Enabling the NVMe™ CMB and PMR Ecosystem

Stephen Bates, PhD. CTO, Eideticom
Oren Duer. Software Architect, Mellanox

NVM Express Developers Day, May 1, 2018

Outline

1. Intro to NVMe™ Controller Memory Buffers (CMBs)
2. Use cases for CMBs
    a. Submission Queue Support (SQS) only
    b. RDS (Read Data Support) and WDS (Write Data Support) for NVMe p2p copies
    c. SQS, RDS and WDS for optimized NVMe™ over Fabrics (NVMe-oF™)
3. Software for NVMe CMBs
    a. SPDK (Storage Performance Developer Kit) work for NVMe copies.
    b. Linux kernel work for p2pdma and for offload.
4. 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).

3.1.11 Offset 38h: CMBLOC – Controller Memory Buffer Location

This optional register defines the location of the Controller Memory Buffer (refer to section 4.7). If CMBSZ is 0, this register is reserved.

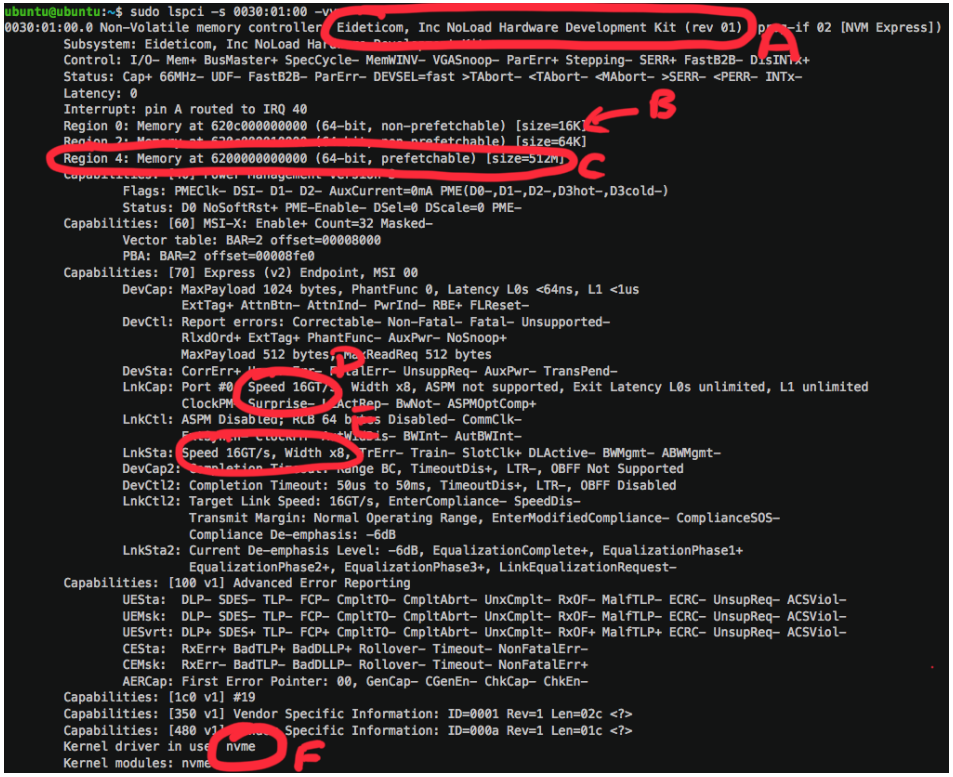
Bit	Type	Reset	Description
31:12	RO	Impl Spec	**Offset (OFST):** Indicates the offset of the Controller Memory Buffer in multiples of the Size Unit specified in CMBSZ. This value shall be 4KB aligned.
11:03	RO	0h	Reserved
02:00	RO	Impl Spec	**Base Indicator Register (BIR):** Indicates the Base Address Register (BAR) that contains the Controller Memory Buffer. For a 64-bit BAR, the BAR for the lower 32-bits of the address is specified. Values 0h, 2h, 3h, 4h, and 5h are valid.

3.1.12 Offset 3Ch: CMBSZ – Controller Memory Buffer Size

This optional register defines the size of the Controller Memory Buffer (refer to section 4.7). If the controller does not support the Controller Memory Buffer feature then this register shall be cleared to 0h.

