DRAM-less FTL I

Prof. Dongkun Shin (dongkun@skku.edu)
TA – Junho Lee (crow6316@skku.edu)
TA – Somm Kim (sommkim@skku.edu)
Embedded Software Laboratory
Sungkyunkwan University
http://nyx.skku.ac.kr

Contents

- Project Overview
- DRAM-less FTL
  - Jasmine Logging L2P Table
- Project 1: DFTL simulator

Project Overview

- Highlights of this coursework
  - Very big impact on your grade

- Design and implementation of DFTL
  - (20%) Project 1: DFTL Simulator
  - (30%) Project 2: DFTL on Jasmine (Porting)
  - (50%) Project 3: Custom DFTL

- Presentation at the end of term

DRAM-less FTL

DRAM-less FTL

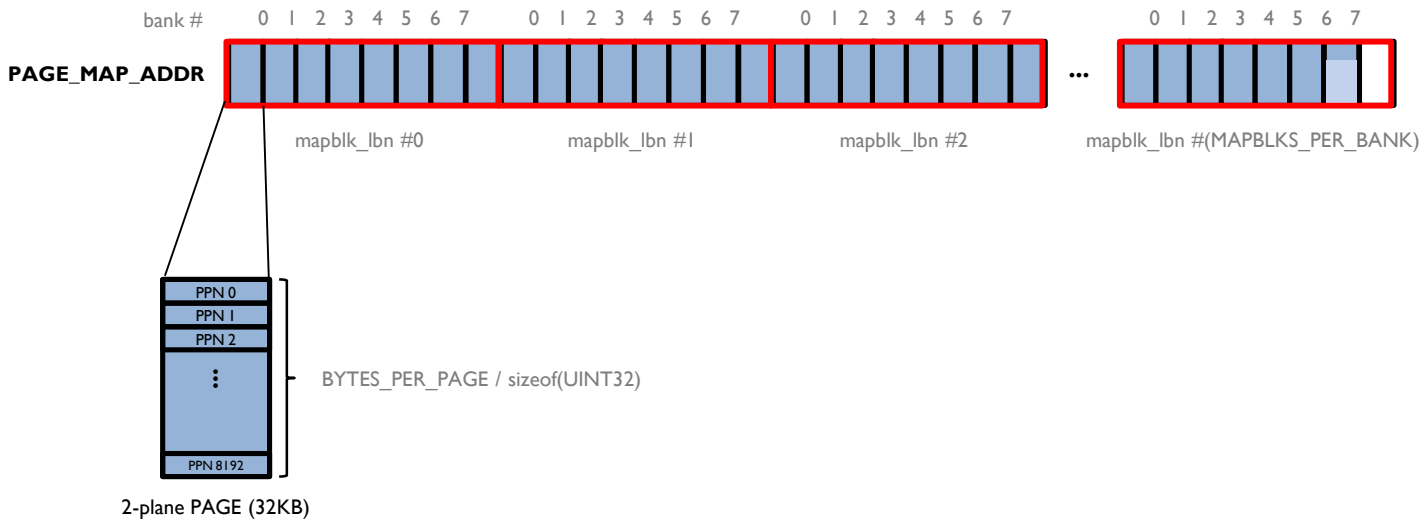
- What happens if we move the location of the page mapping table from DRAM to Flash?
  - L2P table must be read/write during I/O processing.

- DRAM is very expensive among SSD components.

- Price / performance ratio
  - Similar to DFTL with a CMT size of 1 page.

