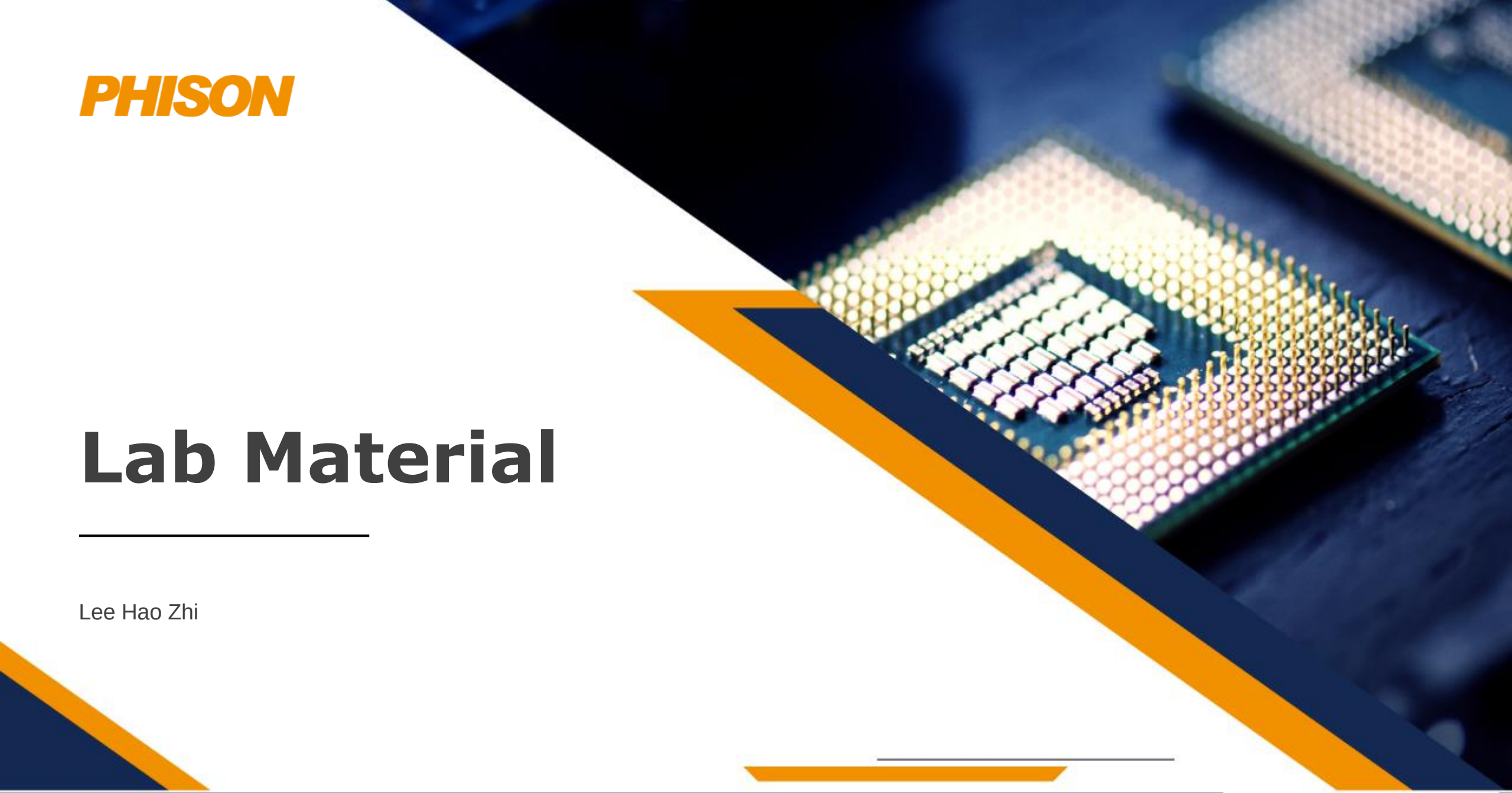
PHISON



Lab Material

Lee Hao Zhi

PHISON

AGENDA

1. FUSE SSD overview
2. FUSE SSD API overview
3. LAB
4. DEMO
5. NOTE

FUSE SSD Overview

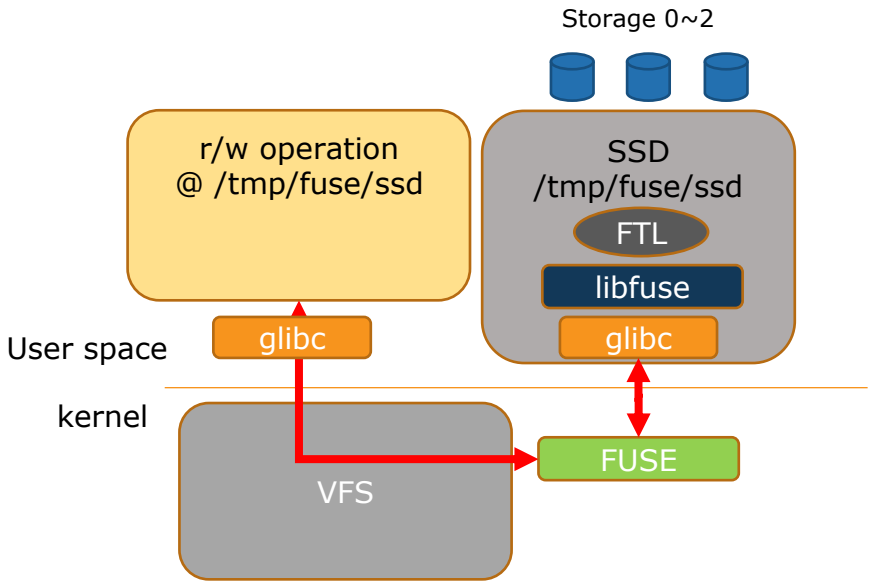
PHISON

FUSE SSD Overview

- Learning Flash Transaction Layer (FTL) algorithm in SSD device by FUSE kernel function

- Background
  - Real storage is NAND_0~NAND_X
    - It always read/write 512B data
    - If data is less than 512B, it still stores 512B
    - If data is larger than 512B, it shall store at different NAND
    - Each NAND_X is 1MB
  - FUSE_SSD will represent as FS (/tmp/fuse/SSD)
    - /tmp/fuse/SSD is a file
    - /tmp/fuse/SSD max size = total NAND_X size
    - FUSE_SSD will split data into 512B chunk size and store at different NANDs

- LBA requirement
  - Handle Logical and Physical address mapping
  - #ls will show logical size (512B align) not physical size
    - EX.
      - User write 256B offset_0 10 times
      - Logical Shall be size 512B
      - Physical shall be 512B*10

