Improving Flash Storage Performance by Caching Address Mapping Table in Host Memory

Wookhan Jeong, Yongmyung Lee, Hyunsoo Cho, Jaegyu Lee, Songho Yoon, Jooyoung Hwang, and Donggi Lee

2017.07.11
Presented at USENIX Hotstorage by

Joo-Young Hwang
(jooyoung.hwang@samsung.com)

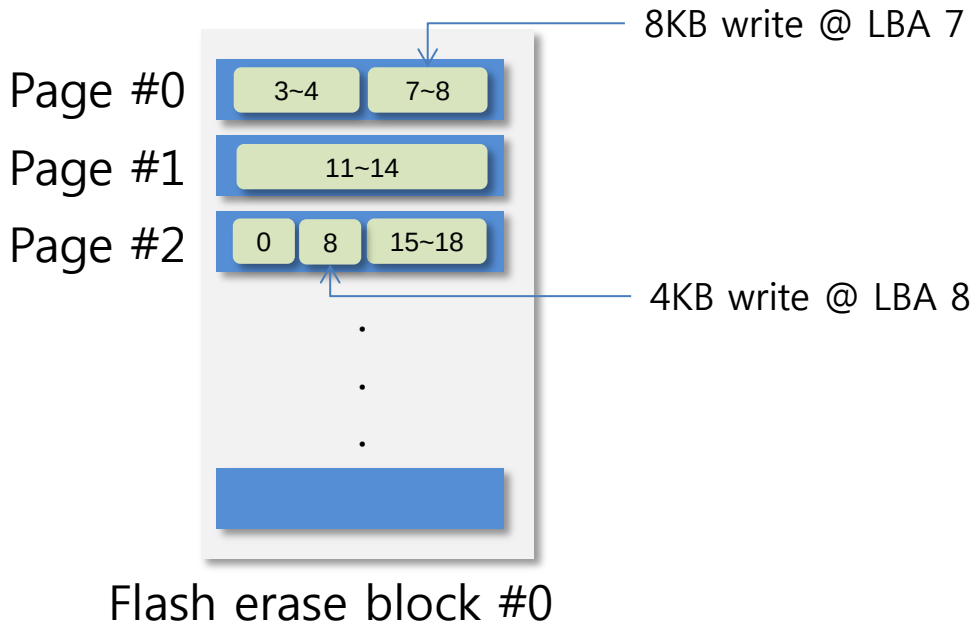
Problem Definition

- Mobile apps are random read performance hungry.

- Bottlenecks of random read in mobile storage
  - Limited parallelism (due to smaller density than desktop SSD)
  - L2P metadata (due to constraints on form factor/power consumption/cost)

What is FTL's L2P Metadata?

- L2P: Logical to physical address translation



L2P table

LBA	EB	Page	Offset
0	0	2	0
1			
2			
3	0	0	0
4	0	0	1
5			
6			
7	0	0	2
8	0	0	3
…			

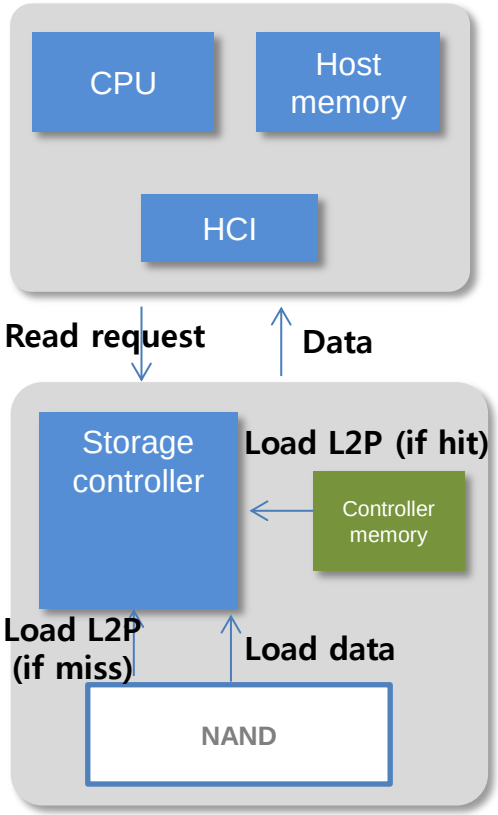
L2P Metadata Size Issue

- 1 L2P entry: 4Bytes (for 4KB logical block)
- For 128GB storage, total L2P size is 128MB which is too large to keep in controller memory.