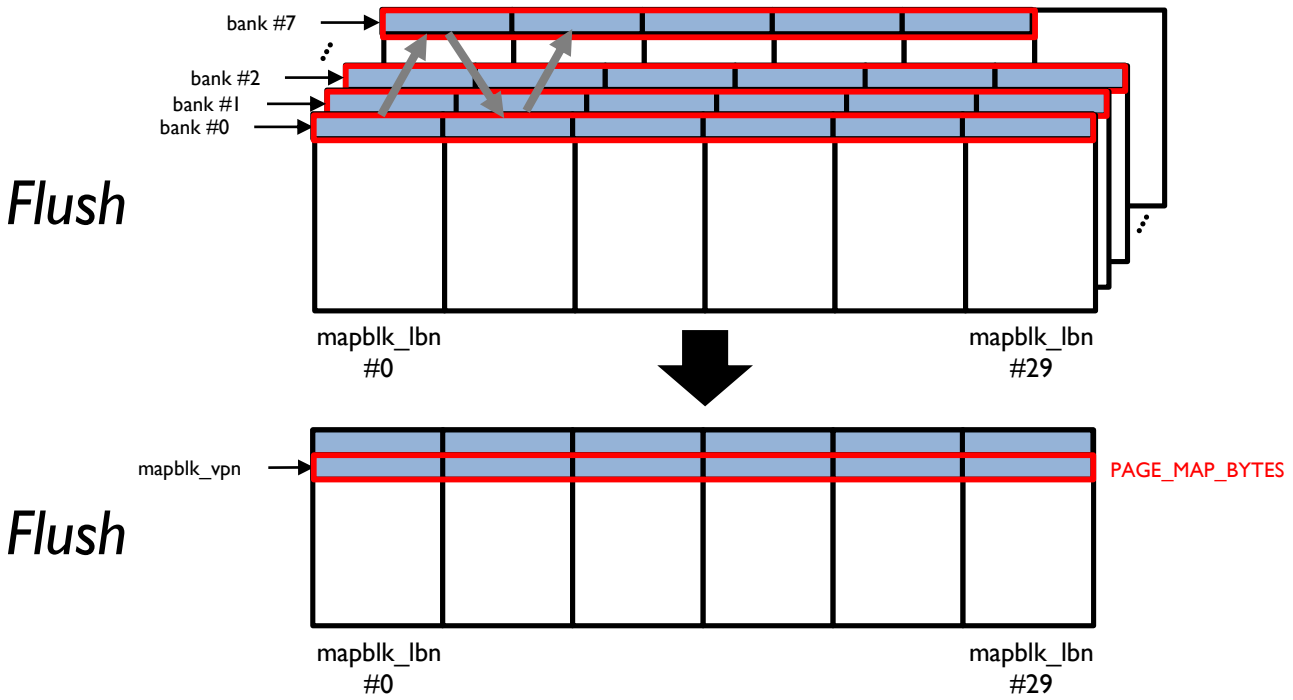
DRAM-less FTL

- Modify only *get_vpn()* and *set_vpn()* functions in Jasmine Greedy FTL *ftl.c* file.
- Logging page mapping table on Flash (for POR)
  - Trace *cur_mapblk_vpn[]* in *misc_metadata*.
    - GTD role in DFTL



