Host Performance Booster (HPB):

Introduction and Current Mainline Support Status

Jaemyung Lee, Alim Akhtar, Samsung

Agenda

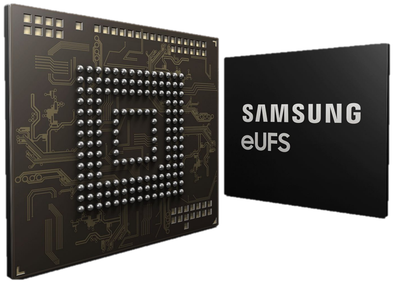
- Technical Backgrounds
- Introduce HPB
- Deeper Implementation
- Performance Improvement
- Current Status
- Conclusion

Technical Backgrounds:

Brief Information about UFS

What is UFS?

- Universal Flash Storage(UFS)
  - Simple, high performance, mass storage device with a serial interface
  - Provides Higher performance & Lower power consumption
  - Widely used in commercial embedded products(e.g., smartphone)
- Linux UFS Subsystem
  - Subsystem that supports UFS storage devices
  - Implemented under the SCSI Subsystem



Performance: eMMC vs UFS

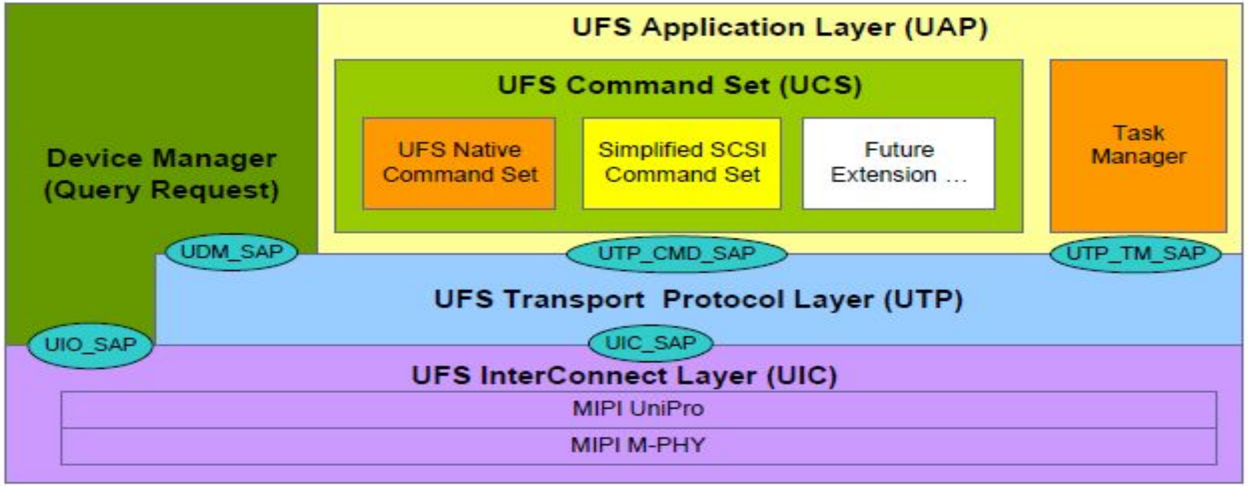
- Performance Comparison with eMMC

Products	Interface	Sequential Read	Sequential Write	Random Read	Rand Write
eMMC	**eMMC 4.5**	140MB/s	50MB/s	7,000 IOPS	2,000 IOPS
	eMMC 5.0	250MB/s	90MB/s	7,000 IOPS	7,000 IOPS
	eMMC 5.1	250MB/s	125MB/s	11,000 IOPS	13,000 IOPS
UFS	**128GB eUFS 2.0**	350MB/s	150MB/s	19,000 IOPS	14,000 IOPS
	256GB eUFS 2.0	850MB/s	260MB/s	45,000 IOPS	40,000 IOPS
	256GB UFS Card	530MB/s	170MB/s	40,000 IOPS	35,000 IOPS
	512GB eUFS 2.1	860MB/s	255MB/s	42,000 IOPS	42,000 IOPS
	1TB eUFS 2.1	1000MB/s	260MB/s	58,000 IOPS	50,000 IOPS
	512GB eUFS 3.1	2100MB/s	410MB/s	63,000 IOPS	68,000 IOPS

* https://news.samsung.com/global/samsung-electronics-doubling-current-smartphone-storage-speed-as-it-begins-mass-production-of-first-512gb-eufs-3-0

Architectural Overview

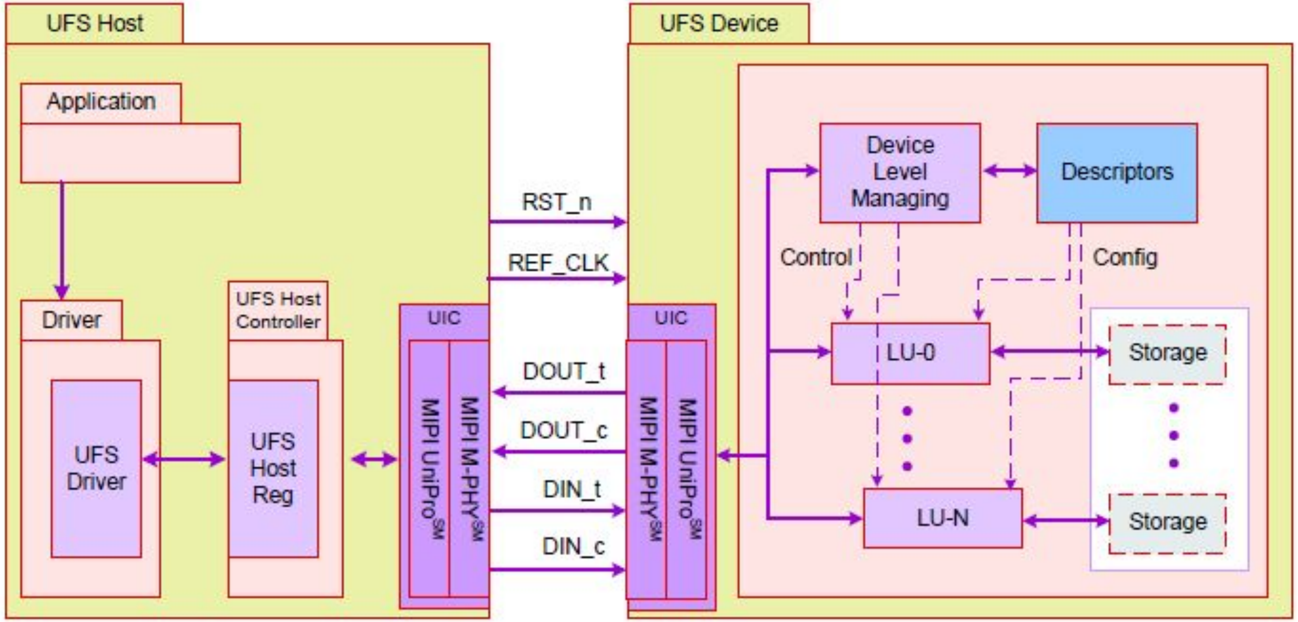
- UFS Top Level Architecture
  - UFS communication is a layered communication architecture
  - Based of SCSI SAM architectural model [SAM]



* JESD220E UFS3.1, JEDEC

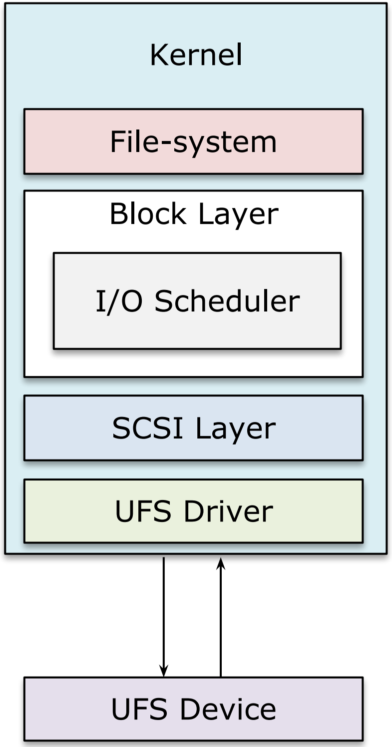
Architectural Overview

- UFS System Model



* JESD220E UFS3.1, JEDEC

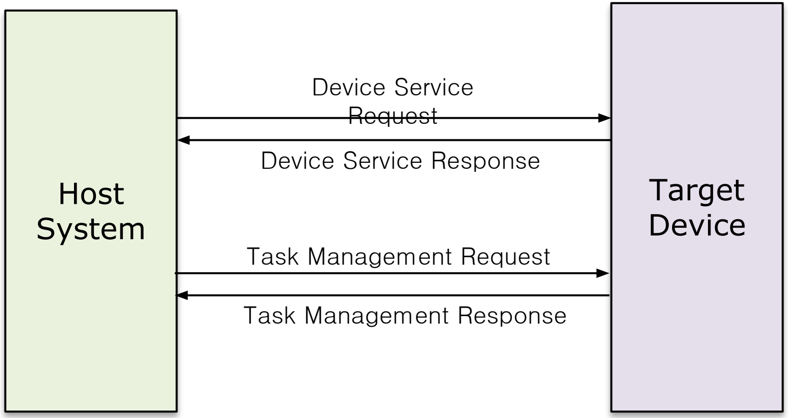
UFS Subsystem



- UFS Subsystem Implementation
  - UFS is implemented under SCSI layer
  - UFS driver uses the SCSI command set
- Location of UFS Subsystem in Linux
  - UFS directory: drivers/scsi/ufs/
  - Core drivers: drivers/scsi/ufshcd.{ch}
  - Platforms: drivers/scsi/ufshcd-pltfrm.{ch}
  - Controller specifics: driver/scsi/ufs-*.{ch}
    - e.g., ufs-exynos.{ch}

