Transparent Host Memory Buffer (THMB)

DRAM-Free Path to High-Performance SSDs

Presenter:

Oleg Kragel, Senior Technologist System Design

Authors:

Oleg Kragel, Vijay Sivasankaran, Leeladhar Agarwal

The Problem with Conventional HMB (CHMB)

- Read-centric SSD workloads with CHMB data path heavily relies on the SSD controller for HMB access descriptor creation and completion processing.
- Buffers containing L2P data must be transferred over PCIe link to SSD controller’s memory.
- This overhead limits random read performance, especially at high queue depths or wide address ranges.
- Why It Matters:
  - DRAM adds cost and power consumption.
  - DRAMless SSDs need efficient alternatives without sacrificing performance.

THMB Concept Overview

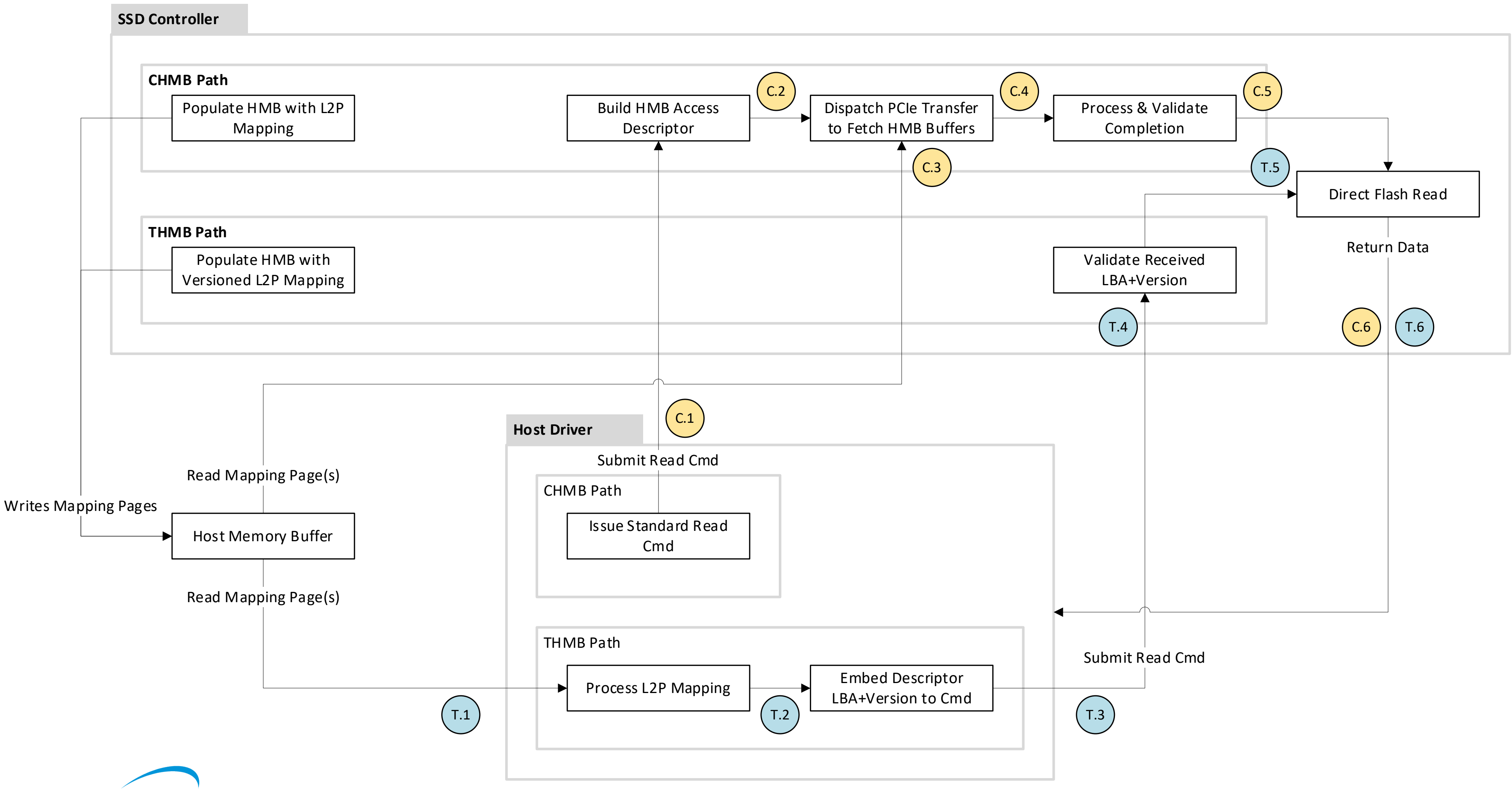
- Transparent Host Memory Buffer is a host-driven optimization that reduces controller overhead and PCIe link utilization.
- Key Approach:
  - Host driver passes additional 8-byte data (4 bytes for LBA location + 4 bytes for version) in the read/write command extracted from L2P directory and buffers that SSD controller maintains in HMB.
  - This allows SSD controller to bypass the standard HMB access descriptor flow.