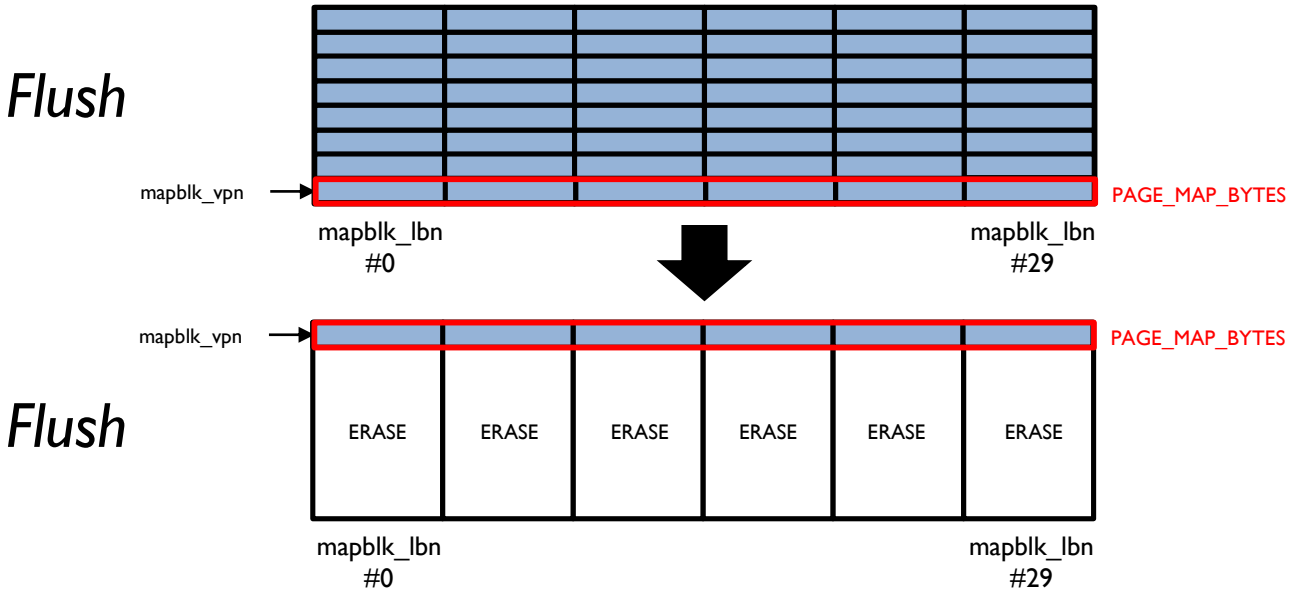
DRAM-less FTL

- Logging page mapping table on Flash (for POR)
  - Placement in Flash



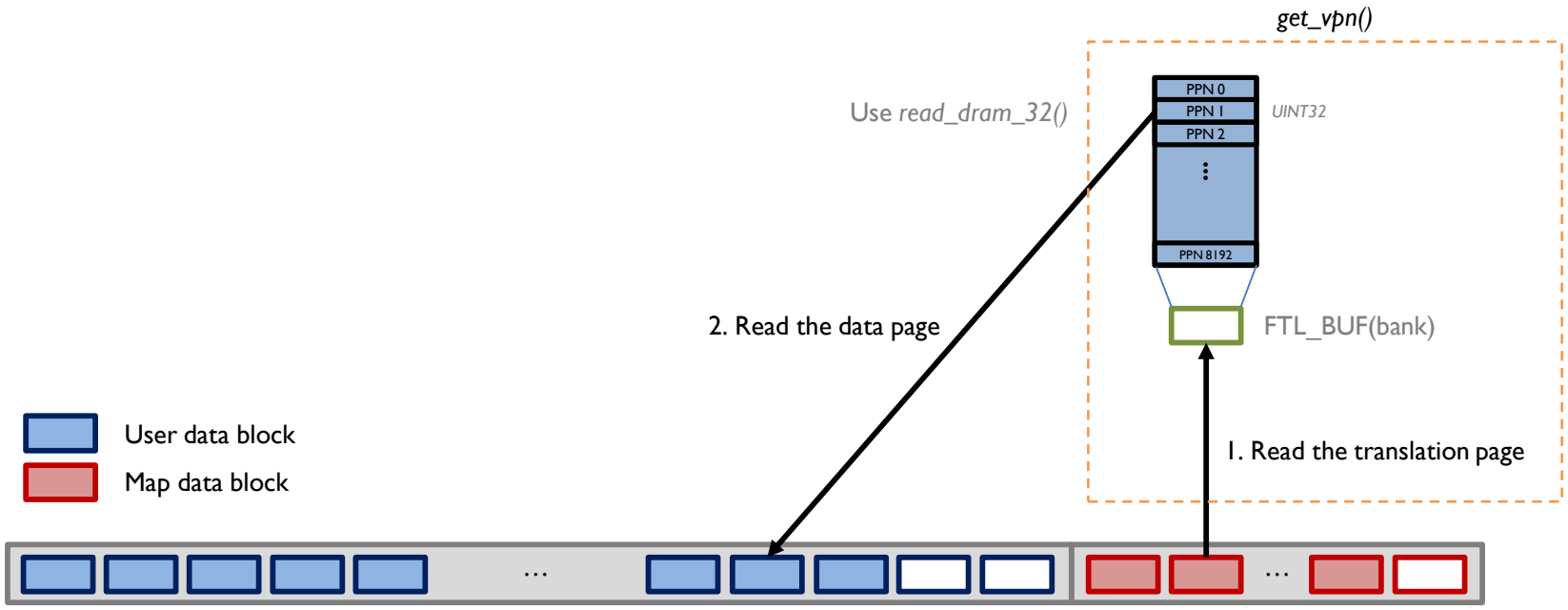
DRAM-less FTL

- Logging page mapping table on Flash (for POR)
  - Placement in Flash



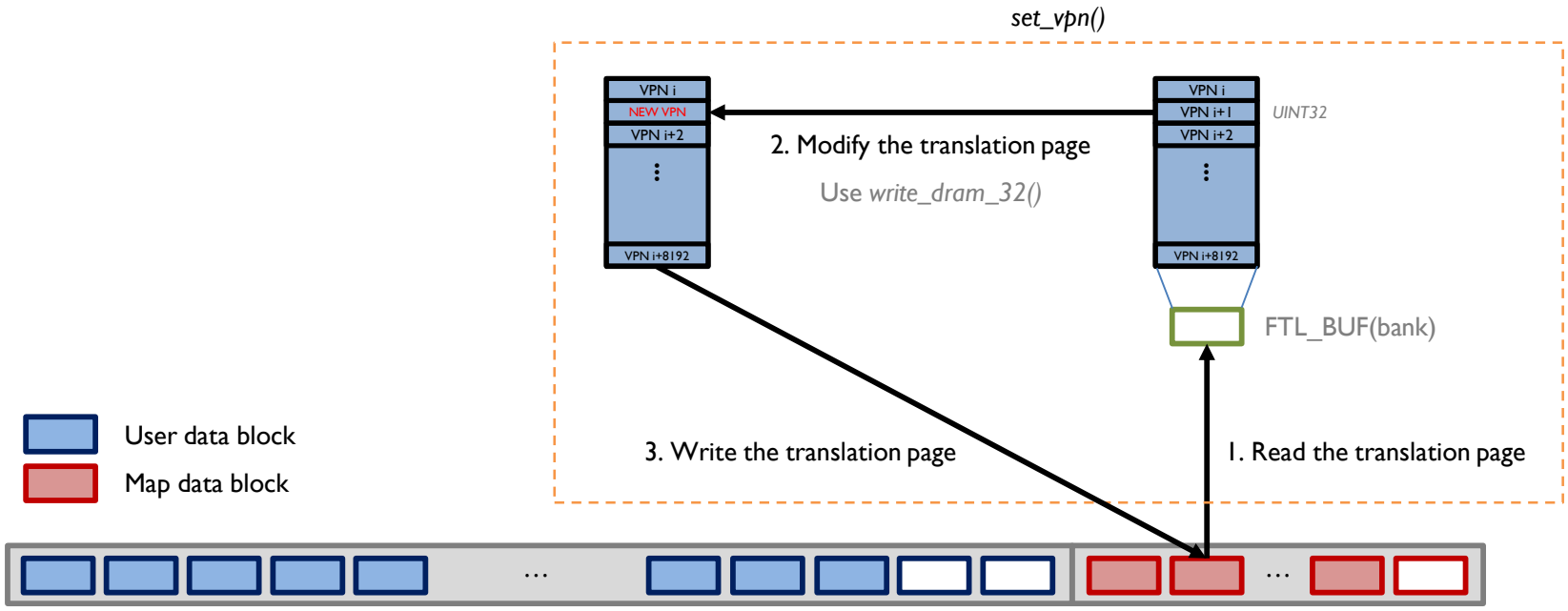
DRAM-less FTL

- *get_vpn()*
  - Read the translation page in Flash.
  - PAGE_MAP of the DRAM is ignored.



DRAM-less FTL

- *set_vpn()*
  - Modify the translation page in Flash.
  - Also update the DRAM's PAGE_MAP.



DRAM-less FTL

- Data structure of L2P table logged for POR
  - See the *logging_pmap_table()* and *load_pmap_table()* functions.
  - Calculate the location of the required map page (bank, block, page).

- Using flash wrapper function
  - When reading or writing the map page, do not use the host-specific Flash commands.
  - Use the *RETURN_WHEN_DONE* flag.
  - Call the *flash_finish()* function first before calling other flash functions.