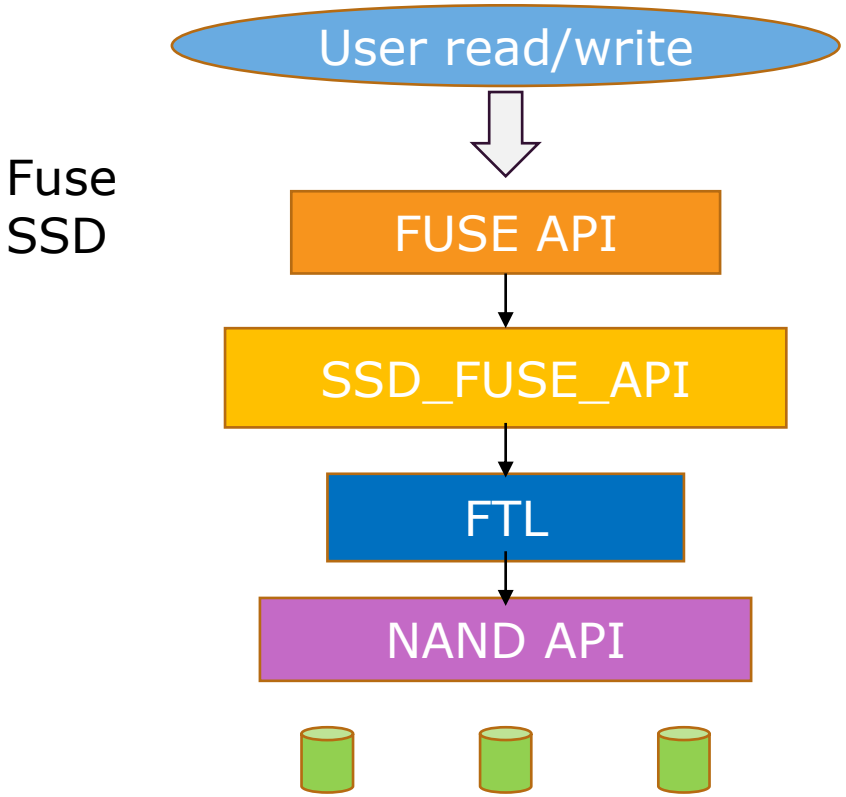


FUSE SSD API Overview

PHISON

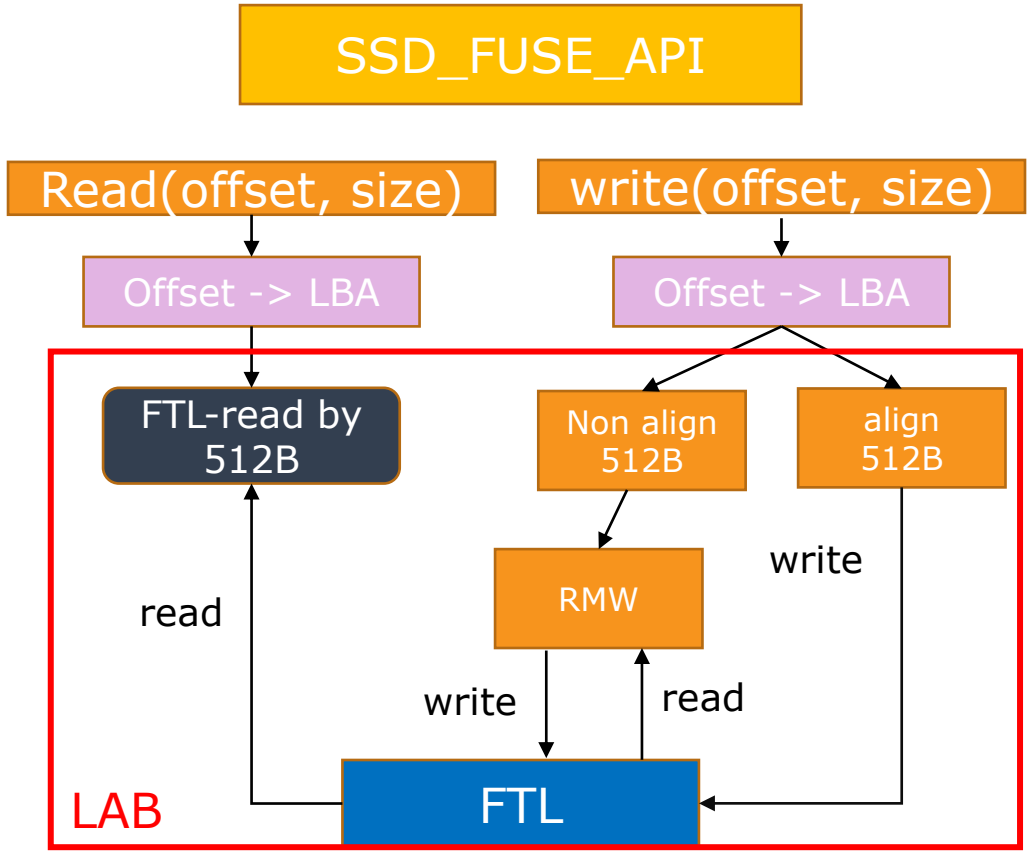
FUSE SSD API Overview

- Read/write in userspace will direct to Fuse SSD
- SSD_FUSE_API (Phison provide)
  - Change data offset into Logical Block Address (LBA)
  - LBA is 512B unit
- FTL
  - Handle LBA to Physical Cluster Address (PCA)
  - PCA is 512B unit
- NAND API (Phison provide)
  - Handle storage read/write
  - Storage is aligned to 512B