Transactions in UFS

- UFS Transport Protocol(UTP) Layer
  - UTP uses a SCSI Architectural model(SAM)
  - **Client-Server** or **Request-Response** model
  - UFS transactions consist of packets called **UFS Protocol Information Unit(UPIU)**



SAM client-server model

Transactions in UFS

- UFS Protocol Information Unit(UPIU)
  - All UPIU consists of a single constant 12 bytes header segment
  - Transaction specific segment
  - Possibly one or more extended header segments
  - Zero or more data segments

Table 10.1 — UPIU Transaction Codes

Initiator To Target	Transaction Code	Target to Initiator	Transaction Code
NOP OUT	00 0000b	NOP IN	10 0000b
COMMAND	00 0001b	RESPONSE	10 0001b
DATA OUT	00 0010b	DATA IN	10 0010b
TASK MANAGEMENT REQUEST	00 0100b	TASK MANAGEMENT RESPONSE	10 0100b
Reserved	01 0001b	READY TO TRANSFER	11 0001b
QUERY REQUEST	01 0110b	QUERY RESPONSE	11 0110b
Reserved	01 1111b	REJECT UPIU	11 1111b
Reserved	Others	Reserved	Others

NOTE 1 Bit 5 of the Transaction Code indicates the direction of flow and the originator of the UPIU: when equal '0' the originator is the Initiator device, when equal '1' the originator is the Target device.

* JESD220E UFS3.1, JEDEC

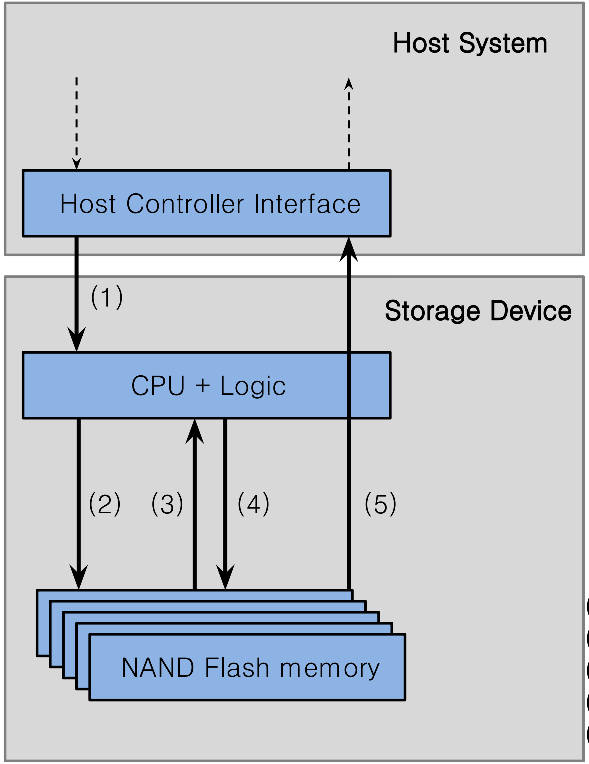
Scope of Improvement: The Read Latency

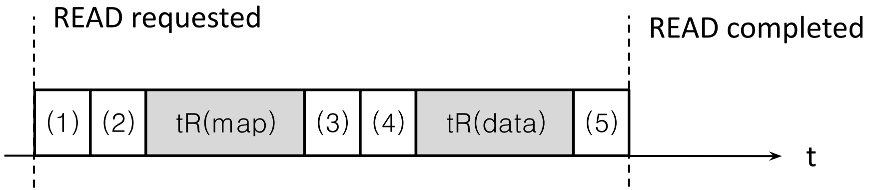
- UFS is a Flash Memory Storage
  - NAND Flash device uses Flash Translation Layer (FTL)
  - To translate logical address of I/O request to flash memory physical address
  - Logical Physical (L2P) mapping entries are managed by FTL
  - Device must read the mapping entry first to reach the actual data
  - This mapping table is maintained in the NAND flash memory.

- The 'Read Latency'
  - Normally UFS devices are having SRAM to cache these entries
  - Because of high cost of SRAM, it only save partial data of whole entries

Scope of Improvement: The Read Latency

- Read Latency of UFS Device





-> Flash memory access latency **doubled**

* when the mapping entries are not cached in SRAM

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

Introduce HPB:

What is the Host Performance Booster?

What is HPB?

UFS
Universal
Flash
Storage

+

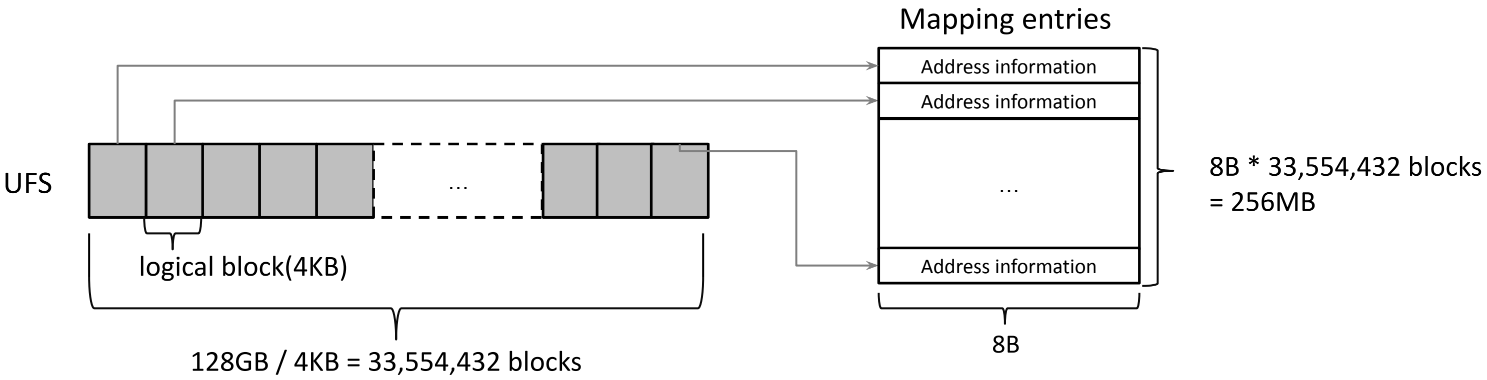
HPB
Host
Performance
Booster

The Host Performance Booster is an **Extension Feature** of **UFS Subsystem**:
to Improve the overall performance through reducing the read latency

* UNIVERSAL FLASH STORAGE (UFS) HOST PERFORMANCE BOOSTER (HPB) EXTENSION, VERSION 2.0, JEDEC
https://www.jedec.org/document_search?search_api_views_fulltext=jesd220-3

Concept of HPB

- What exactly HPB does?
  - HPB **caches** the L2P mapping entries in the **Host memory**
  - The capacity of Host memory is **BIG enough** to save the mapping entries
  - Accessing the Host memory is a lot faster than accessing NAND

