Important new NVMe features for optimizing the data pipeline

Dr. Stephen Bates, CTO
Eideticom
Outline

• Intro to NVMe Controller Memory Buffers (CMBs)
• Use cases for CMBs
  • Submission Queue Support (SQS) only
  • RDS (Read Data Support) and WDS (Write Data Support) for NVMe p2p copies
  • SQS, RDS and WDS for optimized NVMe over Fabrics
• Software for NVMe CMBs
  • SPDK (Storage Performance Developer Kit) work for NVMe copies.
  • Linux kernel work for p2pdma and for offload.
• Roadmap for the future
Intro to Controller Memory Buffers

- CMBs were introduced to the NVMe standard in 2014 in version 1.2.
- A NVMe CMB is a PCIe BAR (or part thereof) that can be used for certain NVMe specific data types.
- The main purpose of the CMB is to provide an alternative to:
  - Placing queues in host memory
  - Placing data for DMA in host memory.
- As well as a BAR, two optional NVMe registers are needed:
  - CMBLOC - location
  - CMBSZ - size and supported types
- Multiple vendors support CMB today (Intel, Eideticom, Everspin) or soon (Toshiba, Samsung, WDC etc).
Intro to Controller Memory Buffers

- A - This device’s manufacturer has registered its vendor ID and device IDs with the PCIe database. This means you get a human-readable description of it.
- B - This device has three PCIe BARs:
  - BAR0 is 16KB and is the standard NVMe BAR that any legitimate NVMe device must have.
  - C - The third BAR is the Controller Memory Buffer (CMB) which can be used for both NVMe queues and NVMe data.
- F - Since this device is a NVMe device it is bound to the standard Linux kernel NVMe driver.

Below is CMBLOC and CMBSZ for a CMB enabled NVMe device
Some Fun Use Cases for CMBs

- Placing some (or all) of your NVMe queues in CMB rather than host memory. Reduce latency [Linux Kernel\(^1\) and SPDK\(^1\)].
- Using the CMB as a DMA buffer allows for offloaded NVMe copies. This can improve performance and offloads the host CPU [SPDK\(^1\)].
- Using the CMB as a DMA buffer allows RDMA NICs to directly place NVMe-oF data into the NVMe SSD. Reduce latency and CPU load [Linux Kernel\(^2\)].

\(^1\) Upstream in the relevant tree.
\(^2\) Proposed patches (see last slide for git repo).
Software for CMBs - SPDK

- Storage Performance Development Kit (SPDK) is a Free and Open Source (FOSS) user-space framework for high performance storage.
- Focus on NVMe and NVMe-oF.
- Code added in Feb 2018 to enable P2P NVMe copies when CMBs allow it.
- A simple example of an application using this new API also in SPDK examples (cmb_copy).

 cmb_copy is an example application using SPDK’s APIs to copy data between NVMe SSDs using P2P DMAs. This bypasses the CPU’s memory and PCIe subsystems.
• `cmb_copy` moves data direct from SSD A to SSD B using NVMe CMB(s).
• 99.99% of UpStream Port (USP) traffic on PCIe switch is eliminated.
• OS is still in complete control of the IO and handles any status/error messages.
• NVMe SQEs can also be in the CMB or not as desired. SGLs or PRPs can be supported.

See https://asciinema.org/a/bkd32zDLyKvlq7F8M5BBvdX42
A P2P framework called p2pdma is being proposed for the Linux kernel.

Much more general than NVMe CMBs. Any PCIe device can utilize it (NICS, GPGPUs etc.).

PCIe drivers can register device memory (e.g. CMBs or BARs) or request access to P2P memory for DMA.

Initial patches use p2pdma to optimize the NVMe-oF target code.

The p2pdma framework can be used to improve NVMe-oF targets. Here we show results from a NVMe-oF target system.

<table>
<thead>
<tr>
<th>Mode of 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>Eidelcom NoLoad p2pmm</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>ConnectX-5 Offload + Eidelcom NoLoad p2pmm</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>

p2pdma can reduce CPU memory load by x50 and CPU PCIe load by x25. NVMe offload can also be employed to reduce CPU core load by x50.
Software for CMBs – The Linux Kernel

- The hardware setup for the NVMe-oF p2pdma testing is as shown on the right.
- The software setup consisted of a modified Linux kernel and standard NVMe-oF configuration tools (mostly nvme-cli and nvmet).
- The Linux kernel used added support for NVMe offload and Peer-2-Peer DMAs using an NVMe CMB provided by the Eideticom NVMe device.

This is the NVMe-oF target configuration used. Note RDMA NIC is connected to switch and not CPU Root Port.
Eideticom puts accelerators behind NVMe namespaces.

We can combine this with CMBs to add compute to p2pdma.

Demo with AMD in Xilinx in Xilinx booth.

>3GB/s compression (input) via U.2 NVMe accelerator (NoLoad) with p2pdma offloading CPU by >99%.

Leverages p2pdma to move data from input NVMe SSD to NoLoad and from NoLoad to output NVMe SSD.
Persistent Memory Regions

- PMRs add persistent to CMBs.
- PMR features being discussed and standard will be updated. Get involved!
- Small PMRs (10MB-1GB) interesting:
  - Write cache
  - Journal for SQEs
  - Persistent scratchpad for meta-data
- Big PMRs (>>1GB) are really interesting.
Persistent Memory Regions

Building a PMoF Target Today: Hardware (v2-pcie)

- Fabric I/F
  - Require CPU utilization on the client side.
  - Not true load/store on client.
  - Challenging to scale.
  - Non-coherent (client and target)

- Control Plane
  - Uh, why is the CPU in the way?
  - Very CPU/ISA dependent
    - DDIO
    - cache effects

- PM Media
  - Expensive
  - Not hot-swappable
  - Capacity/Scale issues
  - MeBo support (A5R) required

- Also
  - Decouples target-side CPU DDR from performance
  - Decouples target-side CPU PCIe from performance
  - Fabric I/F can be upgraded in time (Star Wars!)

Large NVMe PMRs would enable PMoF aware filesystems!
Roadmap for CMBs and PMRs and the Software

- NVMe CMBs have been in the standard for a while. However it’s only now they are starting to become available and software is starting to utilize them.
- SPDK and the Linux kernel are the two main locations for CMB software enablement today.
- SPDK: NVMe P2P copies. NVMe-oF updates coming.
- Linux kernel. p2pdma framework upstream soon. Will be expanded to support other NVMe/PCIe resources (e.g. doorbells).
- Persistent Memory Regions add non-volatile CMBs and will require (lots of) software enablement too. They will enable a path to Persistent memory storage on the PCIe bus.