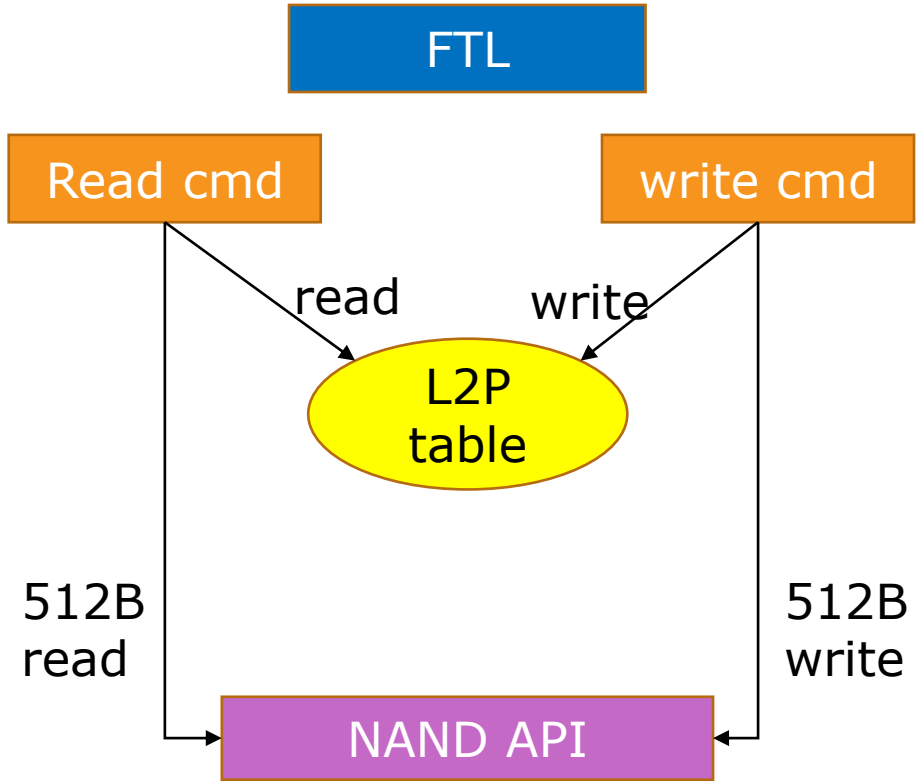
SSD Fuse API

- Read path(src_buffer, offset, size)
  - Change offset into LBA (512B)
  - Divide read cmd into 512B package by size
  - Send FTL-read API by 512B package
    - Allocate buffer for FTL
  - Copy data into src_buffer
- Write path (src_buffer, offset, size)
  - Change offset into LBA
  - Divide write cmd into 512B package by size
    - If not aligned to 512B: do Read modify write operation (RMW)
    - If aligned to 512B: send FTL-write API by LBA



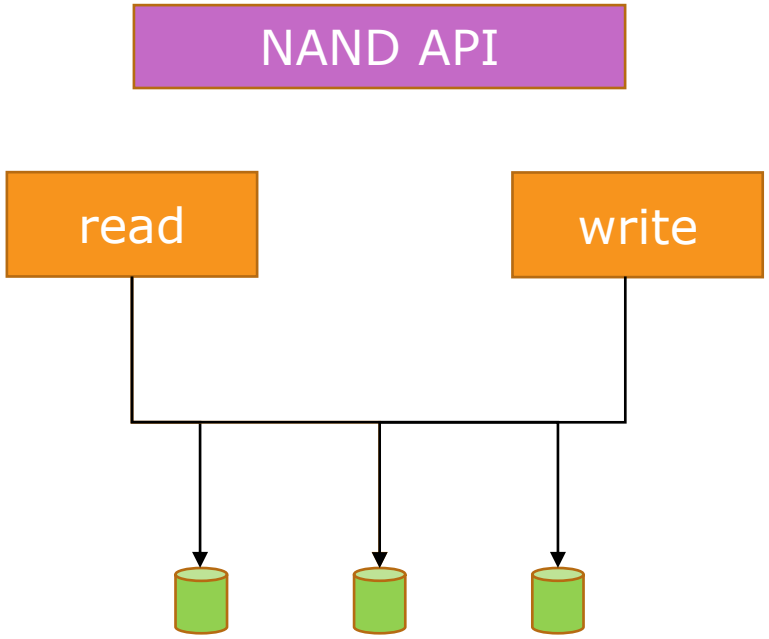
SSD Fuse API

- Read path (tmp_buffer, LBA)
  - Check LBA to PCA (L2P) to get true storage address
  - Send NAND-read cmd
    - Read data into tmp_buffer
- Write path (src_buffer, LBA)
  - Allocate a new PCA address
  - Send NAND-write cmd
  - Update L2P table
- Get logical size
  - Return current logical size
- Get physical size
  - Return current physical size