Project 1
DFTL Simulator

Project 1 : DFTL Simulator

- Develop a DFTL[1] simulator
  - Simulate the operations of Map Cache based FTL
    - The mapping unit is page
  - The host requests I/O based on sector
    - Variable size of write/read request
  - Assumption
    - Sector size: 4B
    - Page size: 32B(data area) + 4B(spare area)
    - Initial state of flash blocks: empty

[1] A. Gupta, Y. Kim, and B. Urgaonkar, "DFTL: a flash translation layer employing demand-based selective caching of page-level address mapping," in Proc. ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Washington D.C., USA, pp. 229-240, March 2009.

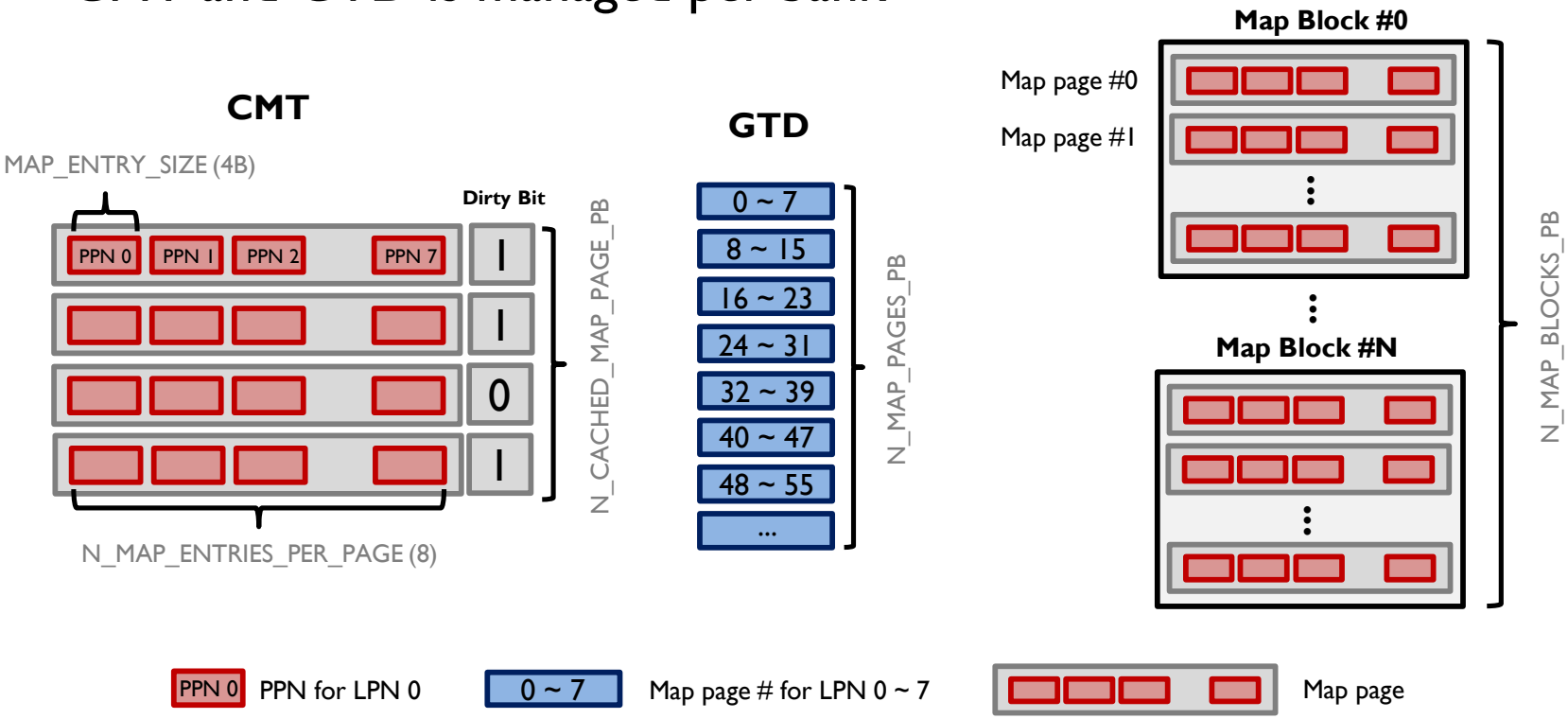
DFTL

- Entry-based Cache
  - Caches to the CMT in the unit of entry (LPN->PPN)
- Map page-based Cache
  - Caches to the CMT in the unit of map page
- Map page includes PPN list for continuous LPN