On-Demand L2P Loading

- Loads a proper L2P page on demand.
- Performs well for reads with good locality.

- For random reads, L2P loading occurs more.
  - 1 L2P page (16KB) may contain 4K entries, and covers 16MB logical block address range.



Mobile workload pattern

- QD1 random reads

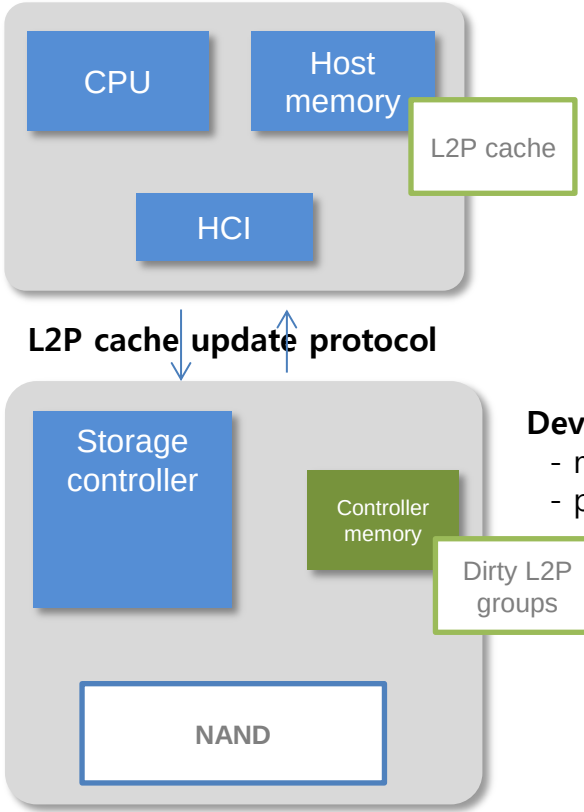
- Prediction and L2P prefetching?

Our Approach

- HPB (Host-aware Performance Booster): Collaboration between host and device

- In essence,
  - Cache L2P in host memory,
  - Host driver includes L2P in I/O request to avoid L2P loading from flash.

Overview

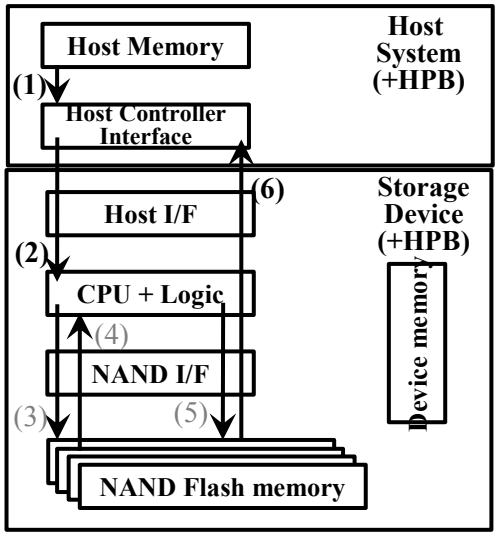


Host-side L2P Cache
- device-provided L2P bookkeeping
- include L2P per read request

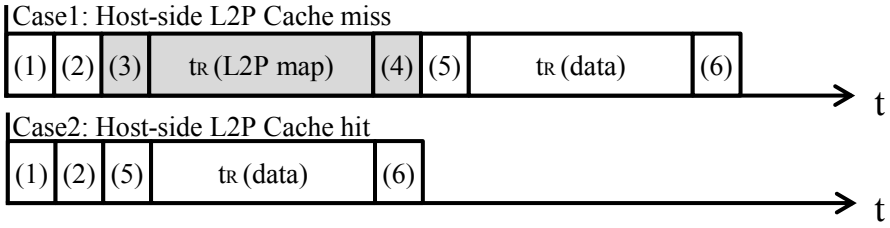
Device-side L2P Manager
- maintains dirty groups
- provide L2P

Verify Host-provided L2P
- authorized information? (detect tampering)
- up-to-date? (detect old information)