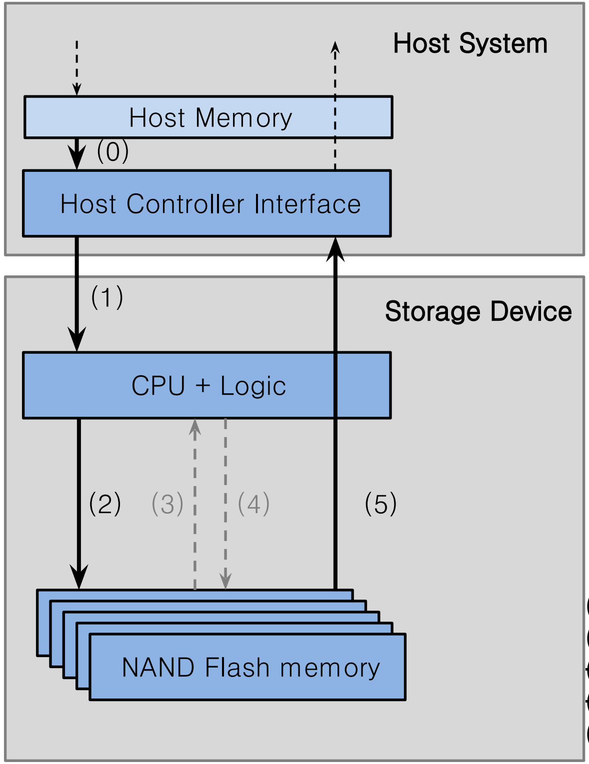


 256MB needed for whole **128GB** mapping entries

* Assuming the logical block size of UFS device is 4KB

Concept of HPB

- Read Latency of UFS Device **with HPB**



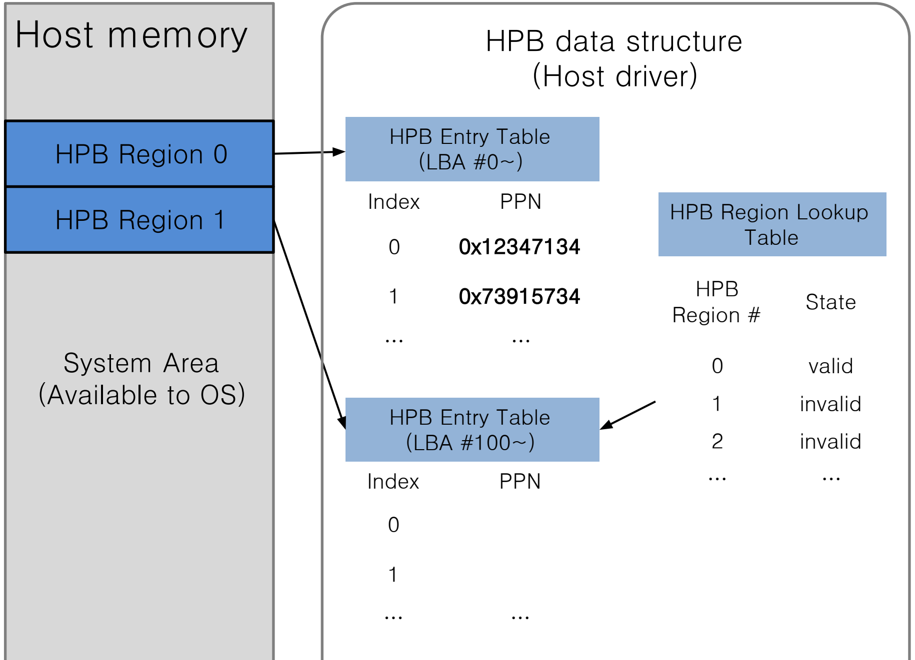
☐ Read latency **reduced**

* When the mapping entries are not cached in SRAM

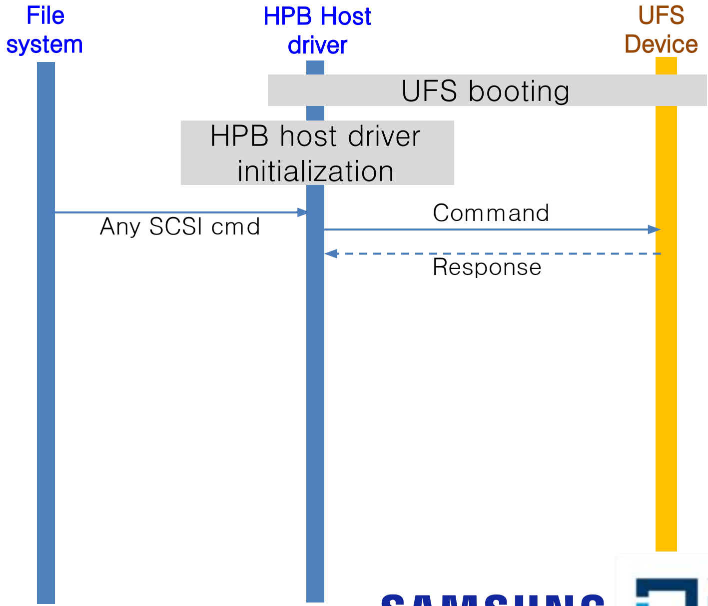
(0) Read Cached L2P entry
(1) Fetch read command
(2) Request L2P entry
~~(3) Read L2P entry~~
~~(4) Request user data~~
(5) Transfer user data

Overall Behavior

- Caching Requested

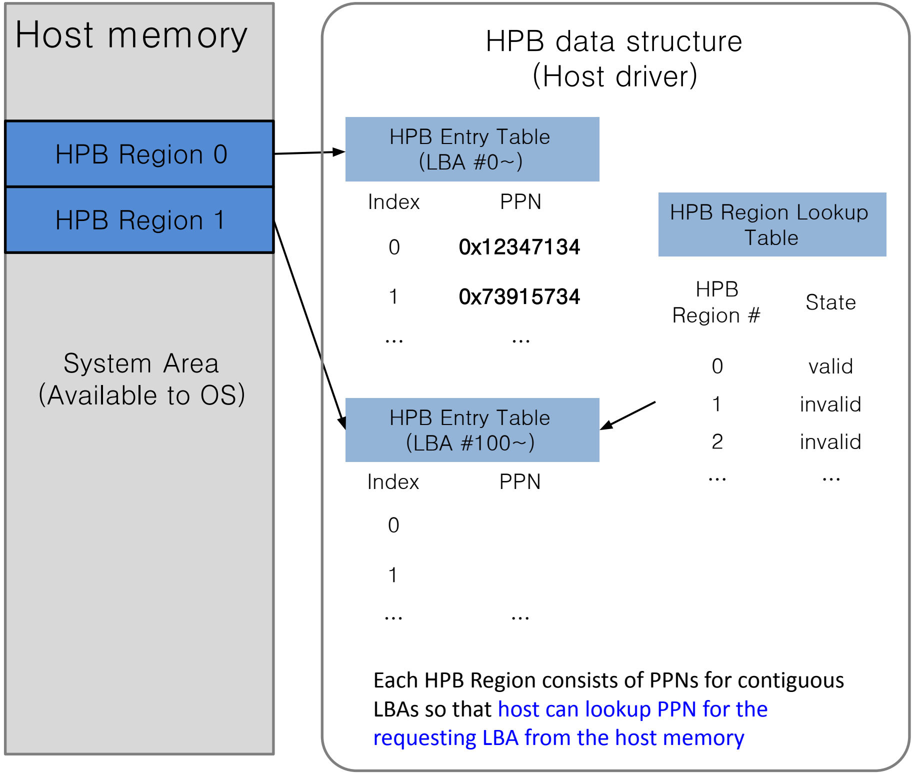


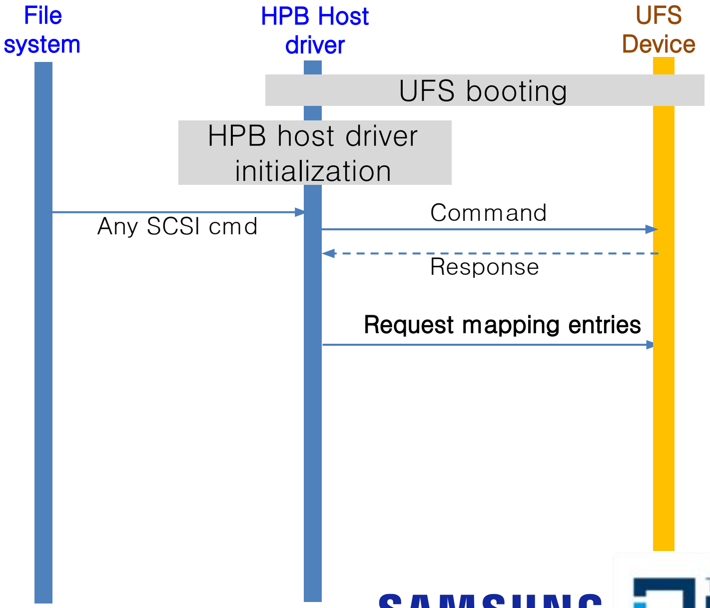
Each HPB Region consists of PPNs for contiguous LBAs so that host can lookup PPN for the requesting LBA from the host memory



