RPMEM 2.0

Intel next generation platform support for RDMA with PMEM

09/24/19
Chet Douglas - DCG NVMS
Remote Persistent Memory (RPMEM) 2.0

Intel Platform Value Prop (Cloud, Enterprise, HPC)

- Increase performance with PMEM Wire protocol changes
  - Remove usage of non-allocating Writes
  - Remove extra messaging for some use cases
  - Eliminate Target node interrupts & core message handling
- Accelerate and reduce CPU utilization through improved NIC offload
- Create a standards-based solution for native remote persistent memory support. PMEM extensions being added to the following public standards/API:
  - IBTA IB/RoCE wire protocol
  - IBTA verbs spec
  - IETF iWARP wire protocol
  - ACPI PCI Firmware Spec
  - OFA libibverbs & libfabric API

- High-performance standards-based solution in collaboration with network partners
- Native PMEM HW support for RNICs, CPUs, and platforms
RPMEM 1.0 – Basic RDMA with PMEM for Intel Purley platform

See backup slides for basic overview of the 2 RPMEM 1.0 solutions Intel supports

PROS of using the RPMEM 1.0 solution

• ULP SW changes only
  • No wire protocol changes
  • No OFA library changes
  • No HW RNIC, cpu, chipset changes required

CONS of using these RPMEM 1.0 techniques

• Vendor specific implementation does not work for all platforms and does not foster industry wide acceptance
  • ULP SW changes could become a defacto protocol (not ideal!)
  • Appliance Method only works on Intel late model server platforms

• Initiator Node drives 2 possible flushing mechanisms
  • Forces target node configuration to be understood by initiator node – Not a sustainable or scalable programming model

• Performance Impact:
  • Extra messaging required to make writes durable – for some use cases
  • RPMEM 1.0 customers prefer the non-allocating write “Appliance Method” since the latency is dramatically reduced versus the DDIO “General Purpose Server Method”
  • Architectural limitations force the user/admin to place an entire PCIe Root Port in to non-allocating write mode
    • Requires intimate/internal knowledge of motherboard layout, add-in card strategy, etc
    • Impacts write performance for all devices on the root port
    • Alters the way all SW running on that node and data caching strategy
    • Applications that normally see quick read performance when accessing data that was RDMA’d potentially now will see significant latency as the first touch pulls the RDMA Write data in to LLC cache from CR
RPMEM 2.0 – Native RDMA with PMEM support for Intel Eagle Stream platform

Why we need “RDMA extensions for PMEM”?  
- Platform Performance:
  - One less round trip on the wire for tail of log SQL use case (see picture on the right)
    - The Data write, its Commit, the Ordered tail pointer Write and its Commit can all be launched by client in pipelined fashion.
    - The serialization of the tail pointer write relative to the Data Commit must still occur within the target system, but those in-system serialization latencies are noticeably lower than the equivalent serialization over the network.
  - Platform steering tags in combination with on the wire knowledge of RDMA Writes to persistent memory allow specific IO to bypass LLC cache for better efficiency
  - No longer need to place entire root port in to non-allocating mode, allowing non RPMEM users to see expected platform DDIO performance not possible with the ULP SW solution of RPMEM 1.0
- Platform efficiency/scalability, Initiator Node SW simplicity:
  - Only those applications require RPMEM need to enable it
  - Initiator node connection no longer needs to understand specific setup and configuration of each target node
  - No longer need to utilize different flush semantics for different target nodes
- Flexibility for future HW and platform architectures
  - Steering tags allow precise routing of network transactions to separate HW entities
RPMEM 2.0 – Native RDMA with PMEM support for Intel Eagle Stream platform

RPMEM 2.0 Intel Eagle Stream Platform HW Architecture

**IB/RoCEv2/iWARP RNIC**
Networking wire protocol & verbs updates:
- RDMA Flush
- RDMA Atomic Write

**TCP/IP Socket**
- iWarp/RoCE/IB

**RNIC HW**
- Intel SPR/EAS HW
- PCI TLP IO Steering
- TH: Steering Enabled
- ST: PMEM/DRAM
- PH: CACHE/NC

**Legacy support only:** Per PCIe Root Port CSRs programmed to control DDIO & Non-Allocating Write flows

**eADR Power Fail Safe Domain**
- ADR Domain + LLC + L1/L2

**ADCR 1.0**

**PCI Root Port**

**DDRT PCIe**
Legacy support only: Per PCIe Root Port CSRs programmed to control DDIO & Non-Allocating Write flows

**ADCR 2.0**

**IMC WPQ + AppDirect CR**

**ADCR Power Fail Safe Domain**
- ADR Domain + IIO Wrt Cache

**PCU assisted ADR**
- Power Fail Safe Domain
- ADR Domain + IIO Wrt Cache
RDMA Flush placement attributes:
- Attributes sent with the RDMA Flush command that directs the HW on how to handle the RDMA Writes and flushes to the given QP
  