Bit	Type	Reset	Description
31:12	RO	Impl Spec	**Size (SZ):** Indicates the size of the Controller Memory Buffer available for use by the host. The size is in multiples of the Size Unit. If the Offset + Size exceeds the length of the indicated BAR, the size available to the host is limited by the length of the BAR.
11:08	RO	Impl Spec	**Size Units (SZU):** Indicates the granularity of the Size field. 0h: 4 KB 1h: 64 KB 2h: 1 MB 3h: 16 MB 4h: 256 MB 5h: 4 GB 6h: 64 GB 7h – Fh: Reserved
07:05	RO	0h	Reserved
04	RO	Impl Spec	**Write Data Support (WDS):** If this bit is set to '1', then the controller supports data and metadata in the Controller Memory Buffer for commands that transfer data from the host to the controller (e.g., Write). If this bit is cleared to '0', then all data and metadata for commands that transfer data from the host to the controller shall be transferred from host memory.

Intro to Controller Memory Buffers




```
ubuntu@ubuntu:~$ sudo lspci -s 0030:01:00 -v
0030:01:00.0 Non-Volatile memory controller: Eideticom, Inc NoLoad Hardware Development Kit (rev 01) (prog-if 02 [NVM Express])
        Subsystem: Eideticom, Inc NoLoad Ha
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr+ Stepping- SERR+ FastB2B- DisINTx+
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0
        Interrupt: pin A routed to IRQ 40
        Region 0: Memory at 620c000000000 (64-bit, non-prefetchable) [size=16K]
        Region 2: Memory at 620c000000000 (64-bit, non-prefetchable) [size=64K]
        Region 4: Memory at 6200000000000 (64-bit, prefetchable) [size=512M]
        Capabilities: [60] Power Management version 3
                Flags: PMEClk- DSI- D1- D2- AuxCurrent=0mA PME(D0-,D1-,D2-,D3hot-,D3cold-)
                Status: D0 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
        Capabilities: [60] MSI-X: Enable+ Count=32 Masked-
                Vector table: BAR=2 offset=00008000
                PBA: BAR=2 offset=00008fe0
        Capabilities: [70] Express (v2) Endpoint, MSI 00
                DevCap: MaxPayload 1024 bytes, PhantFunc 0, Latency L0s <64ns, L1 <1us
                        ExtTag+ AttnBtn- AttnInd- PwrInd- RBE+ FLReset-
                DevCtl: Report errors: Correctable- Non-Fatal- Fatal- Unsupported-
                        RlxdOrd+ ExtTag+ PhantFunc- AuxPwr- NoSnoop+
                        MaxPayload 512 bytes, MaxReadReq 512 bytes
                DevSta: CorrErr+ UncorrErr- FatalErr- UnsuppReq- AuxPwr- TransPend-
                LnkCap: Port #0, Speed 16GT/s, Width x8, ASPM not supported, Exit Latency L0s unlimited, L1 unlimited
                        ClockPM- Surprise- LLActRep- BwNot- ASPMOptComp+
                LnkCtl: ASPM Disabled; RCB 64 bytes Disabled- CommClk-
                        ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
                LnkSta: Speed 16GT/s, Width x8, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
                DevCap2: Completion Timeout: Range BC, TimeoutDis+, LTR-, OBFF Not Supported
                DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis+, LTR-, OBFF Disabled
                LnkCtl2: Target Link Speed: 16GT/s, EnterCompliance- SpeedDis-
                         Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
                         Compliance De-emphasis: -6dB
                LnkSta2: Current De-emphasis Level: -6dB, EqualizationComplete+, EqualizationPhase1+
                         EqualizationPhase2+, EqualizationPhase3+, LinkEqualizationRequest-
        Capabilities: [100 v1] Advanced Error Reporting
                UESta:  DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- UnxCmplt- RxOF- MalfTLP- ECRC- UnsupReq- ACSViol-
                UEMsk:  DLP- SDES- TLP- FCP- CmpltTO- CmpltAbrt- UnxCmplt- RxOF- MalfTLP- ECRC- UnsupReq- ACSViol-
                UESvrt: DLP+ SDES+ TLP- FCP+ CmpltTO- CmpltAbrt- UnxCmplt- RxOF+ MalfTLP+ ECRC- UnsupReq- ACSViol-
                CESta:  RxErr+ BadTLP+ BadDLLP+ Rollover- Timeout- NonFatalErr-
                CEMsk:  RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr+
                AERCap: First Error Pointer: 00, GenCap- CGenEn- ChkCap- ChkEn-
        Capabilities: [1c0 v1] #19
        Capabilities: [350 v1] Vendor Specific Information: ID=0001 Rev=1 Len=02c <?>
        Capabilities: [480 v1] Vendor Specific Information: ID=000a Rev=1 Len=01c <?>
        Kernel driver in use: nvme
        Kernel modules: nvme
```

- **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.

