ICE3028: Embedded System Design - Project 5: Power-Off Recovery

Prof. Jinkyu Jeong (Jinkyu@skku.edu)
TA -- Minwoo Ahn (minwoo.ahn@csl.skku.edu)
TA -- Donghyun Kim (donghyun.kim@csl.skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

Policy

- Your log-block FTL should handle
  - Step #1 : Normal Power-Off (NPO) Recovery
  - Step #2 : Sudden Power-Off (SPO) Recovery
  - Step #3 : Better SPO Recovery

- Submission
  - Due date : 12/13 11:59:59 (Thur)
  - By e-mail : minwoo.ahn@csl.skku.edu
  - Attached file : tar.gz file + **report file**

Policy (cont'd)

- This project is based on each of your Log-block FTL (#5)
- Team project
- There should be no limitation on physical blocks
  - No min_vblk_avail and max_vblk_avail
- You should deliver final presentation based on the project
  - Date : 12/13 (Thur)
  - Duration : 10 minutes per a team
  - Additional information will be noticed

Normal Power-Off

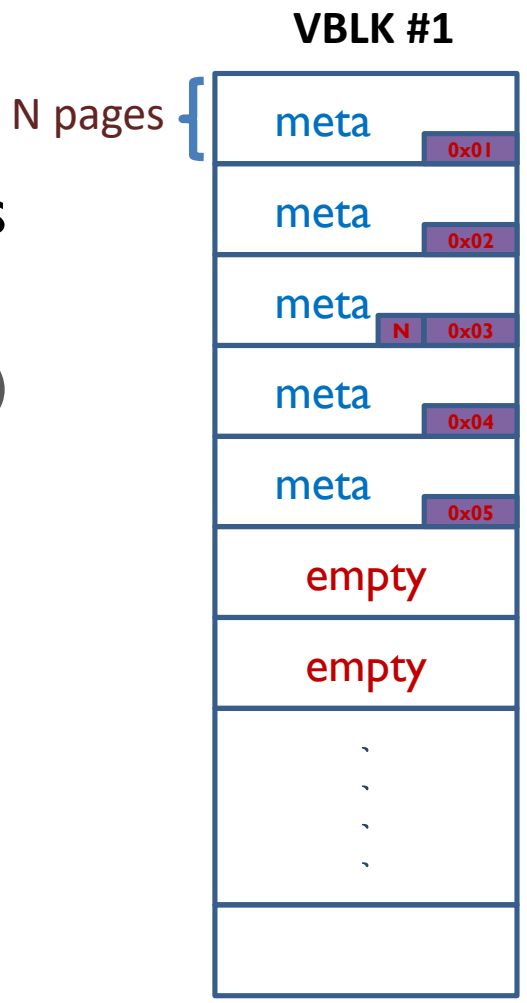
- Store some metadata to NAND flash during NPO (ftl_flush)
  - NOT all of metadata should be stored!
  - Some might be recovered with other metadata
- Recover metadata at boot time (ftl_open)
  - Read metadata from NAND flash
  - Recover other metadata with them
- Do not touch VBLK #0 (special purpose for bad-blk)
- Hint : GreedyFTL in OpenSSD project

Sudden Power-Off

- Your FTL should handle sudden power loss
  - Once sending completion acknowledgment to the host, the write request should be **persistent** after sudden power loss
  - Remind that SPO can be happened anytime, anywhere in your code
    - (That's why it's called 'Sudden' power-off)
    - When ftl_write, nand_program, gc (merge), …
    - Even when ftl_flush, ftl_open and recovering(!)
- You should consider following things
  - Before SPO happens
    - NAND_PROGRAM order (who comes first? data or metadata)
    - What to store to flash memory (which metadata?)
  - After SPO happens (recovery)
    - How to recover metadata (SRAM, DRAM)
    - Are up-to-date data available?