Read Request Processing in HPB

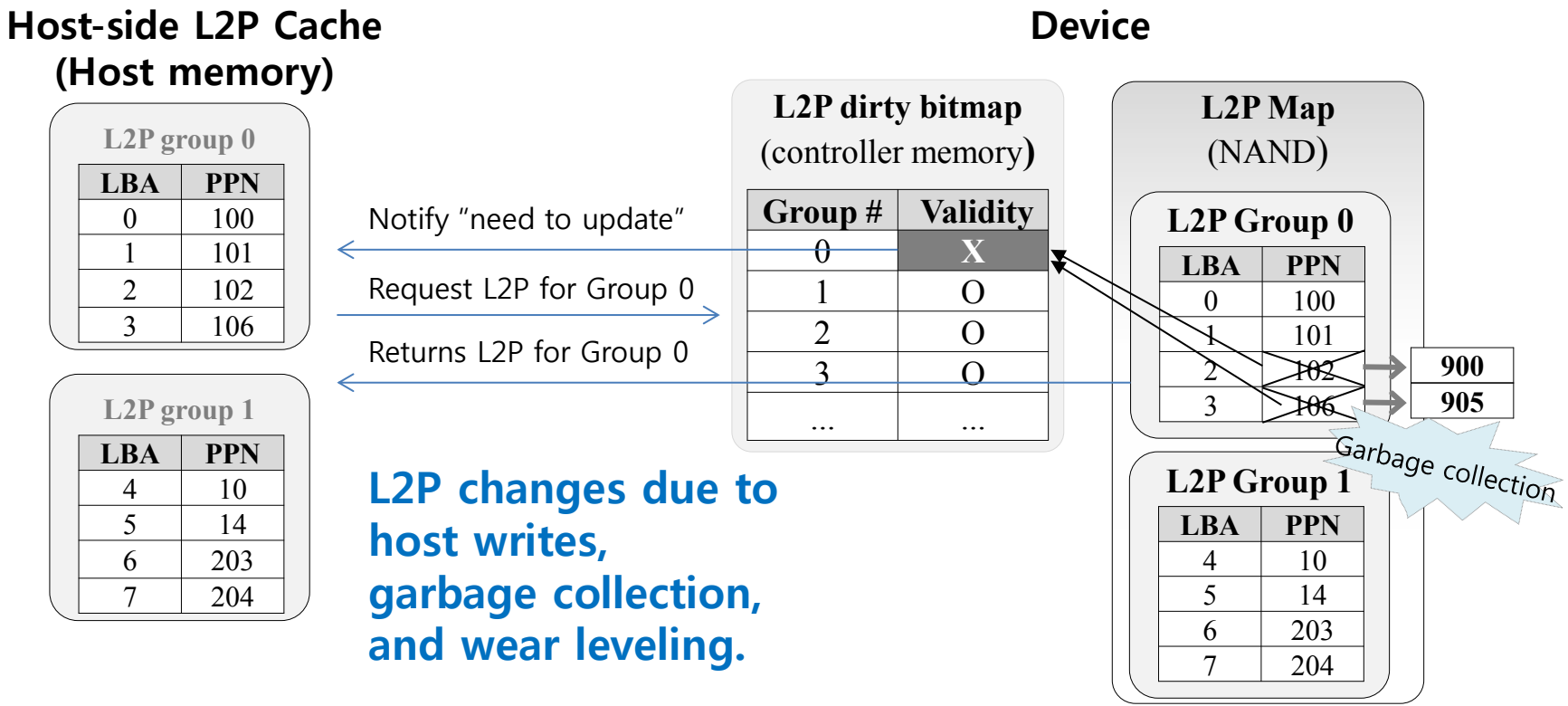


(1) Read L2P entry
(2) Fetch read command
(3) Request L2P entry
(4) Read L2P entry
(5) Request user data
(6) Transfer user data