Overall Behavior

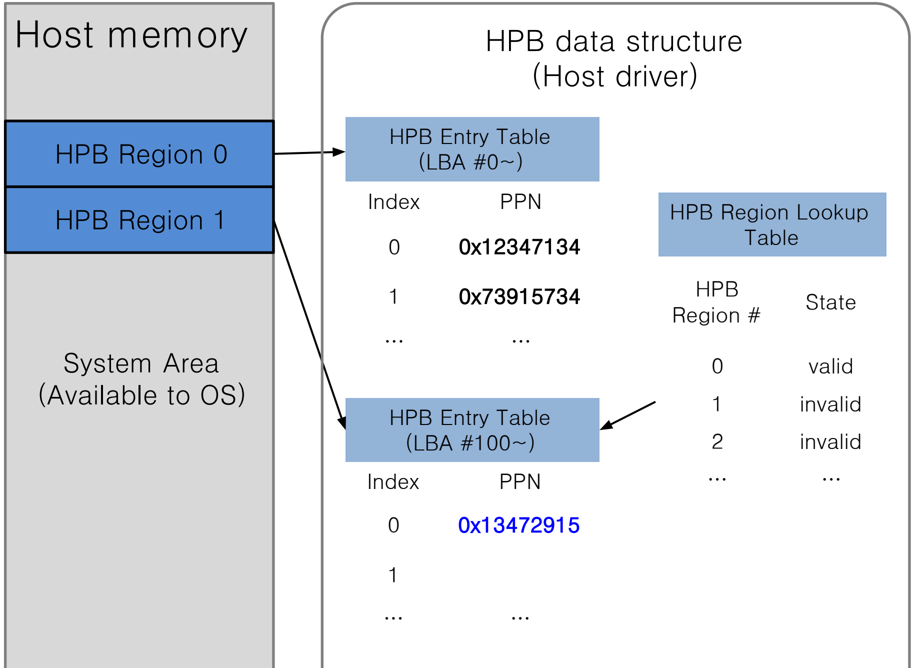
- Caching Mapping Entries



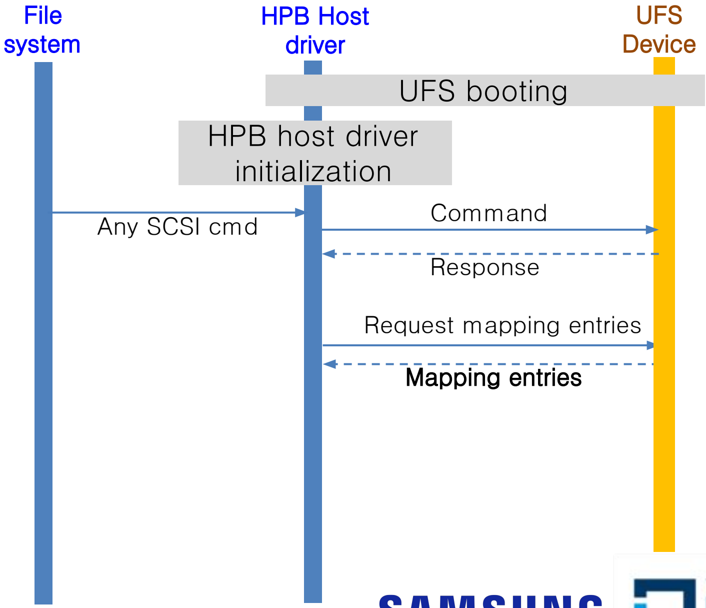


Overall Behavior

- Caching Mapping Entries

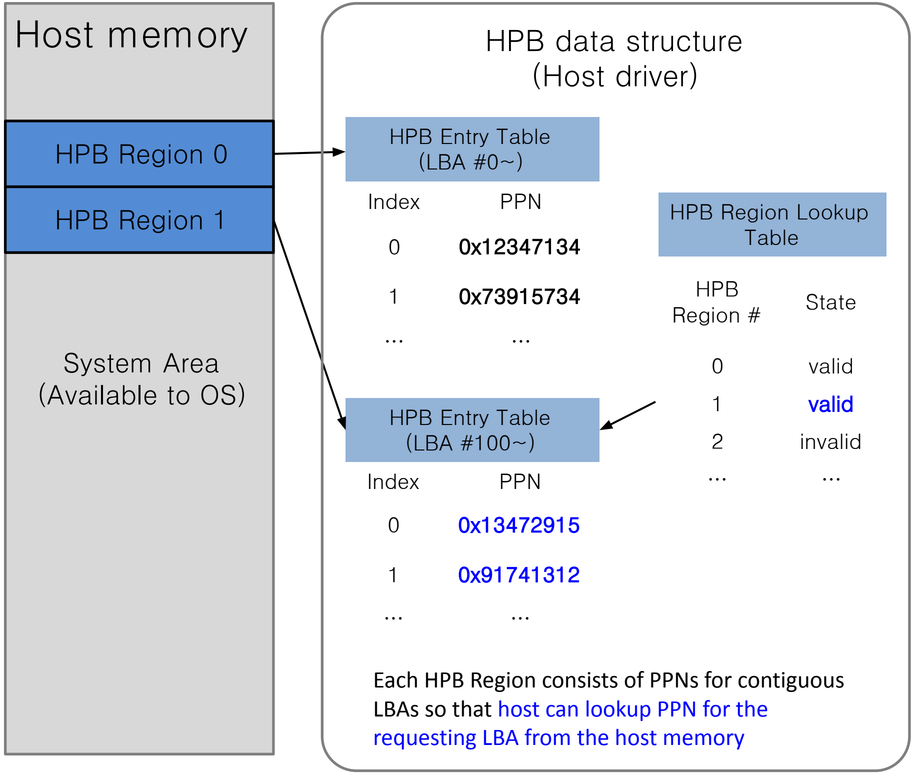


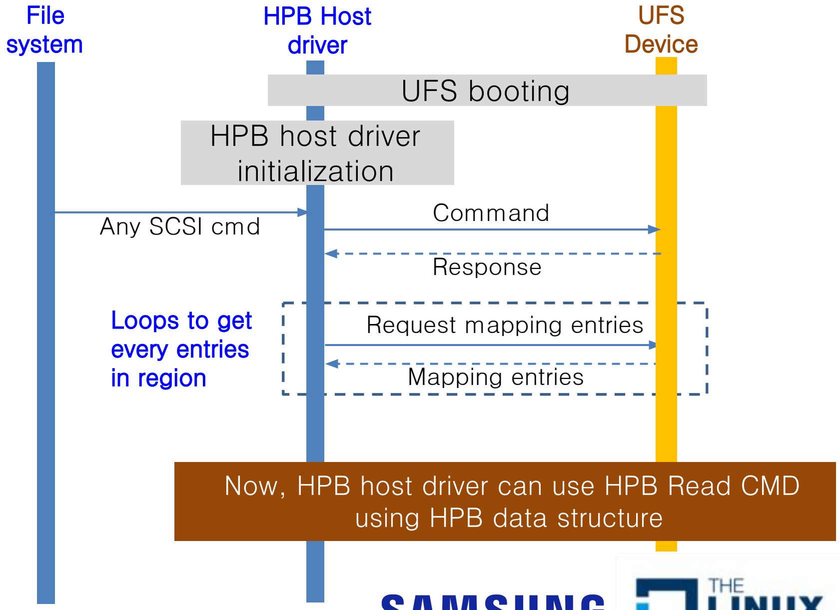
Each HPB Region consists of PPNs for contiguous LBAs so that host can lookup PPN for the requesting LBA from the host memory