SSD Fuse API

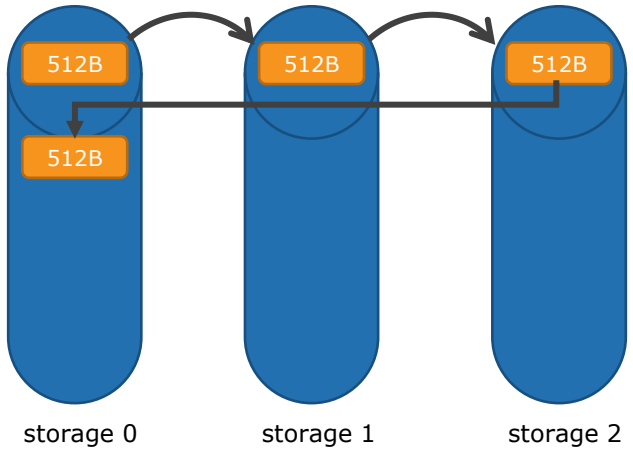
- Read cmd (PCA, buffer)
  - Read data from storage by PCA into buffer
- Write cmd (PCA, buffer)
  - Write data into storage by PCA from buffer



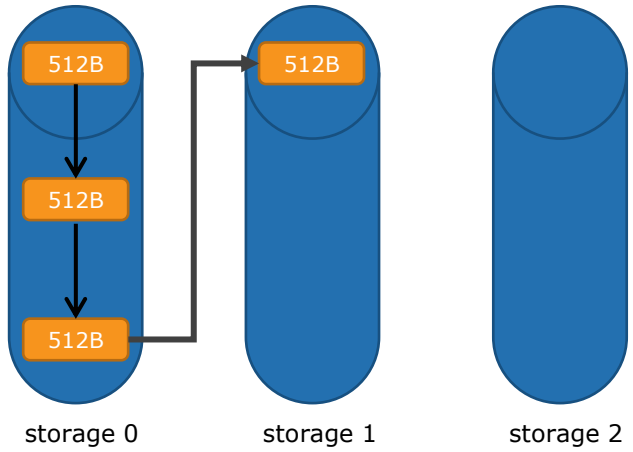
Example

- User space writes a 2KB data
- FUSE_SSD will split data into 512B chunk size

Sequence A



Sequence B



Demo

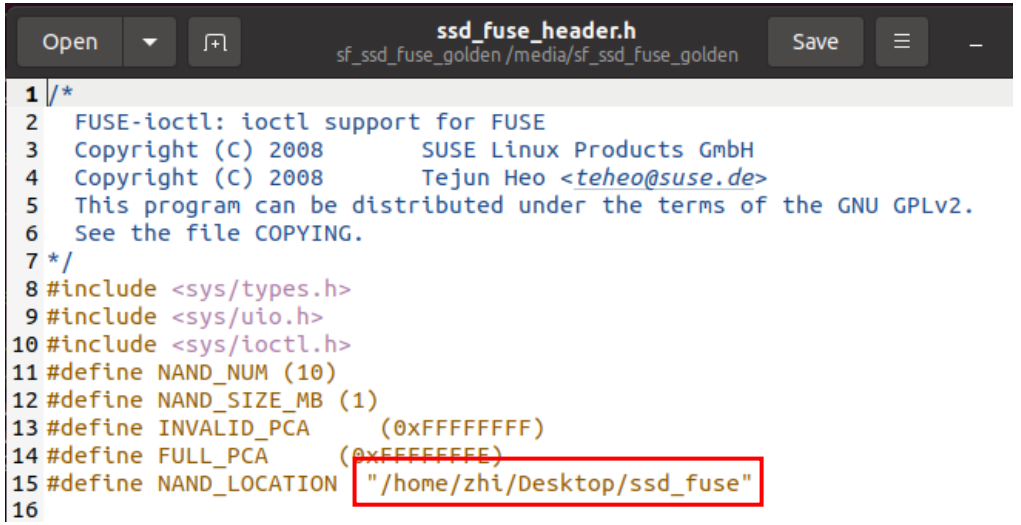
PHISON

Demo

- Fuse ssd: ssd_fuse.c
- Common header: ssd_fuse_header.h
- Dut: ssd_fuse_dut.c
- Package:
  - apt-cache search fuse
    sudo apt-get update
    sudo apt-get install fuse3
    sudo apt-get install libfuse3-dev
  - reboot