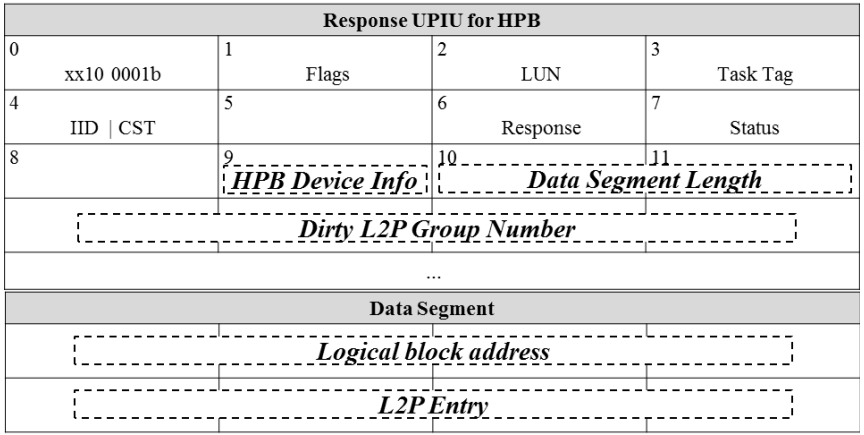
tR : NAND page read latency

L2P Cache Updates



L2P Cache Updates (cont'd)

- Two ways to update the cache
  - Host initiated: host issues commands to fetch L2P of a group.
    - Device notifies host of dirty group in response packet.
  - Device initiated: device piggybacks L2P in response packets.

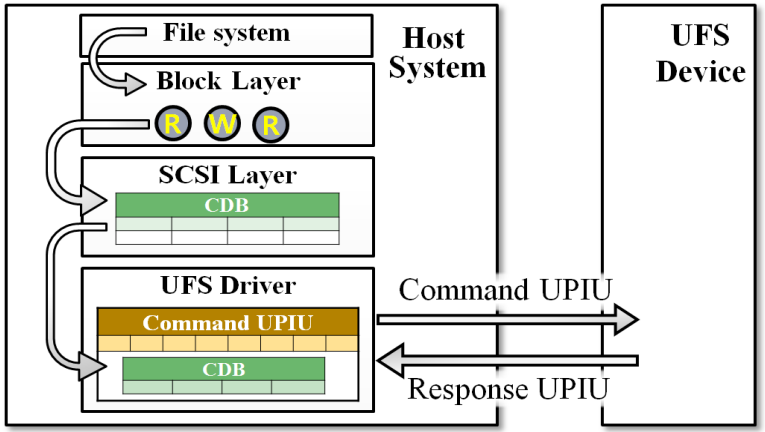


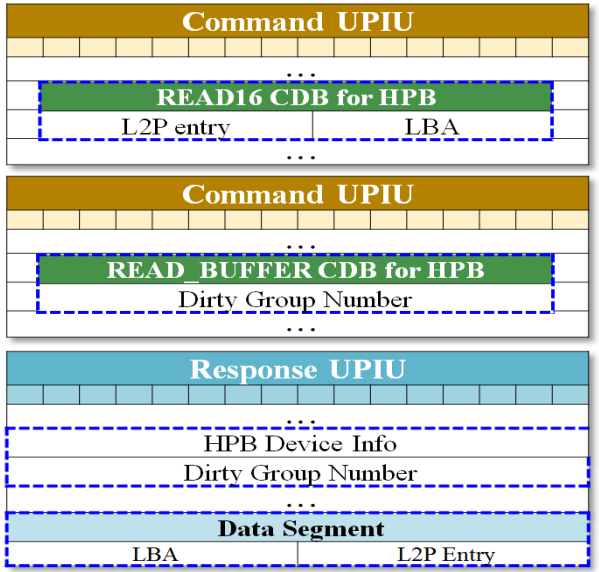
Response UPIU for HPB			
0 xx10 0001b	1 Flags	2 LUN	3 Task Tag
4 IID | CST	5	6 Response	7 Status
8	9 *HPB Device Info*	10	11 *Data Segment Length*
Dirty L2P Group Number			
...			

Data Segment
Logical block address
L2P Entry

Implementation in UFS

- UFS (Universal Flash Storage)
  - Successor of eMMC, shipped in smartphones since 2015.
  - Layered architecture, uses SCSI command sets
  - UFS 2.0 600MB/s per lane, max 2 lanes