Overall Behavior

- Caching Mapping Entries

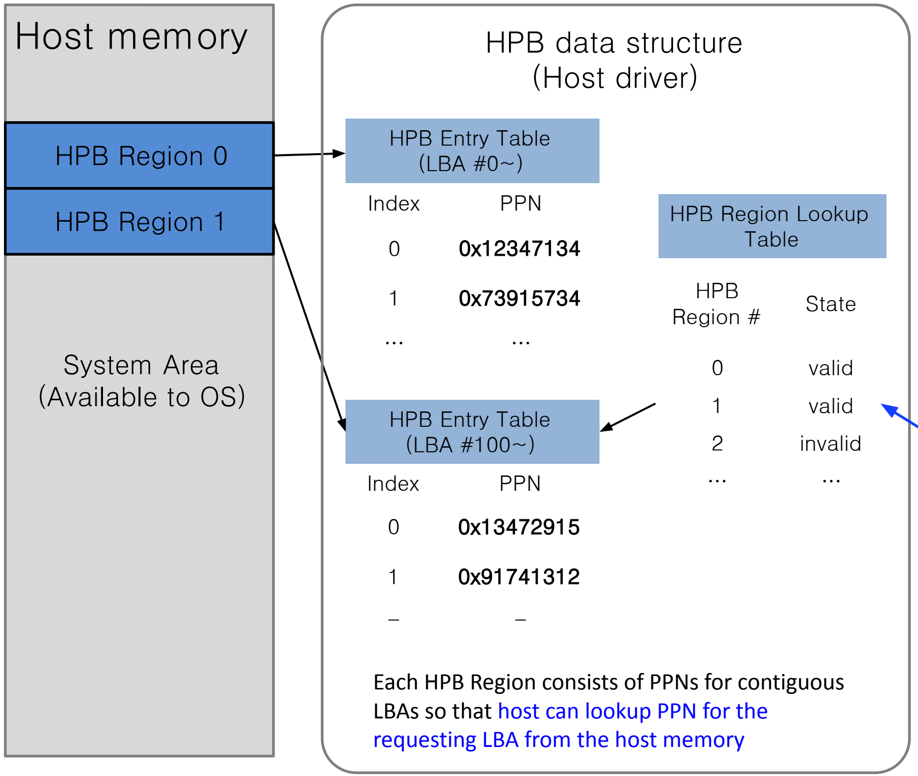


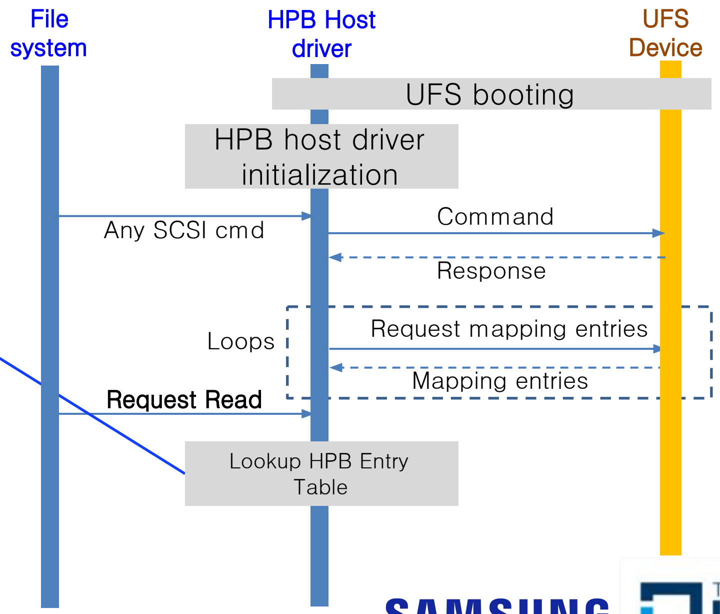


SAMSUNG THE LINUX FOUNDATION

Overall Behavior

- Sending Command

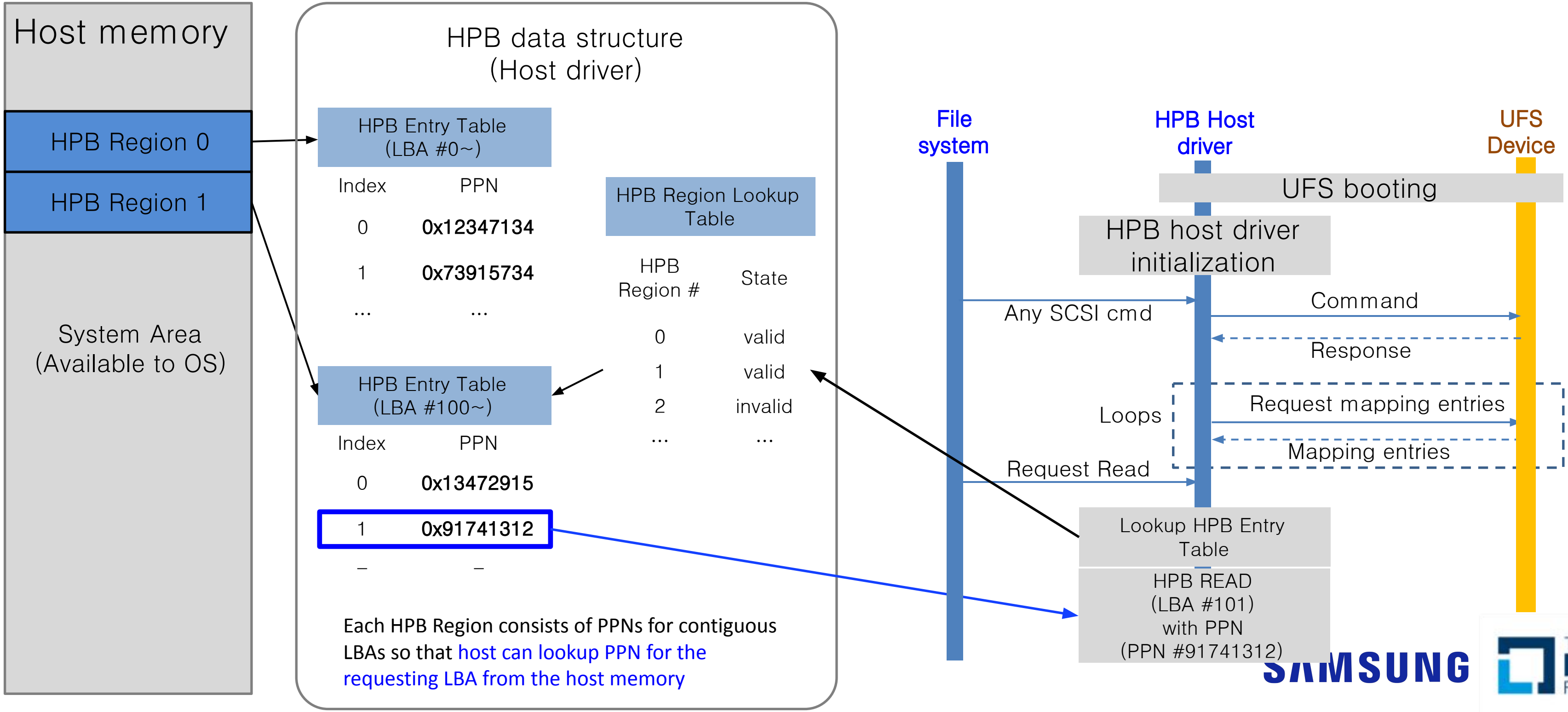




SAMSUNG THE LINUX FOUNDATION

Overall Behavior

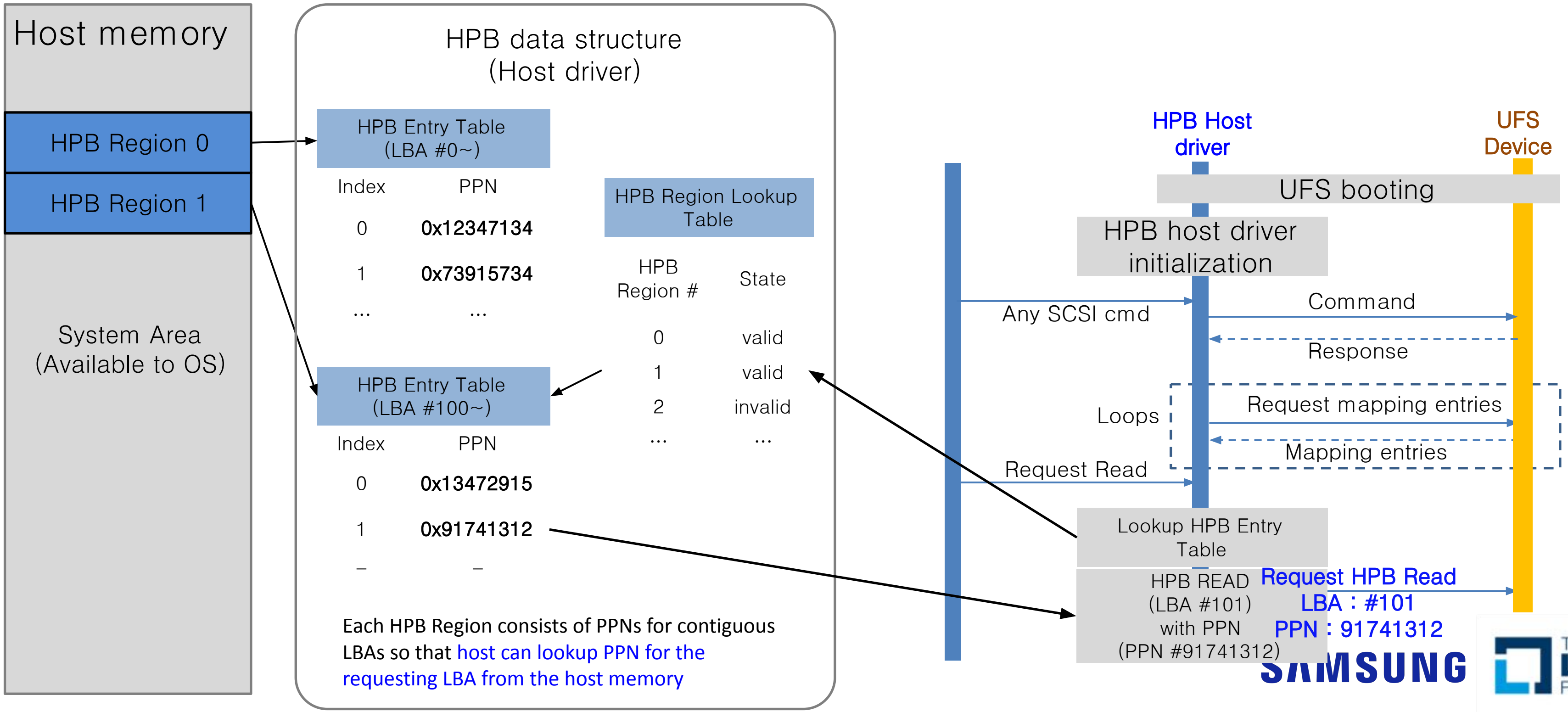
- Sending Command



SAMSUNG THE LINUX FOUNDATION

Overall Behavior

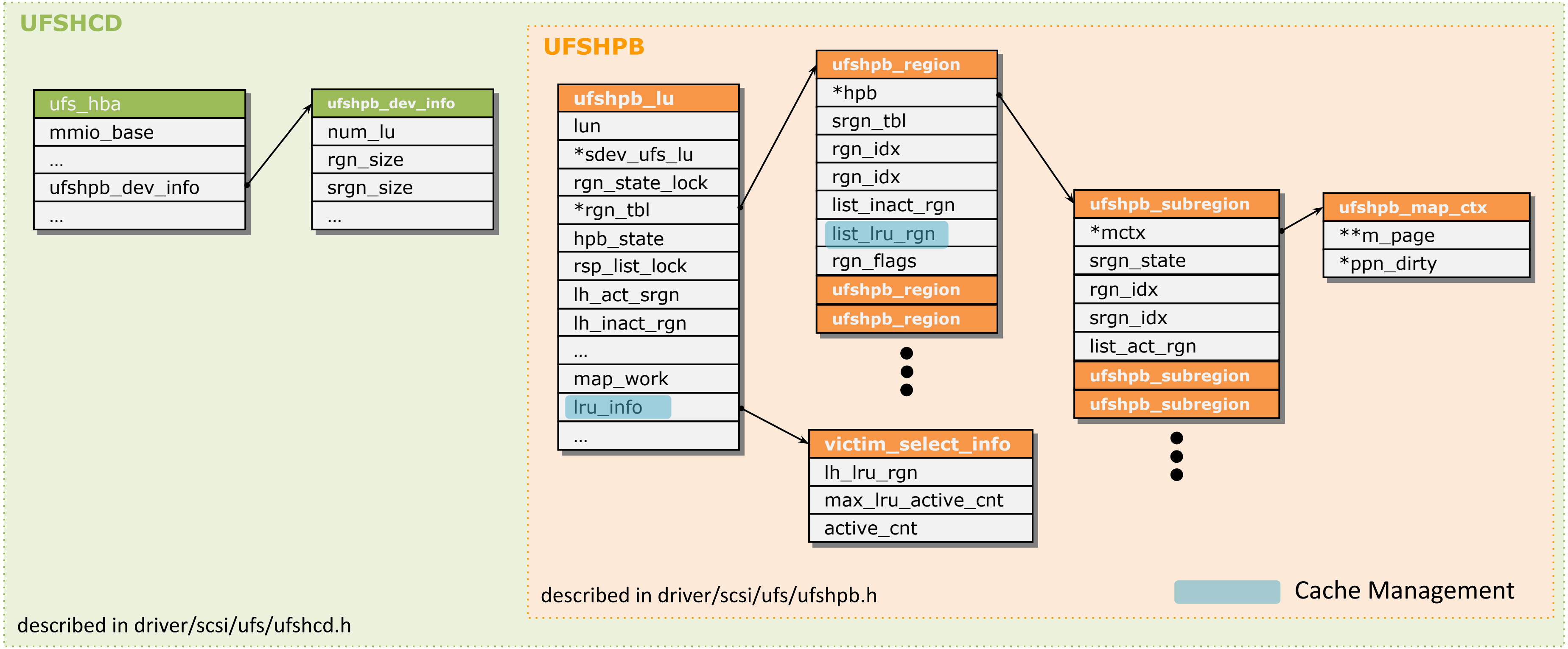
- Sending Command



Deeper Implementation:

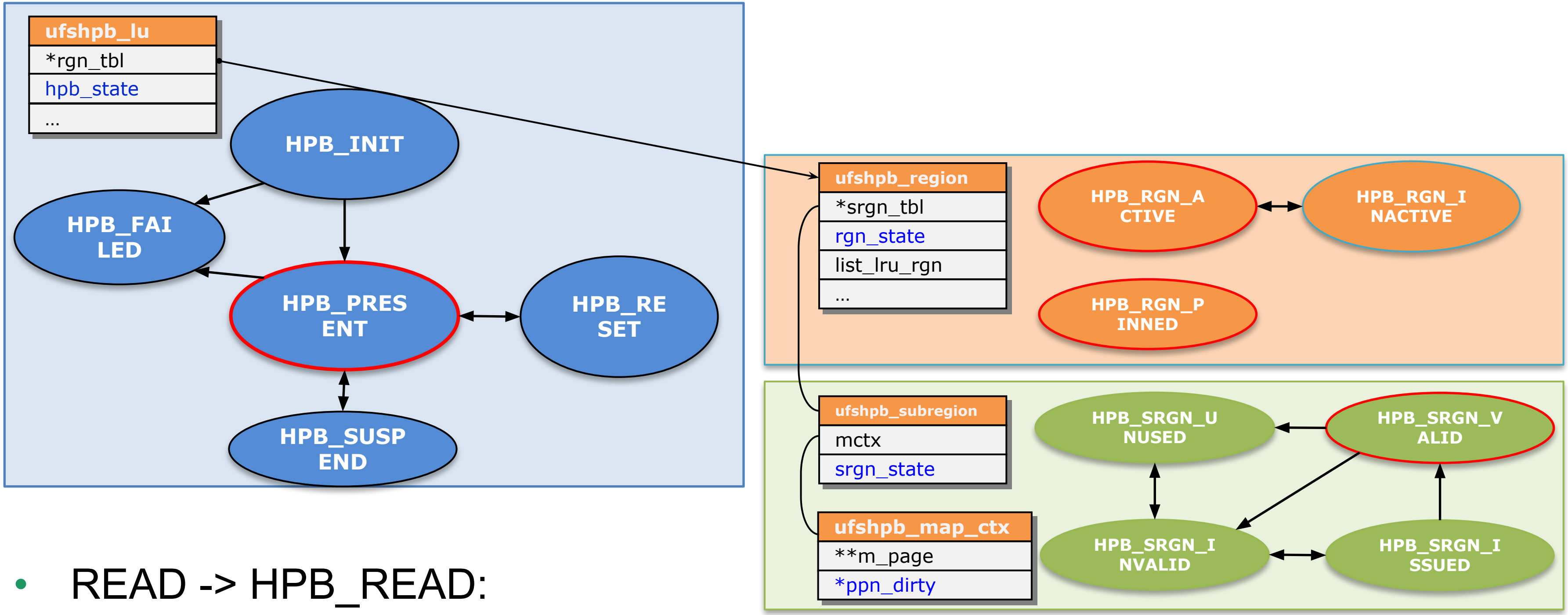
For further understanding

Data Structure



* Some of data structures of HPB is not included in the picture

State change of HPB



- READ -> HPB_READ:
  - **HPB_PRESENT** & **HPB_RGN_ACTIVE/PINNED** & **HPB_SRGN_CLEAN**
  - The cached entry is not **DIRTY**

Region Management

- Activate/Inactivate Information
  - Active/Inactivate region is informed by device through the region number
  - Information is received at each end of transaction through Response UPIU