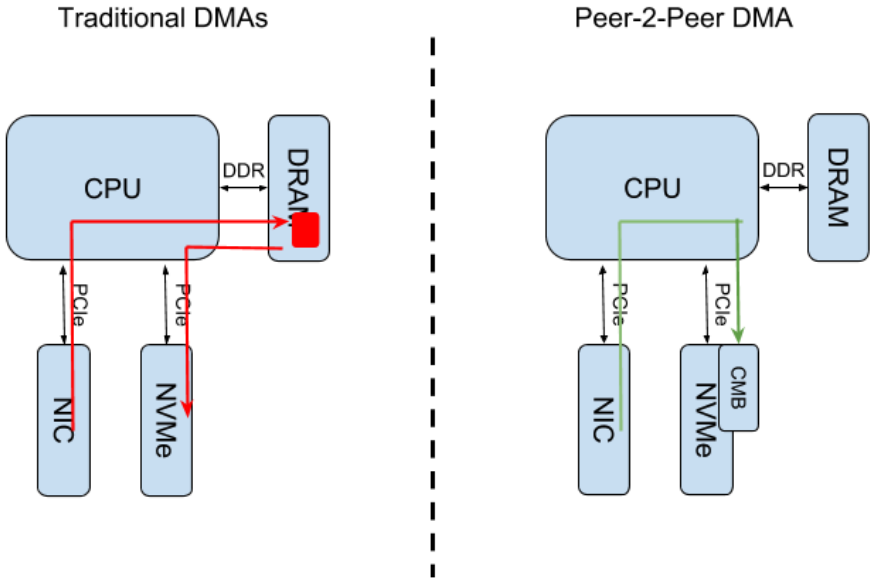
An example of CMBLOC and CMBSZ obtained via nvme-cli:

```
cmbloc  : 3
        Offset                  (OFST): 0 (See cmbsz.szu for granularity)
        Base Indicator Register (BIR): 3

cmbsz   : 500003
        Size                      (SZ): 1280
        Size Units                (SZU): 4 KB
        Write Data Support        (WDS): Write Data and metadata transfer in Controller Memory Buffer is Not supported
        Read Data Support         (RDS): Read Data and metadata transfer in Controller Memory Buffer is Not supported
        PRP SGL List Support    (LISTS): PRP/SG Lists in Controller Memory Buffer is Not supported
        Completion Queue Support (CQS): Admin and I/O Completion Queues in Controller Memory Buffer is Supported
        Submission Queue Support (SQS): Admin and I/O Submission Queues in Controller Memory Buffer is Supported
```

Some Fun Use Cases for CMBs

1. Placing some (or all) of your NVMe™ queues in CMB rather than host memory. Reduce latency [Linux Kernel[1] and SPDK[1]].
2. Using the CMB as a DMA buffer allows for offloaded NVMe copies. This can improve performance and offloads the host CPU [SPDK[1]].
3. 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]]

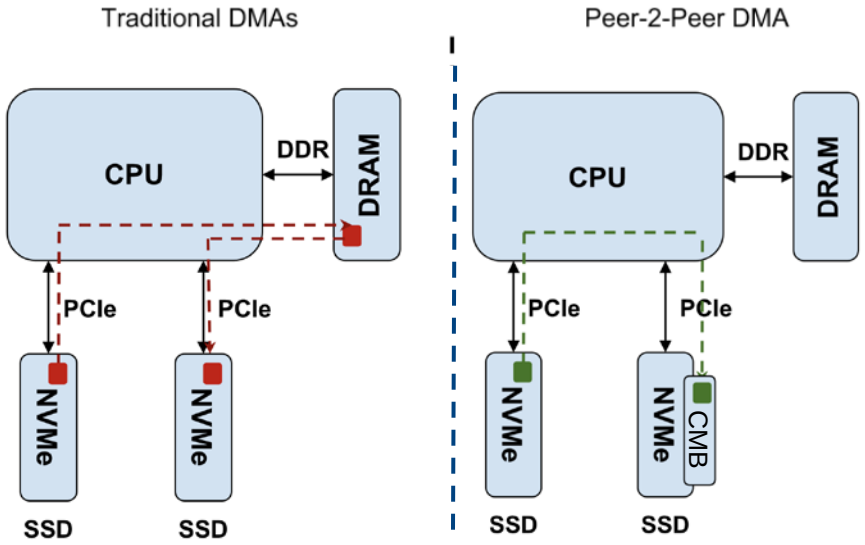


Traditional DMAs (left) load the CPU. P2P DMAs (right) do not load the CPU.

