Optimizing NVMe-over-Fabrics using NVMe CMBs and Accelerators

Andrew Maier, Software Engineer, Eideticom
Dr. Stephen Bates, CTO, Eideticom
1. Introduction to NVMe Acceleration and NoLoad™
2. Integration of Accelerators into NVMe over Fabrics (NVMe-oF)
3. Acceleration via NVMe-oF Example
4. NVMe-oF Target/Server CPU Offloading
5. Peer-to-Peer Transfers using NVMe-oF Offload
Eideticom’s NoLoad™ leverages the NVMe standard to present FPGA Accelerators as NVMe namespaces.

NoLoad Bitfiles

U.2 FPGA Card

COTS PCIe FPGA Card

Cloud Servers i.e. Amazon F1
Introduction to NVMe Acceleration

- Why NVMe?
  - NVMe is a low latency, high throughput, low CPU overhead transfer protocol
  - Usage of built-in and industry-standard drivers and tools
    - Why build and maintain a proprietary driver?
  - Ability to use the emerging NVMe over Fabrics ecosystem for storage (and accelerator) disaggregation
NVMe-oF

NVMe over Fabrics (NVMe-oF) allows namespaces to be shared across existing networks.

- Using built-in drivers, we expose the NVMe namespaces to client machines.
- Since NoLoad is a standard namespace, it can be shared in the same way!
- So how does it work?
Clients request to borrow a namespace(s) (or accelerators) from the server.

Client is given access to the namespace over the connection.
Clients then see the newly acquired namespaces as local NVMe block devices.

- Normal NVMe operations can then be executed as if it were locally in the client machine.
- With the latest (soon to be upstreamed) NVMe over Fabrics passthru patches from Chaitanya Kulkarni, the client has access to all vendor specific functionality as well.
Case Study: Compression over Fabrics

- Implemented compression core FPGA accelerator
  - Each compression core capable of >1GB/s throughput
  - Multiple accelerator cores can be integrated into a NoLoad FPGA
  - Each accelerator core is its own NVMe namespace

- Both NoLoad and a generic NVMe SSD located on remote U.2 JBOF
  - Both are shared via NVMe-oF

- Client process generates data, sends it to the compression accelerator, and then outputs it to the SSD.
Case Study: Compression over Fabrics

- The 2x compression core test over fabrics achieves about 1 GB/s per core
  - This means we are still able to get the same throughput over fabrics! (Given sufficient fabrics bandwidth of course)
- But how much impact is there on resources in the target machine?
• Let’s look at a different example but with the Mellanox ConnectX-5’s.
• In vanilla NVMe-oF target CPU is responsible for handling communication with NVMe drive.
• This data flow heavily uses the target CPU and DRAM.
• How can we reduce the load on the target machine?
  • NVMe-oF offload!
NVMe-oF Offload

- NVMe-oF Offload allows the NIC to directly control NVMe devices
- Using Mellanox ConnectX-5’s we can offload the NVMe work from the target CPU

<table>
<thead>
<tr>
<th>Operation</th>
<th>Latency (read/write) us</th>
<th>CPU Utilization</th>
<th>CPU Memory Bandwidth</th>
<th>CPU PCIe Bandwidth</th>
<th>NVMe Bandwidth</th>
<th>Ethernet Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla NVMe-oF</td>
<td>188/227</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>ConnectX-5 Offload</td>
<td>128/138</td>
<td>0.02</td>
<td>2.40</td>
<td>1.03</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

- ConnectX-5 Offload reduces the target CPU load by x50 but doesn’t decrease the memory bandwidth
- How can we reduce the memory utilization?
  - With peer-to-peer transfers!
NVMe CMBs and P2P Transfers

- For p2p transfers, we need to make use of NVMe CMBs (Controller Memory Buffers)
- A NVMe CMB is a PCIe BAR (or part thereof) that can be used for certain NVMe specific data types.
- A P2P framework called p2pmem is being proposed for the Linux kernel
- PCIe drivers can register memory (e.g. CMBs) or request access to memory for DMA
- With P2P transfers, we can skip the DRAM copy reducing latency and DRAM usage.

- Traditional DMAs (left) load the CPU. P2P DMAs (right) do not load the CPU
Now let’s retry the previous example with P2P/CMB with the ConnectX-5 Offload.

The P2P path offloads both the CPU and the DRAM.
NVMe-oF Offload with P2P and CMB

<table>
<thead>
<tr>
<th>Operation</th>
<th>Latency (read/write) us</th>
<th>CPU Utilization</th>
<th>CPU Memory Bandwidth</th>
<th>CPU PCIe Bandwidth</th>
<th>NVMe Bandwidth</th>
<th>Ethernet Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla NVMe-oF</td>
<td>188/227</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>ConnectX-5 Offload</td>
<td>128/138</td>
<td>0.02</td>
<td>2.40</td>
<td>1.03</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Eideticom NoLoad p2pmem</td>
<td>167/212</td>
<td>0.55</td>
<td>0.09</td>
<td>0.01</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>CX5 Offload + Eideticom NoLoad p2pmem</td>
<td>142/154</td>
<td>0.02</td>
<td>0.02</td>
<td>0.04</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

- Combining p2pmem and CX5 Offload provides significant reduction of CPU utilization (x50), CPU memory bandwidth (x50), and CPU PCIe bandwidth (x25)
FMS 2018 Eideticom Demos

- The discussed compression example NVMe-oF with Broadcom at booth #729
- Compression/Decompression acceleration via P2P transfers with Xilinx at booth #313
- Come check them out ;)

Santa Clara, CA
August 2018