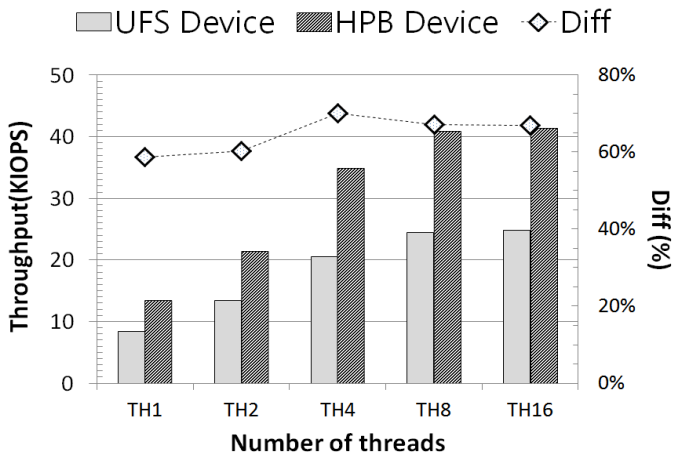
Delivering L2P Hints

- Modify READ(16) commands to include L2P.
  - READ(16): 8Bytes LBA, 4Bytes Transfer Length
  - Modified READ(16): **4Bytes L2P**, 4Bytes LBA, 4 Bytes Transfer Length

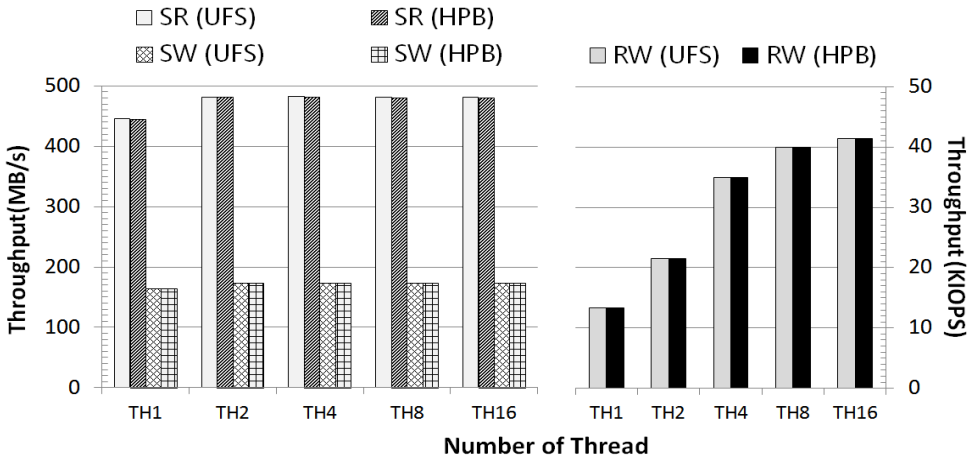
READ16 CDB for HPB								
B \ b	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	OPERATION CODE (88h)							
1	PDPROTECT			DPO	FUA	RSV	FUANV	***HPB***
2	***L2P entry***							
...								
5								
6	***Logical block address***							
...								
9								
10	***Transfer Length***							
...								
13								
14								

Experimental Results

- 59~67% random read performance improvements
- Little or no effect on sequential R/W and random write performances



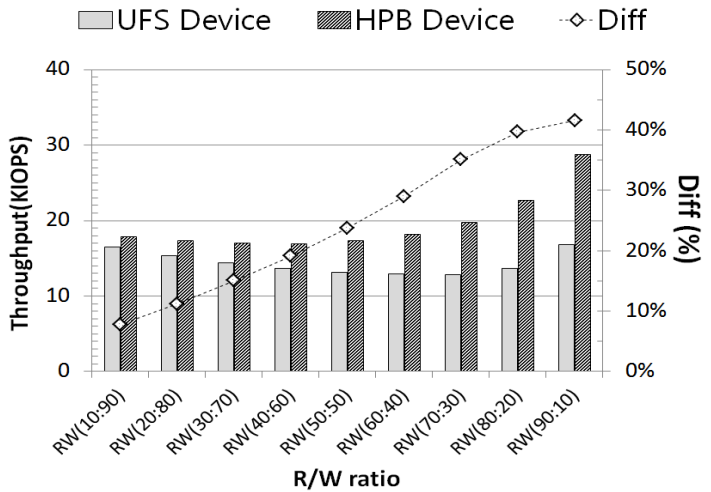
tiobench 4KB RR (Random Read) performance

