402445ARZK65QAIA00 Eideticom NoLoad Accelerator Alpha       2           0.00   B /   8.59  GB    512   B +  0 B   1.180911
/dev/nvme1n3     2D402445ARZK65QAIA00 Eideticom NoLoad Accelerator Alpha       3           0.00   B /   8.59  GB    512   B +  0 B   1.180911
/dev/nvme1n4     2D402445ARZK65QAIA00 Eideticom NoLoad Accelerator Alpha       4           0.00   B /   8.59  GB    512   B +  0 B   1.180911
/dev/nvme2n1     PHKE750400JF750BGN   INTEL SSDPE21K750GA                      1         750.16  GB / 750.16  GB    512   B +  0 B   E2010325
/dev/nvme3n1     PHKE7504005L750BGN   INTEL SSDPE21K750GA                      1         750.16  GB / 750.16  GB    512   B +  0 B   E2010325
amaier@styx:~$ sudo nvme eid list
Node             Accelerator Name                                                 Version    Status
---------------- ---------------------------------------------------------------- ---------- ----------
/dev/nvme1n1     Eideticom RAM Drive                                              0x20170123 0x00080007
/dev/nvme1n2     Eideticom NoLoad Compression                                     0x20180510 0x10970003
/dev/nvme1n3     Eideticom NoLoad Compression                                     0x20180510 0x10970003
/dev/nvme1n4     Eideticom NoLoad Compression                                     0x20180510 0x10970003
amaier@styx:~$
```

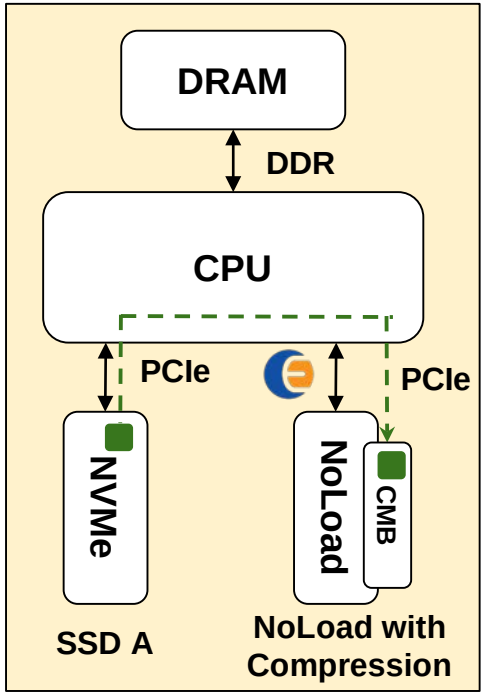
NVMe NSs: 3 Optane SSDs, 3 Compression Accels, 1 RAM-Drive

Peer-to-Peer Access

- P2P Transfers bypass CPU memory and other PCIe subsystems
- P2P uses PCIe EP's memory (e.g., CMB,BAR)
- A P2P capable Root Complex or PCIe switch is needed



Legacy Datapath

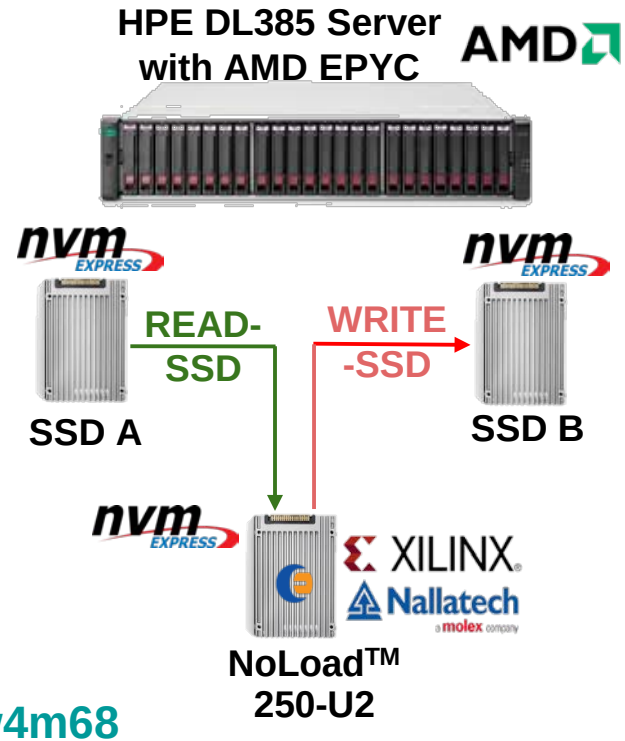


Peer-2-Peer Datapath

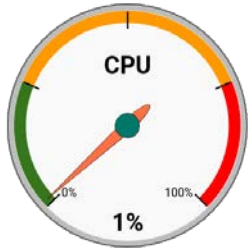
Peer-to-Peer Access, Cont'd

- NoLoad™ with three compression cores
- Process steps:
  1. **SSD-A → NoLoad::CMB**
  2. **NoLoad Compression**
  3. **NoLoad::CMB → SSD-B**
- Eideticom's P2P Compression demo with Xilinx, AMD, HP:

www.youtube.com/watch?v=4Sg8cgw4m68



3+ GB/s Compression T'put per NoLoad



<1% load on CPU

Why hardware compression?