- We will implement map page based DFTL

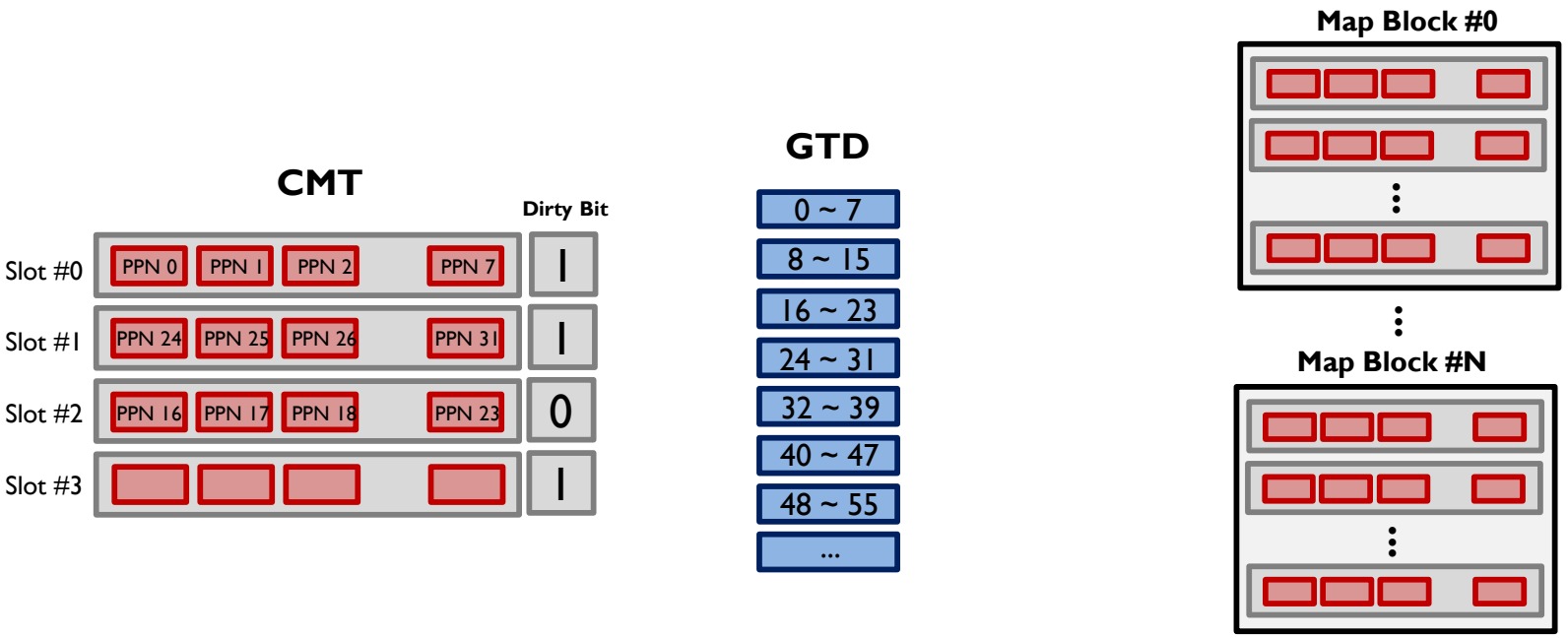
Descriptions

- CMT (Cached Map Table)
  - It is cached and evicted in map page units, not in map entry units.
- GTD (Global Translation Directory)
  - Store the physical page address (PPN) of the map page.
- CMT and GTD is managed per bank