Demo

- Modify NAND_LOCATION to file location
- Compile ssd_fuse.c/ssd_fuse_dut.c
- #gcc –Wall ssd_fuse.c `pkg-config fuse3 --cflags –-libs` -D_FILE_OFFSET_BITS=64 –o ssd_fuse
- #gcc –Wall ssd_fuse_dut.c –o ssd_fuse_dut



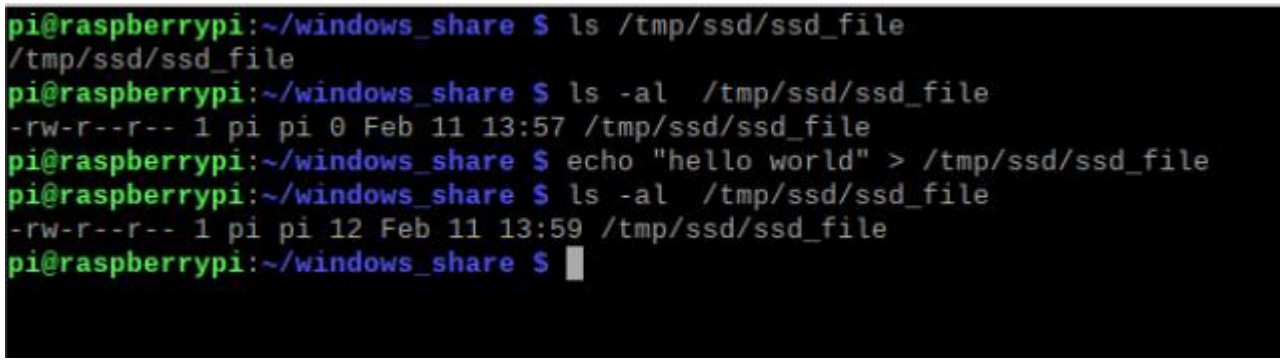
Demo

- Start SSD fuse lib with debug mode enable
- Create dir by #mkdir /tmp/ssd
- Mount at /tmp/ssd
- #./ssd_fuse –d /tmp/ssd

```
pi@raspberrypi:~/windows_share $ ./ssd_fuse -d /tmp/ssd
FUSE library version: 3.10.3
nullpath_ok: 0
unique: 2, opcode: INIT (26), nodeid: 0, insize: 56, pid: 0
INIT: 7.32
flags=0x03fffffb
max_readahead=0x00020000
   INIT: 7.31
   flags=0x0040f039
   max_readahead=0x00020000
   max_write=0x00100000
   max_background=0
   congestion_threshold=0
   time_gran=1
   unique: 2, success, outsize: 80
unique: 4, opcode: ACCESS (34), nodeid: 1, insize: 48, pid: 855
   unique: 4, error: -38 (Function not implemented), outsize: 16
unique: 6, opcode: LOOKUP (1), nodeid: 1, insize: 47, pid: 855
LOOKUP /.Trash
getattr[NULL] /.Trash
   unique: 6, error: -2 (No such file or directory), outsize: 16
unique: 8, opcode: LOOKUP (1), nodeid: 1, insize: 52, pid: 855
LOOKUP /.Trash-1000
getattr[NULL] /.Trash-1000
   unique: 8, error: -2 (No such file or directory), outsize: 16
```

Demo

- We can ls /tmp/ssd/ssd_file to get file details
- Can write a data to it (echo)
- Will notify data size is increased
- 0 -> 12