  FI_PLACEMENT_PMEM – The data being flushed it targeted at PMEM vs DDR
  FI_PLACEMENT_GO - The data being flushed must be Globally Ordered (visible) before the flush command completes

RDMA Flush:
- Force flush of previous RDMA Write data utilizing placement hints to optimize how the flushing takes place

Ordering:
  - Behaves the same as RDMA Read
  - RDMA Flush shall not begin execution until all previous operation on the QP have been executed
  - Subsequent RDMA Write & RDMA Atomics may bypass RDMA Flush
  - RDMA Atomic Write may not bypass RDMA Flush

RDMA Atomic Write:
- 8 Byte pointer update only
- Separating from RDMA Write - Allows RNIC pipelining & ordering for these updates orthogonal to RDMA Write
- Optional IMMEDIATE flag to generate target node SW interrupt

Ordering:
  - Won't begin execution until ALL previous received inbound requests have begun executing
  - Subsequent inbound RDMA Write, RDMA Send, RDMA Atomic operations can all bypass the Atomic Write
  - Atomic Write shall not complete in memory until all preceding operations, except RDMA Read, have been completed in memory

IEFT RDMA Write w Validate:
- End-to-end CRC generation and checking when writing directly in to PMEM.
- Important for Push model with PMEM

Ordering:
  - Write completion back to the initiator after CRC validation is complete
PCI TLP Processor Hint Definition

- TH – indicates the presence of TLP Processing Hints (TPH) in the TLP header and optional TPH TLP Prefix (if present for 16-bit ST)
- PH bits are hints used by completer to optimize internal data placement for anticipated subsequent access by device or host.
- Steering Tags are system-specific values that provide information about the host or cache structure in the system cache hierarchy. These values are used to associate processing elements within the platform with the processing of Requests.
RPMEM 2.0 – Native RDMA with PMEM support for Intel Eagle Stream platform

<table>
<thead>
<tr>
<th>TLP Hints:</th>
<th>TH</th>
<th>PH</th>
<th>ST</th>
<th>Durable/Volatile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volatile Memory Write (DRAM)</td>
<td>0</td>
<td>na</td>
<td>na</td>
<td>Global Visible Order</td>
</tr>
<tr>
<td>Volatile Write w/o caching hint</td>
<td>1</td>
<td>00/11</td>
<td>!A</td>
<td>Global Visible Order</td>
</tr>
<tr>
<td>Volatile Memory Write NO caching</td>
<td>1</td>
<td>01</td>
<td>!A</td>
<td></td>
</tr>
<tr>
<td>Volatile Memory Write w/ Caching</td>
<td>1</td>
<td>10</td>
<td>A</td>
<td>Persistent Ordered</td>
</tr>
<tr>
<td>Persistent Write w/o caching hint</td>
<td>1</td>
<td>00/11</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>Persistent Write NO caching</td>
<td>1</td>
<td>01</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>Persistent Write w/ Caching</td>
<td>1</td>
<td>10</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

PCI TLP Processor Hint Definition

- **fi_mr_reg** memory registration attributes
- **RDMA Write** memory registration attributes
- **RDMA Flush** placement attributes

<table>
<thead>
<tr>
<th>Backend RNIC PCIe Steering Tag output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All other combinations</td>
<td>Target RNIC rejects RDMA Flush sent with any unsupported placement</td>
</tr>
</tbody>
</table>

A = Value programmed via new ACPI interfaces or default value if one is programmed

<table>
<thead>
<tr>
<th>Inbound RDMA Request</th>
<th>Backend RNIC PCIe Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDMA Read</td>
<td>PCIe Read</td>
</tr>
<tr>
<td>RDMA Write</td>
<td>PCIe Write</td>
</tr>
<tr>
<td>RDMA Flush</td>
<td>PCIe Flushing Read w Length=0</td>
</tr>
</tbody>
</table>