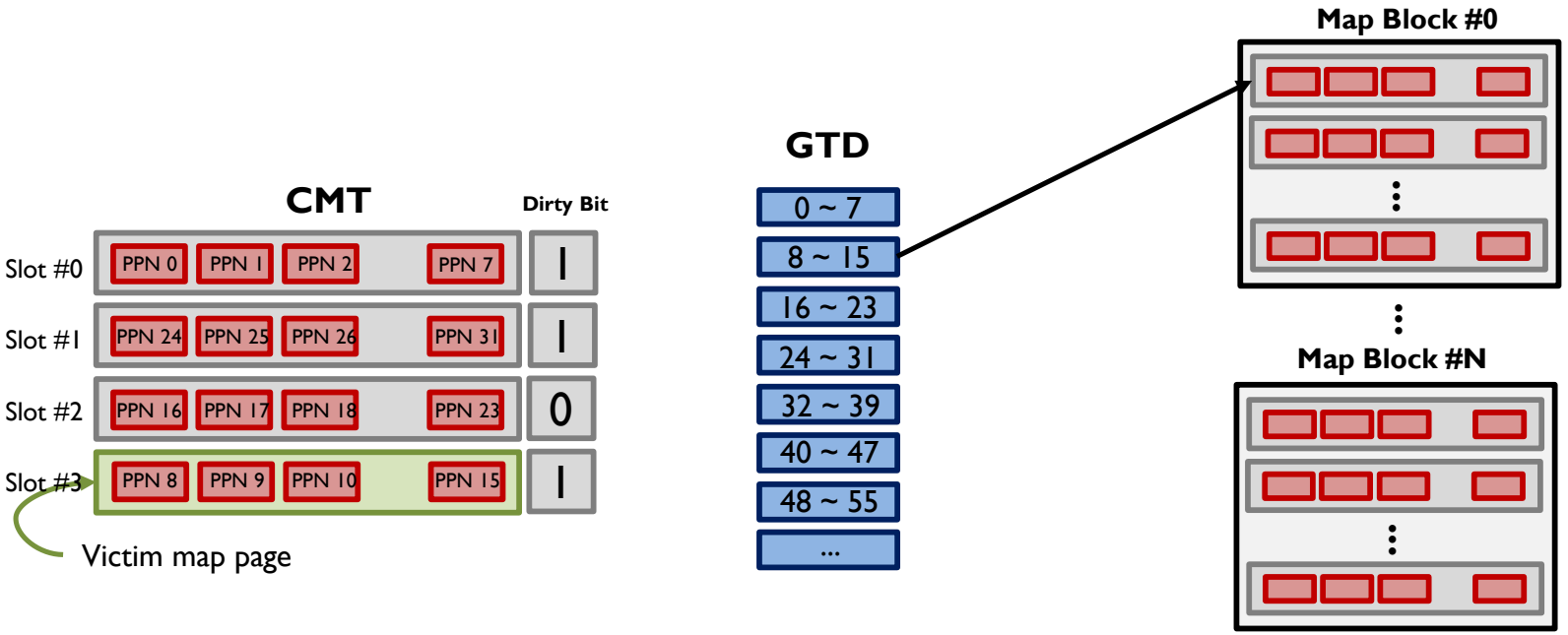
Descriptions

- If a free slot exists in the CMT, use the free slot to store PPN.
  - One slot in the CMT stores a ppn list of map page.