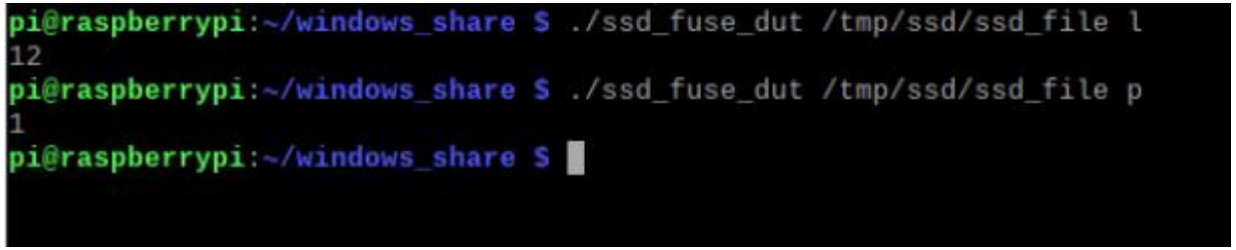
Demo

- In last page, we write “hello world” to ssd
- The fuse ssd will print out it allocate a pca to store data

```
getattr[NULL] /ssd_file
   NODEID: 2
   unique: 42, success, outsize: 144
unique: 44, opcode: OPEN (14), nodeid: 2, insize: 48, pid: 13747
open flags: 0x20201 /ssd_file
   open[0] flags: 0x20201 /ssd_file
   unique: 44, success, outsize: 32
unique: 46, opcode: FLUSH (25), nodeid: 2, insize: 64, pid: 13747
   unique: 46, error: -38 (Function not implemented), outsize: 16
unique: 48, opcode: WRITE (16), nodeid: 2, insize: 92, pid: 13747
write[0] 12 bytes to 0 flags: 0x20001
ssd wirte lba 0, range 1
ssd do write non align idx 0, size 12
gen first pca 0
nand write 0 pca pass
ftl update l2p lba=0, pca=0
ssd write return size = 12
   write[0] 12 bytes to 0
   unique: 48, success, outsize: 24
unique: 50, opcode: RELEASE (18), nodeid: 2, insize: 64, pid: 0
   unique: 50, success, outsize: 16
unique: 52, opcode: LOOKUP (1), nodeid: 1, insize: 49, pid: 17339
LOOKUP /ssd_file
getattr[NULL] /ssd_file
   NODEID: 2
   unique: 52, success, outsize: 144
```

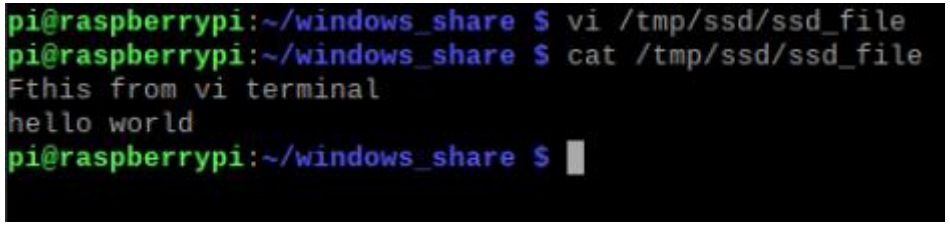
Demo

- Although logical size is 12B, the physical size is actually 512B.
- We can get this info by DUT
  - #./ssd_fuse_dut /tmp/ssd/ssd_file l
  - Return logical size
  - #./ssd_fuse_dut /tmp/ssd/ssd_file p
  - Return physical size (512B unit)



Demo

- User can use C to read/write ssd_file or use vi to modify ssd_file
- But can “not” use geany (GUI text editor)



Lab

PHISON

Lab Challenge

- Tasks:
  - Complete basic write/read function
  - Modify writing sequence A to sequence B
  - Implement garbage collection

- Target:
  - Keep WAF as minimum as possible
  - Compare data pass

Lab Challenge

- ftl_write (left blank):
  - Use get_next_pca to find empty PCA for data writing
  - Write data from buffer to PCA
  - Update L2P table
- ssd_do_write (left blank):
  - Divide write cmd into 512B package by size
  - Use ftl_write to write data
    - Need to handle writing non-aligned data

Lab Challenge

- get_next_pca (need modify):
  - Change next pca address to next page instead of next NAND

- ftl_gc (left blank):
  - Decide the source block to be erased
  - Move all the valid data that in source block to another block
  - Update L2P table
  - Erase the source block with invalid data

Note

PHISON

Step 5: Grouping

- Form a group of 4 students (by 5/4)

https://docs.google.com/spreadsheets/d/1n63uOx266j7DIyaaBsQIp59YoSdRnzWn8J-6a9cyvPw/edit?usp=sharing

Next Action

- Contact mentors for Q&A
  - haozhi_lee@phison.com
  - brian_liu@phison.com
  - irene_chen@phison.com
- Complete lab challenge and provide source code to mentors 5/11, before 9am
- Phison will generate host behaviors for testing
- Group with the most efficient GC algorithm (smallest WAF) can win prizes

PHISON

THANK YOU!