[1]Upstream in the relevant tree.
[2]Proposed patches (see last slide for git repo).

Software for CMBs - SPDK

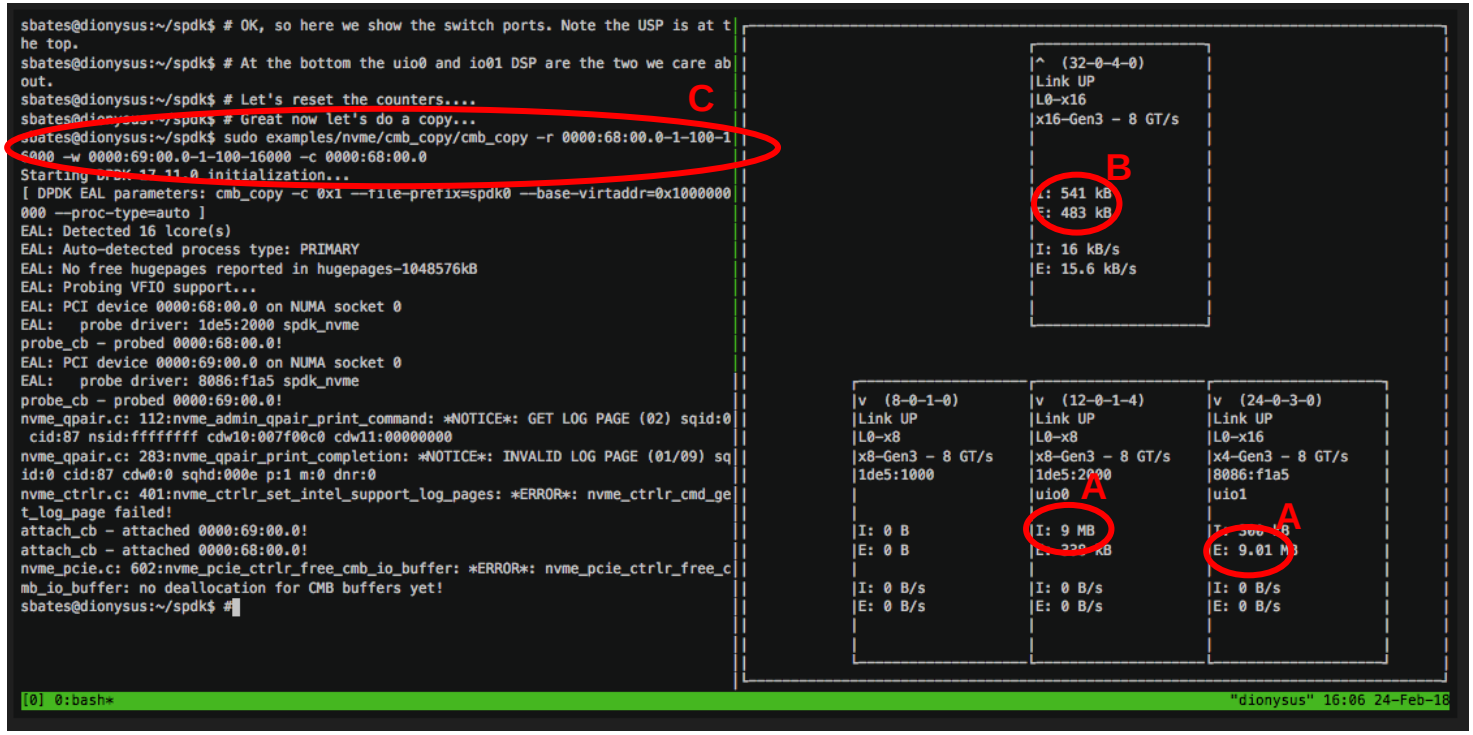
- Storage Performance Development Kit (SPDK) is a Free and Open Source (FOSS) user-space framework for high performance storage.
- Focus on NVMe™ and NVMe-oF™.
- Code added in Feb 2018 to enable P2P NVMe copies when CMBs allow it.
- A simple example of an application using this new API also in SPDK examples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