RNIC PCIe transaction handling

RNIC PCIe Steering Tag based transaction handling
Putting it all together
Putting it all Together:

**RDMA Flush:**

- **RKEY:**
- **Starting VA,**
- **Starting Length,**
- **Placement Attribute:** GO | PMEM

- VA buffer range specified by the flush must lie within one of the pre-registered target RNIC memory ranges
- The flush domain will cover all writes to the same RKEY wholly contained within the specific buffer range of the flush command
- The Target RNIC will check the RTAG registered memory range against the range specified in the RDMA Write or Flush & errors will be treated as a connection error
- The Target RNIC will send a PCIe Flushing Read with length 0 to force the read after write ordering semantics & flush behavior without the performance penalty of a data transfer
- The Target RNIC chases all the writes that are in range and insures they are flushed to the destination media
Putting it all together

Intel – PMDK \pmem.io

OFA - libfabric

OFA - libibverbs

ACPI 6.4 Steering Tag API
Intel HW Design Guide for NIC vendors

RPMEM 2.0 SOFTWARE ARCHITECTURE

DCG
Data Center Group
RPMEM 1.0 Backup Material

SW mechanisms to utilize PMEM on Intel Purley platforms
RPMEM 1.0 – Basic RDMA with PMEM for Intel Purley platform

RPMEM 1.0 Intel Purley Platform HW Architecture

Per PCIe Root Port CSRs programmed to control DDIO & Non-Allocating Write flows

PCU assisted ADR Power Fail Safe Domain
ADR Domain + IIO Wrt Cache

DDIO Writes

Non-allocating Writes

DCG
Data Center Group
- **ADR Domain, No DDIO**

- **“Appliance Method”**
  - Enable “non-allocating Write” transactions per Root PCI Port OR use PCIe “no snoop” bit in the PCI Transaction Header for a specific RNIC
    - Requires BIOS Enabling
    - Forces RDMA Write data directly to iMC
    - Enable on PCI Root Port with RNIC or specific RNIC
  - Follow RDMA Write(s) with RDMA Read to force remaining IIO buffer write data to ADR Domain
    - RDMA Read acts as a fencing function for the previous non-allocating write data and forces remaining write data out of the pipeline
    - Must force a PCI Read on the local PCI Root Port that handled the writes
    - Since RDMA Write and Read are silent, there is little or no change to the SW on the node supplying the Sink buffers for RDMA Write
    - RDMA Read – Read can be for any address, length
RPMEM 1.0 – Basic RDMA with PMEM for Purley and CR1.0/1.5

- ADR Domain, No DDIO
- “Appliance Method”
- See Sequence Diagram at Right for example flow

**NEW ADDITION:**
RDMA READ TO FORCE REMAINING WRITE DATA IN TO THE ADR DOMAIN

**NEW ADDITION:**
RELY ON ADR LOGIC TO MAKE WRITE DATA IN THE ADR DOMAIN DURABLE ON POWER FAIL
• No ADR Domain, With DDIO
  (Platform Default configuration)

• “General Purpose Server Method”
  • Use standard “allocating Write” transactions for Root PCI Port to IIO
  • Follow RDMA Write(s) with Send/Receive to get server RDMA SW callback
    • Send/Receive will contain list of cache lines that were written (or write addresses/lengths)
    • ISA – CLFLUSHOPT/SFENCE – Flush CPU cache lines and wait for flush to complete (invalidates cache contents). The list of cache lines from the Send message is used to identify the cache lines that need to be flushed.
    • Rely on ADR logic to make write data in the ADR Domain durable on a power loss
    • Internal IIO buffers will be flushed as part of ADR logic allowing “allocating writes” to be used.
- No ADR Domain, With DDIO
- “General Purpose Server Method”
- See Sequence Diagram at Right for example flow

**NEW ADDITION:**
RDMA SEND/RECEIVE CALLBACK TO FORCE DATA IN TO IMC ADR DOMAIN

**NEW ADDITION:**
RELY ON ADR LOGIC TO MAKE WRITE DATA IN THE ADR DOMAIN DURABLE ON POWER FAIL

RPMEM 1.0 – Basic RDMA with PMEM for Purley and CR1.0/1.5