- Conventional (CHMB) Read Path:
  1. SSD controller builds descriptor to fetch L2P mapping from HMB.
  2. L2P mapping buffer is transferred over PCIe link from host memory to controller memory.
  3. Controller processes descriptor completion.
- THMB Read Path:
  1. Host driver embeds necessary L2P details directly into read command.
  2. Controller checks version and on success - locates data without further HMB buffer transfers or L2P translation actions.

THMB Data Path & Architecture

- HMB Transparency:
    - SSD controller updates HMB with the latest L2P mapping (similarly to CHMB data path) and version data.
    - The host driver leverages this directly, minimizing SSD controller-side overhead.
- Version Control:
    - Each HMB entry is tagged with a version number.
    - Ensures stale data is detected by SSD controller, preserving data consistency.

CHMB vs. THMB: Side-by-Side Data Path Comparison



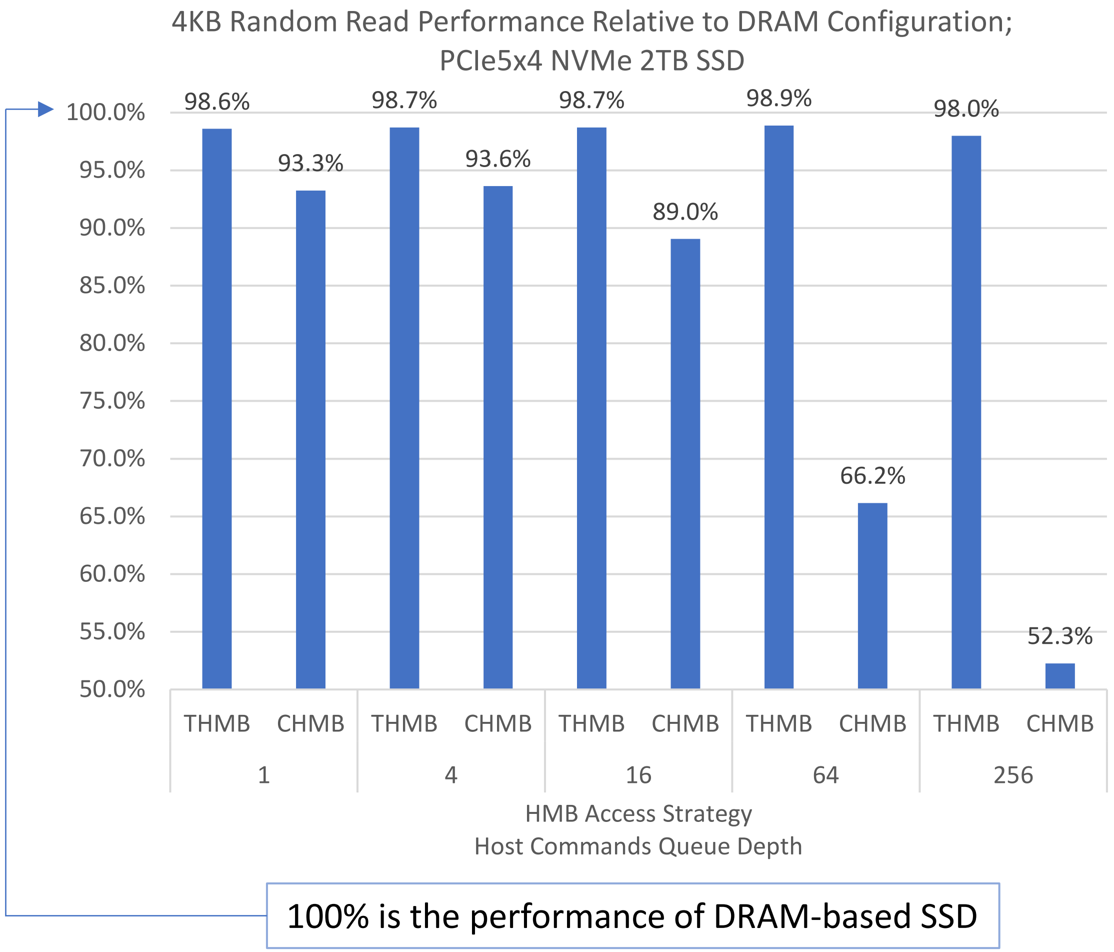
CHMB Flow:
1. Submit Read Cmd to SSD
2. Build HMB Access Descriptor
3. Dispatch PCIe Transfer to Fetch HMB Buffers
4. Process & Validate Completion
5. Direct Flash Read
6. Return Data to Host