- Data compression when done right:
  - **Decreases:** storage requirement, network/media access time, and power consumption
  - **Increases:** storage capacity, data rate

- Why using hardware?
  - Order of magnitude improvement in data rate and power consumption
  - Frees host CPU/Memory resources, especially if in peer-to-peer mode

Deflate Algorithm

- Default compression algorithm in the popular GZIP/ZLIB formats
- Open standard; no licenses needed
- Combines LZ77 and Huffman algorithms:
  - **LZ77:** Replaces duplicate strings with (distance, length) pairs. Duplicates can be up to 258B long and 32KB apart
  - **Huffman:** Encodes the literal, distance, and length symbols with the minimum number of bits

Deflate HW implementation

- Targeting scalable design for different data rates
- Low area design to allow multi-core / threads
- Can trade off between Compression Ratio, Speed and Area
- Supporting Static-Huffman only to reduce latency
- Low power

Deflate HW implementation, Cont'd

- Implementation on XCVU3P-2 (single core)

Core	LUTs	BRAM36s	T'put (max)	T'put (Calgary)	CR [1] (Calgary)	Power [2] (Calgary)
Compression	30K	49	1.7GB/s	700MB/s	2.23	1.24W/(GB/s)
Decompression	5K	9	2.0GB/s	1.5GB/s	-	-

1. Compression Ratio = Original/Compressed file size
2. Power measured in NoLoad. Ranges from 0.75W/(GB/s) for un-compressible data to 1.5W/(GB/s) for highly compressible data

NoLoad Compression Performance

- NoLoad (3-core, FPGA) vs QAT-8955 (6-core, ASIC)
- calgary.1G and cal4K.1G were built from Calgary corpus files [1]

	calgary.1G		cal4K.1G	
Engine	**CR**	**T'put**	**CR**	**T'put**
ZLIB-1 on CPU [2]	2.622	81 MB/s	29.564	340 MB/s
QAT-8955 [3]	2.597	1463 MB/s	7.299	2850 MB/s
NoLoad ZLIB [2]	**2.224**	**2039 MB/s**	**35.809**	**2973 MB/s**

1. Intel, "Programming Intel QuickAssist Technology Hardware Accelerators for Optimal Performance", April 2015, URL: https://01.org/sites/default/files/page/332125_002_0.pdf .
2. Tests were performed on a single core of an Intel i5-6500 @3.2GHz machine running Ubuntu 16.04.
3. Intel QuickAssist 8955 with six compression cores on it's ASIC chipset. All of the compression cores were used for this test [1].

Reduced CPU Usage

- CPU is only used to manage data transfers and NVMe commands and responses
- Compression tasks are entirely offloaded
- CPU utilization is determined primarily by transfer block size:

Transfer block size	32KB	64KB
Throughput per CPU core	5GB/s	10GB/s

Thanks!

Eidetic Communications Inc.
3553 31st NW, Calgary, AB
Canada T2L 2K7

www.eideticom.com, sales@eideticom.com