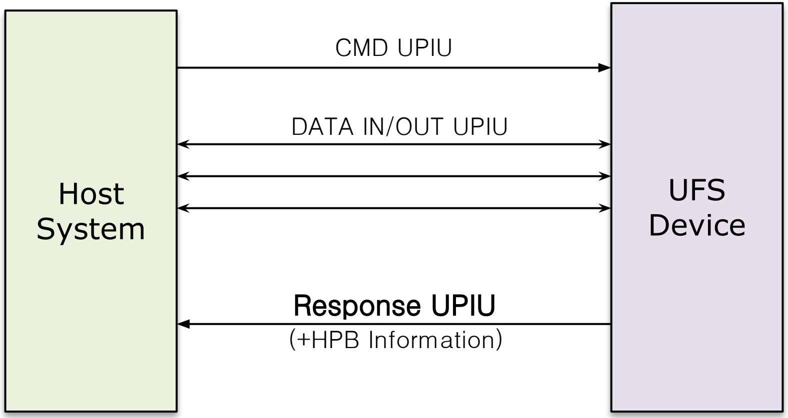


Table 6.9 — RESPONSE UPIU indicating HPB Active/Inactive Region and Sub-Region

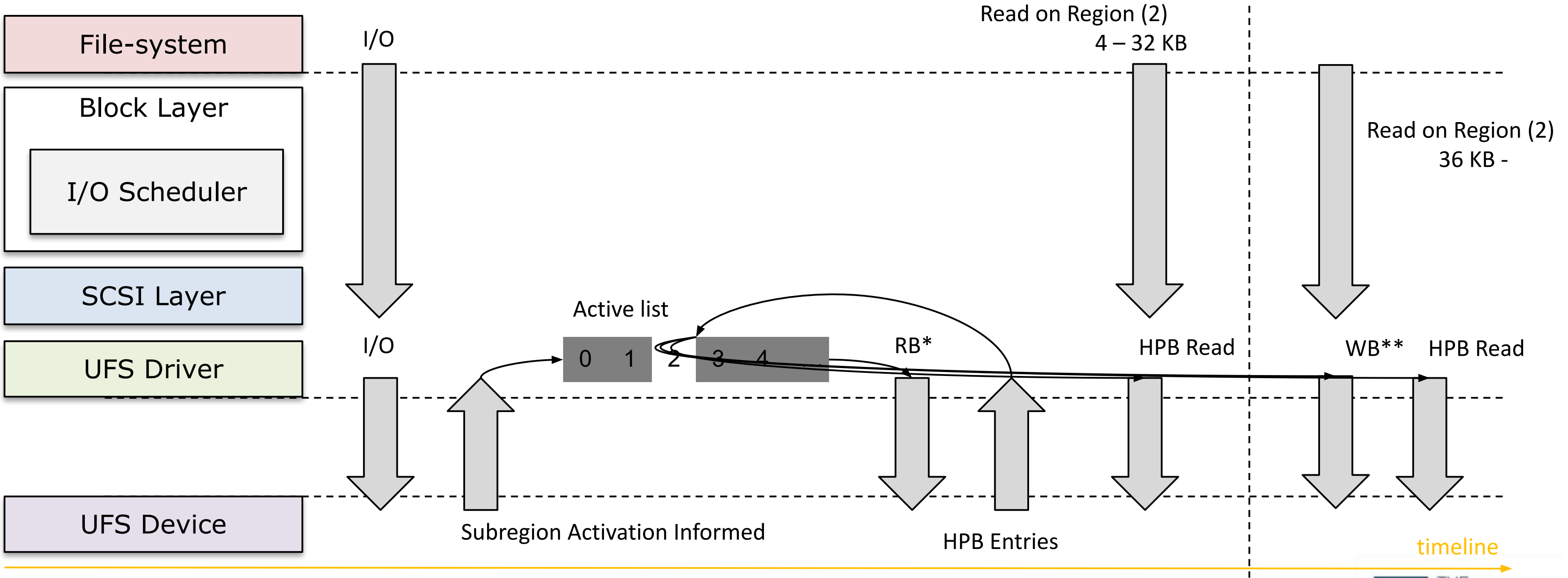
RESPONSE UPIU			
0 xx10 0001b	1 Flags	2 LUN	3 Task Tag
4 IID / Command Set Type	5 Reserved	6 Response	7 Status = Good (00h)
8 Total EHS Length (00h)	9 Device Information bit 1: HPB_UPDATE_ALERT = 1	10 (MSB) Data Segment Length (14h)	11 (LSB)
12 (MSB) Residual Transfer Count	13	14	15 (LSB)
16 Reserved	17	18	19
20 Reserved	21	22	23
24 Reserved	25	26	27
28 Reserved	29	30	31
Header E2ECRC (omit if HD=0)			
K (MSB) Sense Data Length (12h)	K+1 (LSB)	K+2 Descriptor Type (80h)	K+3 Additional Length (10h)
K+4 HPB Operation	K+5 LUN	K+6 Active HPB Count	K+7 Inactive HPB Count
K+8 (MSB) Active HPB Region 0	K+9 (LSB)	K+10 (MSB) HPB Sub-Region of [K+8 : K+9]	K+11 (LSB)
K+12 (MSB) Active HPB Region 1	K+13 (LSB)	K+14 (MSB) HPB Sub-Region of [K+12 : K+13]	K+15 (LSB)
K+16 (MSB) Inactive HPB Region 0	K+17 (LSB)	K+18 (MSB) Inactive HPB Region 1	K+19 (LSB)
Data E2ECRC (omit if DD=0)			

HPB Sense Data

* JESD220-3A, JEDEC

Region Management

- Activate Region
  - Active subregion is informed by Device

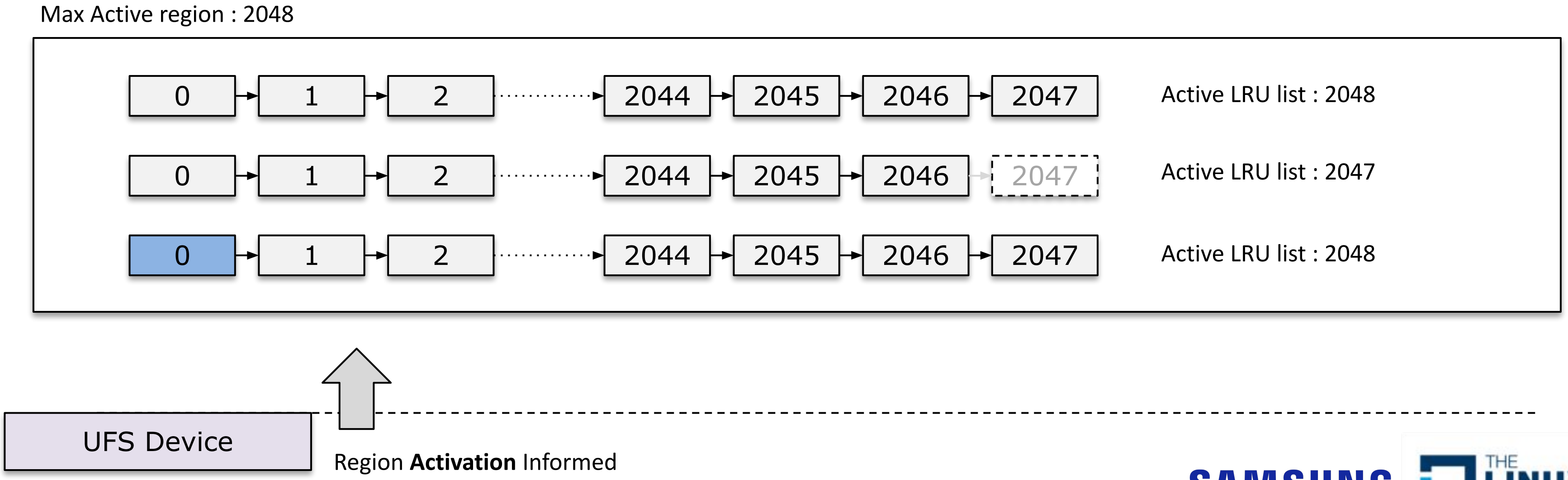


* RB: READ BUFFER command: used in HPB for requesting mapping entry

** WB: WRITE BUFFER command: used in HPB as prefetching command

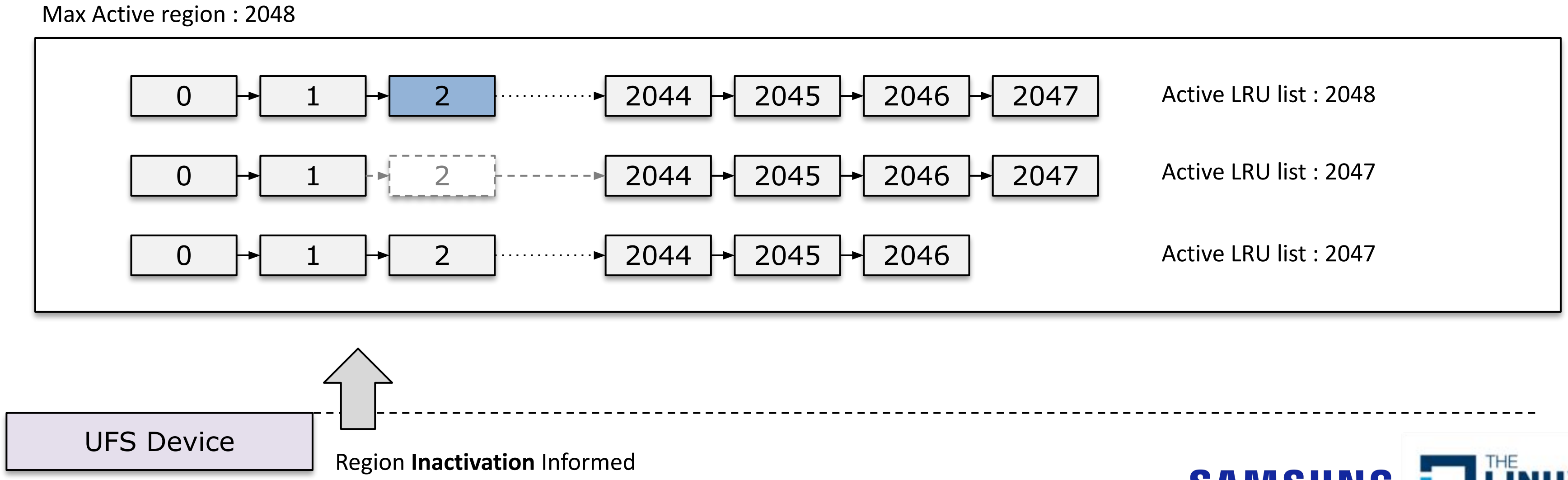
Region Management

- Inactivate Region – Victimized by LRU
  - Active regions are managed by LRU algorithm(Active LRU list)
    - If the list is full, the last used region has to be selected as **victim**
    - Then, selected victim will be **inactivated**
    - Total number of active region is not changed



Region Management

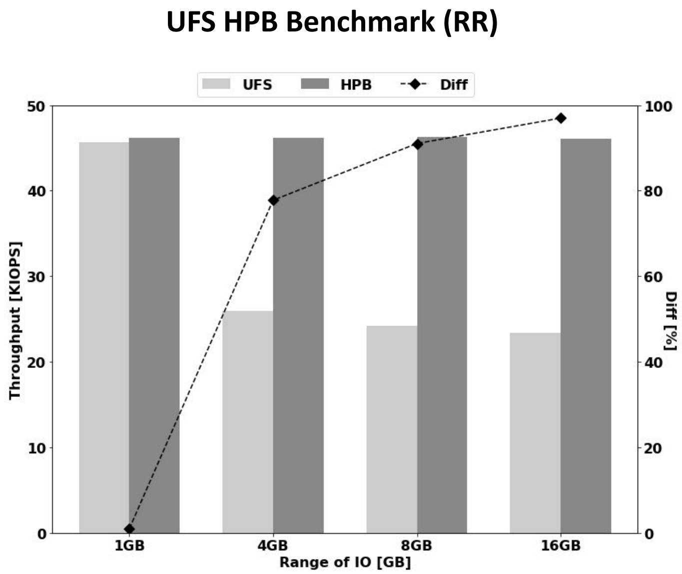
- Inactivate Region – Informed by Device
  - Also, device can inform the region which should be inactivated
    - HPB finds the region which is informed in Active LRU list
    - Informed region is **deleted directly** from list and **inactivated**
    - Total number of active region is reduced



Performance Improvement:

Is it work?