See https://asciinema.org/a/bkd32zDLyKvIq7F8M5BBvdX42

Software for CMBs - SPDK



A - copied 9MB from SSD A to SSD B.
B - less than 1MB of data on PCIe switch Upstream Port.
C - SPDK command line

Software for CMBs - The Linux Kernel

- 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 memory (e.g. CMBs) or request access to memory for DMA.
- Initial patches use p2pdma to optimize the NVMe-oF™ target code.

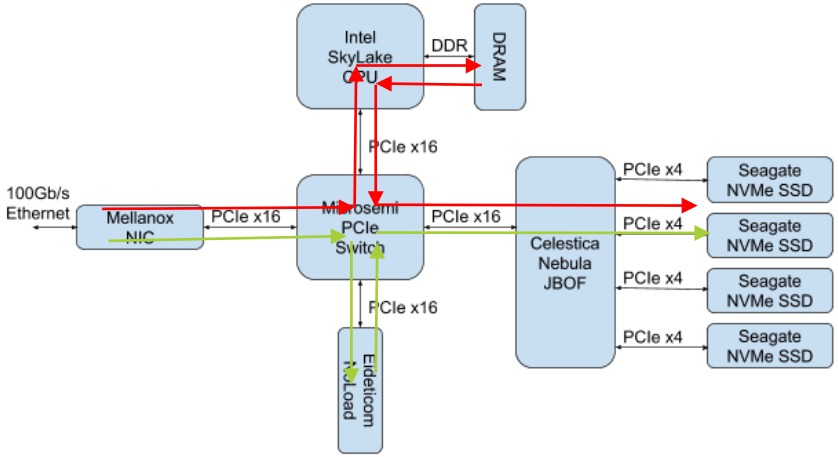
Mode of Operation	Latency (read/ write) us	CPU Utilization	CPU Memory Bandwidth	CPU PCIe Bandwidth	NVMe Bandwidth	Ethernet Bandwidth
Vanilla NVMe-oF	188/227	1.00	1.00	1.00	1.00	1.00
ConnectX-5 Offload	128/138	0.02	2.40	1.03	1.00	1.00
Eideticom NoLoad p2pmem	167/212	0.55	0.09	0.01	1.00	1.00
ConnectX-5 Offload + Eideticom NoLoad p2pmem	142/154	0.02	0.02	0.04	1.00	1.00

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

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.

Roadmap for CMBs, 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.

Further Reading, Resources and References

1. Current NVM Express™ Standard - http://nvmexpress.org/wp-content/uploads/NVM-Express-1_3a-20171024_ratified.pdf.
2. PMR TP - http://nvmexpress.org/wp-content/uploads/NVM-Express-1.3-Ratified-TPs.zip.
3. SPDK - http://www.spdk.io/ and https://github.com/spdk/spdk.
4. p2pdma Linux kernel patches - https://github.com/sbates130272/linux-p2pmem/tree/pcp-p2p-v4.
5. Mellanox offload driver - https://github.com/Mellanox/NVMEoF-P2P
6. SDC talk on p2pmem - https://www.youtube.com/watch?v=zEXJ549ealM.
7. Offload+p2pdma kernel code - https://github.com/lsgunth/linux/commits/max-mlnx-offload-p2pdma.
8. Offload+p2pdma white paper link - https://github.com/Mellanox/NVMEoF-P2P
9. https://docs.google.com/document/d/1GVGCLALneyw3pyKYmRRG7VTNWPtzL0XqrFlYA53rx_M/edit?usp=sharing.

2018 Storage Performance Development Kit (SPDK) Summit

May 15th -16th

Dolce Hayes Mansion, San Jose

200 Edenvale Avenue, San Jose, California 95136

This will be a great opportunity to meet with other SPDK community members and listen to a new series of talks from SPDK users and developers; everything from case studies and analysis to tech tutorials and live demos.

This year we will dedicate a second day just for developers that will include a hands-on lab, as well as a few hours set aside for active contributors to tackle design issues and discuss future advancements.

Registration is free!!!!!

http://www.cvent.com/d/qgqnn3

Sponsored by Intel® Corporation


nvm
EXPRESS®
Architected for Performance