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

- **Management**
  - nvme-cli
  - nvme-of
  - etc

- **Applications**
  - GitHub
  - libnoload
  - Intel
  - SPDK

- **Userspace**: both kernel & userspace frameworks supported

- **OS**: use inbox NVMe driver (no changes)

- **Hardware**: NoLoad™ Hardware Eval Kits
Accelerators as NVMe devices

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
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
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)

<table>
<thead>
<tr>
<th>Core</th>
<th>LUTs</th>
<th>BRAM36s</th>
<th>T’put (max)</th>
<th>T’put (Calgary)</th>
<th>CR [1] (Calgary)</th>
<th>Power [2] (Calgary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compression</td>
<td>30K</td>
<td>49</td>
<td>1.7GB/s</td>
<td>700MB/s</td>
<td>2.23</td>
<td>1.24W/(GB/s)</td>
</tr>
<tr>
<td>Decompression</td>
<td>5K</td>
<td>9</td>
<td>2.0GB/s</td>
<td>1.5GB/s</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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]

<table>
<thead>
<tr>
<th>Engine</th>
<th>calgary.1G</th>
<th>cal4K.1G</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CR</td>
<td>T'put</td>
</tr>
<tr>
<td>ZLIB-1 on CPU [2]</td>
<td>2.622</td>
<td>81 MB/s</td>
</tr>
<tr>
<td>QAT-8955 [3]</td>
<td>2.597</td>
<td>1463 MB/s</td>
</tr>
<tr>
<td>NoLoad ZLIB [2]</td>
<td>2.224</td>
<td>2039 MB/s</td>
</tr>
</tbody>
</table>

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](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:

<table>
<thead>
<tr>
<th>Transfer block size</th>
<th>32KB</th>
<th>64KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput per CPU core</td>
<td>5GB/s</td>
<td>10GB/s</td>
</tr>
</tbody>
</table>
Thanks!

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

www.eideticom.com, sales@eideticom.com