Benchmark & UX experience

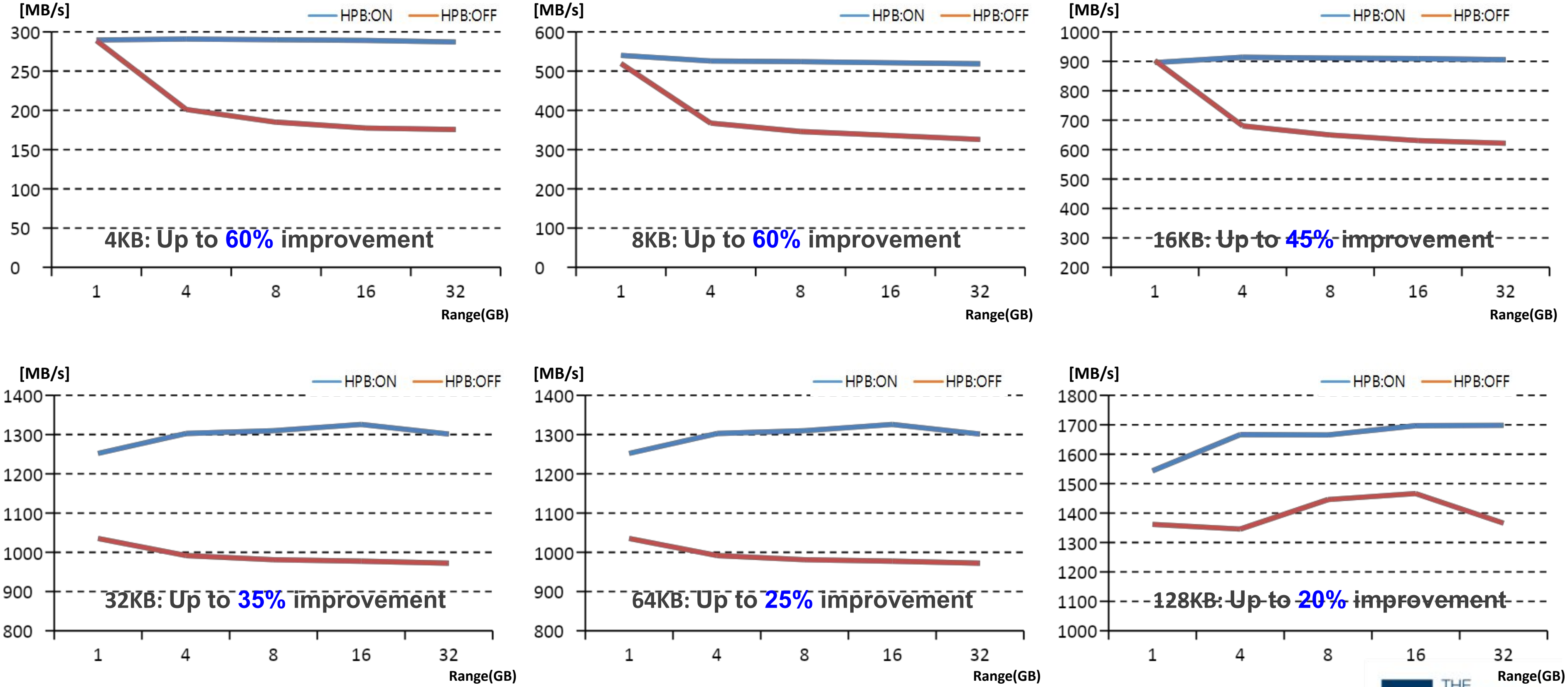


UFS HPB UX (App Launch Time)

CYCLE	UFS [S]	HPB [S]	DIFF [S]
1	272.4	264.9	7.5
2	250.4	248.2	2.2
3	226.2	215.6	10.6
4	230.6	214.8	15.8
5	232.0	218.1	13.9
6	231.9	212.6	19.3

* Measured with UFS 3.1 samples manufactured by Samsung

Chunk-Range Result



* Measured with UFS 3.1 samples manufactured by Samsung
** Measured through IOzone test: https://www.iozone.org/

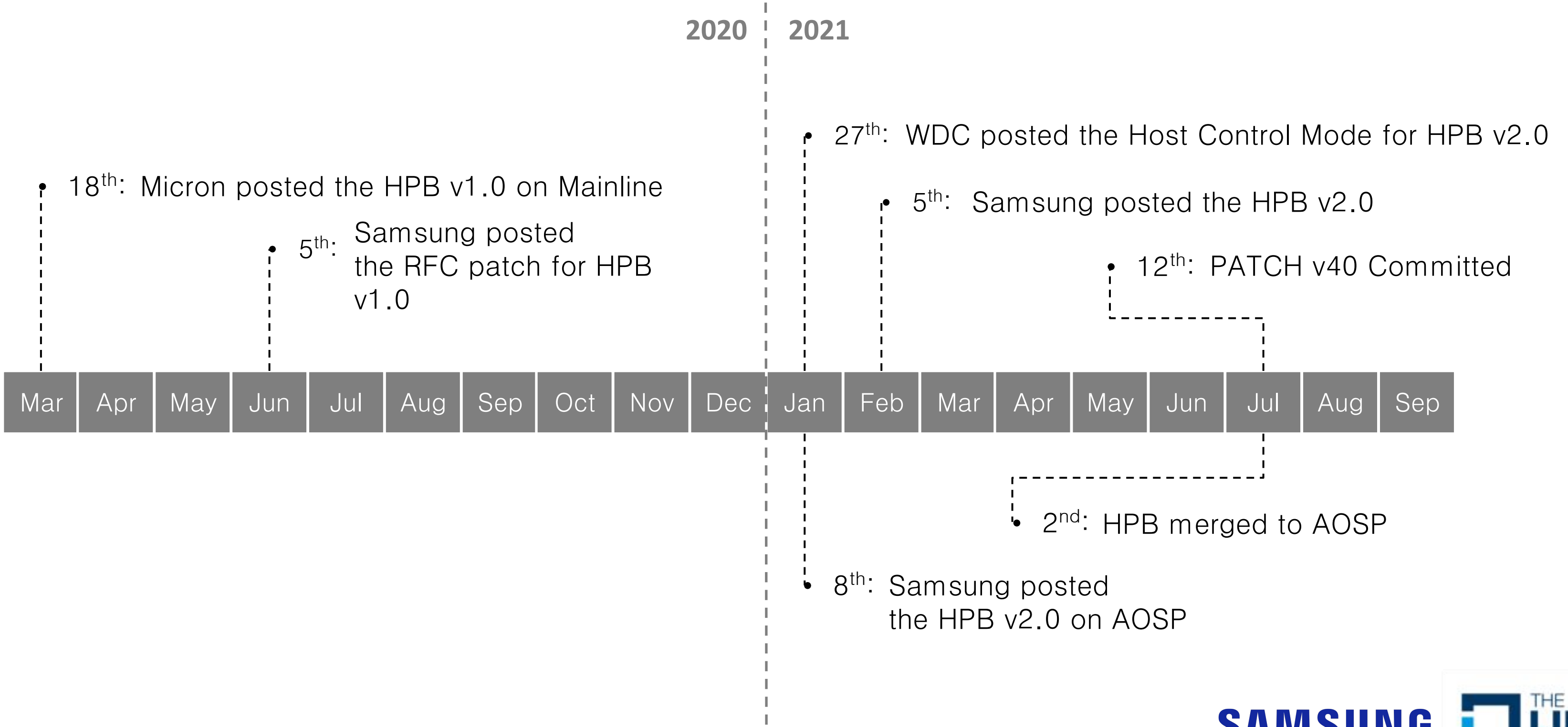
Current Status:

What is going on?

HPB Linux Upstream Status

- Upstream Started in Q2 of 2020
  - v40 of HPB 2.0 support (Daejun Park, Samsung)
  - Host Control Mode for HPB (Avri Altman, WDC)
  - Committed to SCSI tree in July 2021!
- Credits and Contribution: Thanks!
  - Avri Altman (WDC)
  - Bart Van Assche (acm)
  - Bean Huo (Micron)
  - Can Guo (Qualcomm)
  - Greg Kroah-Hartman (Linux Foundation)
  - Stanly Chu (Mediatek)

Timeline



2020
2021
18th: Micron posted the HPB v1.0 on Mainline
5th: Samsung posted the RFC patch for HPB v1.0
27th: WDC posted the Host Control Mode for HPB v2.0
5th: Samsung posted the HPB v2.0
12th: PATCH v40 Committed
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
2nd: HPB merged to AOSP
8th: Samsung posted the HPB v2.0 on AOSP

Conclusion

- HPB is still changing:
  - PATCH v40 is committed at 2021-07-12
  - Lots of suggestions are included since last year
- Contribute!
  - Review it, Test it, Share the bug fixes, Please!

Embedded Linux Conference