NVMe Based Reconfigurable Compression Engine

David Sloan
Overview

- Compression Engine for Reconfigurable Platform
- Why NVMe?
Compression Engine Targets

- Standard compression output
  - Must be decompressible without proprietary SW

- Ditch the CPU
  - Reduced CPU usage and higher data rates

- Efficient design
  - FPGA must be able to fit other modules
Compression: Output standard

- Deflate
  - Used in Zip, Gzip, and zlib
  - Zlib headers used
  - Static Huffman only
  - Raw data blocks are not used
Compression: No CPU?

- Application specific HW can significantly out perform the CPU for well defined tasks
  - Protein Corpus: CPU* 58 MB/s, HW: 677 MB/s
- CPU is only used to manage data transfers to/from compression device and manage NVMe transfers

*CPU results from zlib level 1 on an Intel i5-6500@3.2 GHz
Compression: Shared FPGA

- Modular design allows for simplification of meeting specific needs
- Many compression cores or many different cores
## Compression: Large compressible files

<table>
<thead>
<tr>
<th>Engine</th>
<th>Compression Ratio</th>
<th>Throughput</th>
<th>Compression Ratio</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZLIB-1 on CPU [2]</td>
<td>2.62</td>
<td>81 MB/s</td>
<td>29.56</td>
<td>340 MB/s</td>
</tr>
<tr>
<td>QAT-8955 [3]</td>
<td>2.60</td>
<td>1.46 GB/s</td>
<td>7.30</td>
<td>2.85 GB/s</td>
</tr>
<tr>
<td>Eideticom-H [2,4]</td>
<td>2.22</td>
<td>2.04 GB/s</td>
<td>35.81</td>
<td>2.97 GB/s</td>
</tr>
<tr>
<td>Eideticom-F [2,4]</td>
<td>2.12</td>
<td>2.19 GB/s</td>
<td>27.93</td>
<td>3.14 GB/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 its ASIC chipset. All of the compression cores were used for this test [1].
4. FPGA test were performed on a NoLoad with three compression cores. The -H option provides higher compression while the -F option provides higher data throughput (~ same area)
NVMe

- High speed, CPU efficient standard
- In-box drivers
- Allows for use of peer to peer HW in PCIe
  - Drastically reduces system memory usage
  - More memory BW free for CPU compute tasks
NVMe: CPU load

- CPU compute tasks are completely offloaded to the NVMe Accelerator
- CPU utilization is determined primarily by transfer size
  - 32 kB → ~5 GB/s / physical CPU core*
  - 64 kB → ~10 GB/s / physical CPU core*

*CPU Intel i7-7800X@3.5 GHz
Accelerator cores exposed as NVMe namespaces

PCI express

NVMe Device

nvmeXn0
Accelerator 0

nvmeXn1
Accelerator 1

...

nvmeXnY
Accelerator Y
Data and config operations are memory mapped to the block device
NVMe Accelerator

- Reads map to device output
- Writes map to device input
NVMe: Peer to Peer

- PCIe allows for direct communication between NVMe devices
- Further reduces CPU overhead for offloaded acceleration
Jobs from CPU can be processed then stored directly on NVMe SSD without touching system memory.
NVMe: Peer to Peer - Demo

- NVMe SSD input and output
- NoLoad NVMe Accelerator
- Both P2P and Standard Operation