THMB Flow:
1. Process L2P Mapping from HMB
2. Embed Descriptor LBA+Version to Cmd Context
3. Submit Read Cmd to SSD
4. Validate Received LBA+Version
5. Direct Flash Read
6. Return Data to Host

Performance Advantages

DRAM SSD vs. Conventional HMB vs. THMB

- THMB primarily benefits host random reads, where overhead is high for each IO command.
  - Host random write benefits can vary, as much of the overhead involves data path steps beyond the scope of this presentation and unique to each SSD vendor.
- Key Observation:
  - THMB enables near DRAM-based SSD performance without additional DRAM cost or power draw.
- Why 4KB?
  - It's one of the most common block size in modern operating systems and many real-world applications.



100% is the performance of DRAM-based SSD

Implementation Highlights

- Producer-Consumer Model:
  - SSD controller continually updates the L2P mapping data (and corresponding version info) in the HMB.
  - Host driver acts as the consumer, retrieving mapping information from the HMB to embed into read/write command.
- Host Driver Changes:
  - The HMB resides in a region of host memory that can be updated by the SSD controller at any time.
    - To avoid reading stale data, the host driver should use non-cached memory accesses for HMB buffers when that is necessary as per SSD-to-Host update-protocol (*).
  - Minimal modifications to pass additional 8-byte info in read/write commands.
- SSD Controller Changes:
  - Logic to update HMB with L2P translation table directory and version data.
  - Simple validation of the translation version information between read command’s context and reference version information stored in FTL and fallback to regular path in case of mismatch.

Host Side Data Structures

Flash Address Page Descriptor Array

Bits	Description
127 : 0	Flash Address Page Descriptor Entry 0
255 : 128	Flash Address Page Descriptor Entry 1
383 : 256	Flash Address Page Descriptor Entry 2
. . .	
n*128+127 : n*128	Flash Address Page Descriptor Entry n, where n is N_FLASH_ADDRESS_PAGE_DESCRIPTORS

Size, KB 3816.625 (1TB)

Flash Address Page Descriptor

Bits	Description
31 : 0	Page Version
63 : 32	Reserved
95 : 64	Page Buffer Lower Address
127 : 96	Page Buffer Upper Address