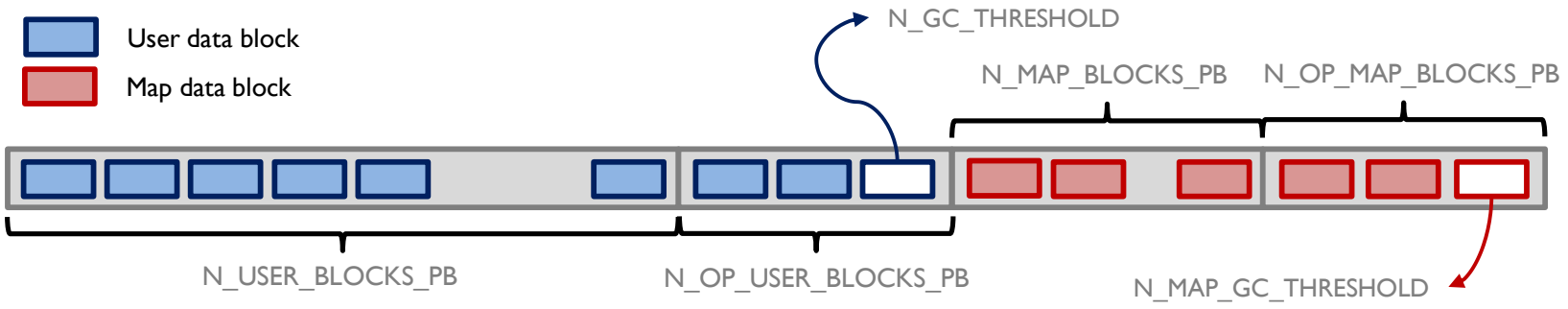
Descriptions

- CMT Management
  - Victim map page selection policy: LRU (Least Recently Used)
    - Select the map page accessed from the CMT the oldest time ago
- If the CMT is full, select the victim map page from the CMT.
- Write the victim map page to NAND.
- Set GTD to physical page address (PPN) of victim map page.



Descriptions

- Block Management
  - All blocks can be used as user blocks or map blocks.
  - Garbage collection
    - Perform when # of free blocks becomes *N_GC_THRESHOLD* in (*N_USER_BLOCKS_PB* + *N_OP_USER_BLOCKS_PB*)
  - Map garbage collection
    - Perform when # of free blocks becomes *N_MAP_GC_THRESHOLD* in (*N_MAP_BLOCKS_PB* + *N_OP_MAP_BLOCKS_PB*)
  - GC policy: Greedy



ftl.c

- ftl_open()
- ftl_read(u32 lba, u32 num_sectors, u32 *read_buffer)
  - *lba*: start sector #
  - *num_sectors*: # of sectors
  - Read the PPN from the CMT, not from the L2P table
  - If the PPN is not in the CMT, read and caching the corresponding map page from the NAND
- ftl_write(u32 lba, u32 num_sectors, u32 *write_buffer)
  - *lba*: start sector #
  - *num_sectors*: # of sectors
  - Store the PPN to the CMT, not to the L2P table
  - If the CMT does not have a free slot, evict a victim map page and caching new map page
  - You should add *s.ftl_write* for every *nand_write* call
- garbage_collection(u32 bank)
  - You should add *s.gc_write* for every *nand_write* call

ftl.c

- map_read(u32 bank, u32 map_page, u32 cache_slot)
  - *map_page*: Map page # to read from NAND
  - *cache_slot*: Slot # to caching map page
- map_write(u32 bank, u32 map_page, u32 cache_slot)
  - *map_page*: Map page # to write to NAND
  - *cache_slot*: Slot # to evict map page
  - You should add *s.map_write* for every *nand_write* call
- map_garbage_collection(u32 bank)
  - You should add *s.map_gc_write* for every *nand_write* call

- You should count *s.cache_hit*, *s.cache_miss*.
- The unit of write count variable of '*struct ftl_stat*' is sector.
- You can modify only ftl.c and nand.c.

Miscellaneous

- Recommended environment : Linux (Ubuntu is ok!)
  - You can do it in Windows, but be sure that your work also runs in Linux (I'll score all the works only in Ubuntu 16.04)
- Personal Project
- You should submit a report
  - Describe your code in detail.
  - Analyze the performance vs. CMT size (Draw a graph)
    - Configure *CMT_RATIO*
    - Configure *NO_CACHE* (only one map page cached)
  - Analyze the performance vs. # of OP blocks for Map
    - Configure *MAP_OP_RATIO*
- Submit to the icampus
  - Due: 11/06(Wed.) 23:59:59
  - File to submit: ftl_sim.c, ftl.h, ftl.c, nand.h, nand.c, Makefile, report.pdf
  - File name: $(STUDENT_ID).tar.gz
- Note: Project 1 has a large percentage of score, unlike Lab!

Any Questions?