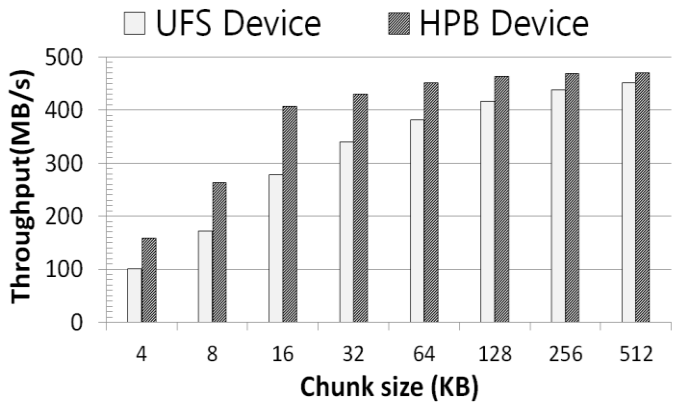


tiobench SR(Sequential Read), SW(Sequential Write), RW(Random Write) performance.

Experimental Results (cont'd)

- HPB shows better performance in overall R:W mix ratio and chunk sizes (4 ~ 512KB).





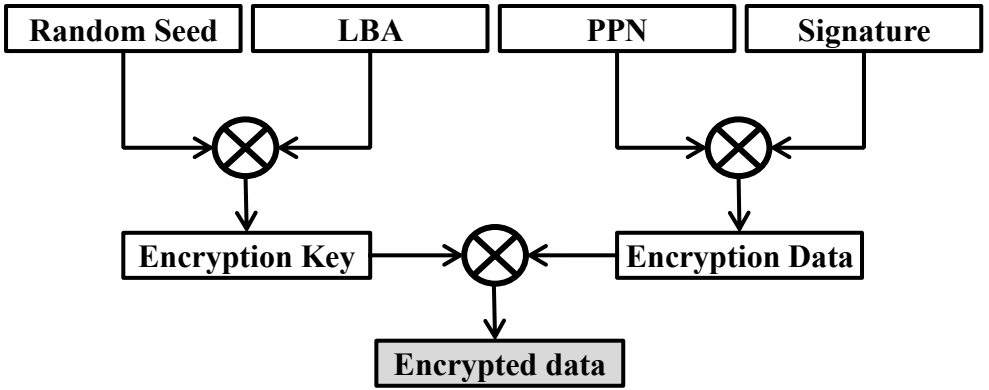
Mixed pattern performance (4KB record size, 1GB I/O issue, 16 threads).
In RW(x:y), x is read portion and y is write portion.

Further Works

- Standardization
  - EHS(Extra Header Segment) in UFS 3.0
    - Host can deliver L2P for a chunk that is physically fragmented.
- Host-side memory management
  - Deal with host memory pressure
- More performance benchmark
  - Benefits in phone user scenarios
- L2P verification implementation

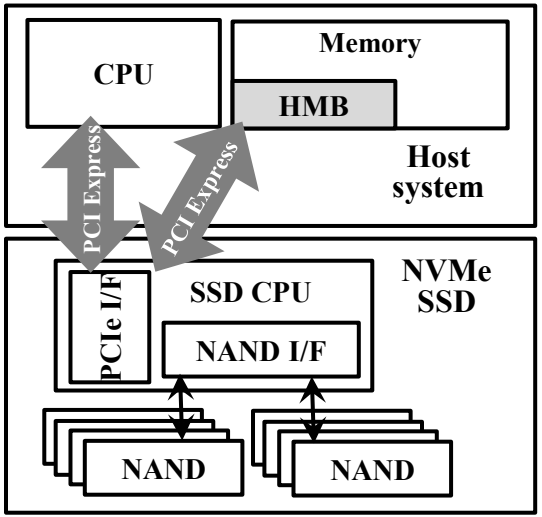
L2P Verification

- Check if a host-provided L2P has not been tampered.
- Requires encrypt/decrypt hardware support to avoid overhead.



Related Works

- Other approaches
  - Interconnects that allows device to access host memory directly.
    : PCIe/NVMe provides HMB (Host Memory Buffer)
    : UFS UME (Unified Memory Extension)
  - Static allocation of host memory
  - Latency of accessing host memory from device is in critical path.



Summary

- HPB (Host Performance Booster)
  - Improve random read performance by caching L2P map in host memory and delivering L2P hint when sending I/O request.
- HPB implementation in UFS
  - Modified READ(16) to piggyback L2P hints.
- Improved random read performance by 59~67%

THE NEXT CREATION STARTS HERE

Placing **memory** at the forefront of future innovation and creative IT life