Size, Bits 128

Flash Address Page Cache Parameters

Bits	Description
31 : 0	N_FLASH_ADDRESS_PAGE_DESCRIPTORS - number of entries in Flash Address Page Descriptor Arrray
63 : 32	FLASH_ADDRESS_ENTRY_SIZE_BYTES
95 : 64	FLASH_ADDRESS_PAGE_SIZE_BYTES

Size, Bits 96

Flash Address Page Buffer

Bytes	Description
m-1 : 0	Flash Address Entry 0
2*m-1 : m	Flash Address Entry 1
3*m-1 : 2*m	Flash Address Entry 2
. . .	
k-1 : k-m	Flash Address Entry n-1, where m is FLASH_ADDRESS_ENTRY_SIZE_BYTES, k is FLASH_ADDRESS_PAGE_SIZE_BYTES and n = k / m

Flash Address Page Descriptor Index = LBA / n
Flash Address Entry Index = mod(LBA, n)

n = FLASH_ADDRESS_PAGE_SIZE_BYTES / FLASH_ADDRESS_ENTRY_SIZE_BYTES

SANDISK™

Business & Technical Implications

- Cost Savings:
  - Eliminates or reduces the need for on-board DRAM in many use cases.
  - Reduces Bill of Materials for SSD manufacturers.
- Energy Efficiency:
  - DRAMless design means lower power consumption and cooling needs.
- Market Differentiation:
  - DRAM-like performance at DRAMless price points.
- Scalability:
  - Applicable to multiple SSD form factors and capacities.

SANDISK™

Standardization

- SQE Encoding for THMB
  - CDW12.CETYPE = TBD to select the new THMB Tweak Mode (derived from TP4189 Key-Per-IO Tweak Mode)
  - CDW14: 4 bytes for LBA Location
  - CDW15: 4 bytes for LBA Location Version
- Compatibility Note
  - THMB Tweak Mode incompatible with Protection Information (PI) and Key-Per-IO Tweak Modes

THMB IO Command Submission Queue Entry

<table>
<tr><td>Bits</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bytes</td><td colspan="8">Byte 3</td><td colspan="8">Byte 2</td><td colspan="8">Byte 1</td><td colspan="8">Byte 0</td></tr>
<tr><td>CDW0</td><td colspan="16">CID</td><td colspan="2">PSDT</td><td colspan="4">RSVD</td><td colspan="2">Fuse</td><td colspan="8">Opcode</td></tr>
<tr><td>CDW1</td><td colspan="32">Namespace Identifier (NSID)</td></tr>
<tr><td>CDW2</td><td colspan="32"></td></tr>
<tr><td>CDW3</td><td colspan="32"></td></tr>
<tr><td>CDW4</td><td colspan="32"></td></tr>
<tr><td>CDW5</td><td colspan="32"></td></tr>
<tr><td>CDW6</td><td colspan="32" rowspan="2">PRP1</td></tr>
<tr><td>CDW7</td></tr>
<tr><td>CDW8</td><td colspan="32" rowspan="2">PRP2</td></tr>
<tr><td>CDW9</td></tr>
<tr><td>CDW10</td><td colspan="32"></td></tr>
<tr><td>CDW11</td><td colspan="32"></td></tr>
<tr><td>CDW12</td><td colspan="12"></td><td colspan="4">CETYPE=TBD</td><td colspan="16"></td></tr>
<tr><td>CDW13</td><td colspan="16"></td><td colspan="16">CEV</td></tr>
<tr><td>CDW14</td><td colspan="32">LBA Location</td></tr>
<tr><td>CDW15</td><td colspan="32">LBA Location Version</td></tr>
</table>

FMS the Future of Memory and Storage

Conclusion and Future Outlook

- Conclusion:
    - THMB proves DRAM is not strictly required for high random read performance.
    - CHMB overhead is largely eliminated in the read path, closing the gap with DRAM SSDs.
- What's Next:
    - Broader adoption in next-gen DRAMless SSD architectures for cost-effective, high-performance solutions.
- THMB provides a transparent, efficient, and cost-saving alternative to DRAM for read-centric workloads.

Thank You!