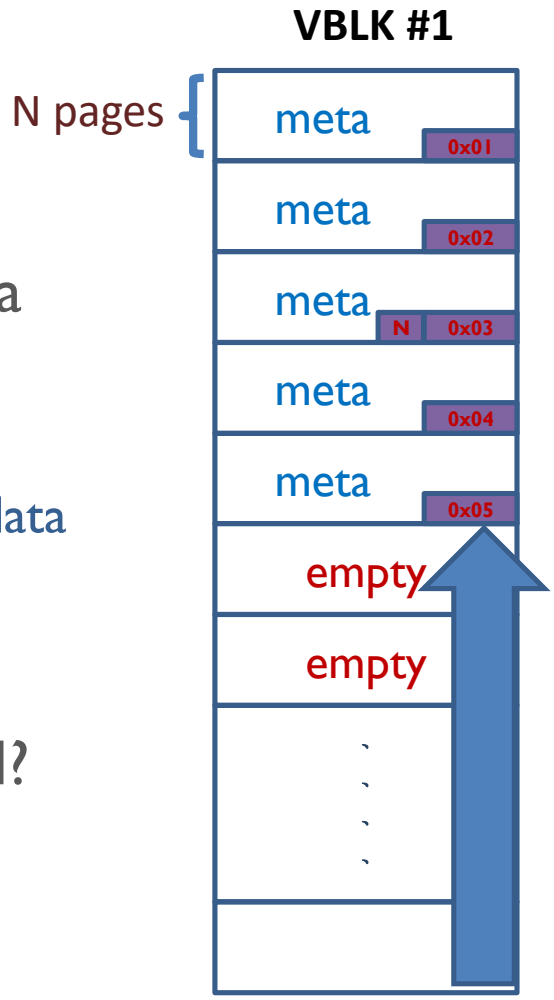
Guideline in SPO

- Consider all cases that sudden power off may happens
- For every ftl_write, do following things
  - Data write
  - Metadata write to special VBLK (#1 or #2)
    - Block map
    - Log-block map (which vblk is log-block?)
    - Page map of log-block
    - Current write page offset of log-block
    - Anything else?
  - Mark written metadata as 'up-to-date'
    - Like a special marker
  - At ftl_flush, add special marker
    - Necessary at boot time



Guideline in SPO (cont'd)

- Figure out whether NPO or SPO
  - Read VBLK #1 from tail to head
    - Until finding valid mark
  - If NPO, there's no headache; just read meta
  - If SPO, do recovery
    - Read written metadata at up-to-date zone
    - With a recovered metadata, recover other metadata



- Do not trust cur_write_page at SPO
  - What if power loss at NAND_PROGRAM?
  - You will need VBLK #2 for metadata
  - What about log-blocks?
  - **What if power loss during recovery?**

Better SPO Recovery

- Current SPO policy
  - Store all metadata for every write (slow write)
  - Load all metadata at boot time (slow boot)
  - How to make it better?

- You can think and make any idea freely
- We will consider effect, novelty, realization of idea

Report

- Please make your best effort to report
- Best report is a simple, neat, clean, graphical, understand able and **short** report
- We'll evaluate your final presentation based on the report

Debugging Tips

- NPO
  - Modify ftl_test.c to flush and load metadata
  - Real tests will be done with real file system and real files
    - Installing game
    - Compiling your codes on your FTL
- SPO
  - Make _fault_injection in your codes
    - Simple _fault_injection function code

```
UINT32 _fault_injection (void)
{
        uart_printf("Fault_injection: Power-Off!");
        uart_printf("Shutdown Jasmine board right now");
        while(1);
}
```

Debugging Tips (cont'd)

- SPO
  - Real test will be done with timer interrupt
  - Enabling timer interrupt
    - At target_spw/initialize.c

```
90
91          SETREG(SDRAM_ECC_MON, 0xFFFFFFFF);
92
93          // configure SDRAM interrupt
94          SETREG(SDRAM_INTCTRL, SDRAM_INT_ENABLE);
95
96          // clear interrupt flags in DRAM controller
97          SETREG(SDRAM_INTSTATUS, 0xFFFFFFFF);
98
99          // configure ICU
100         SETREG(APB_ICU_CON, INTR_SATA); // SATA = FIQ, other = IRQ
101         SETREG(APB_INT_MSK, INTR_SATA | INTR_FLASH | INTR_SDRAM | INTR_TIMER_1 | INTR_TIMER_2 | INTR_TIMER_3);
102
103         // set timer interrupt
104         SET_TIMER_CONTROL(TIMER_CH2, 0);
105         CLEAR_TIMER_INTR(TIMER_CH2);
106         SET_TIMER_LOAD(TIMER_CH2, 87489064);    // initial value of the timer
107         SET_TIMER_CONTROL(TIMER_CH2, TM_ENABLE | TM_BIT_32 | TM_INTR | TM_MODE_PRD | TIMER_PRESCALE_0);
108
109         // clear interrupt flags in ICU
110         SETREG(APB_INT_STS, 0xFFFFFFFF);
111
112         flash_reset();
113
114         SETREG(FCONF_PAUSE, 0);
115         SETREG(INTR_MASK, 0);
```

87489064 = 1 second

Debugging Tips (cont'd)

- At target_spw/misc.c

```
80  #ifdef __GNUC__
81  void irq_handler(void) __attribute__ ((interrupt ("IRQ")));
82  void irq_handler(void)
83  #else
84  __irq void irq_handler(void)
85  #endif
86  {
87          UINT32 intr_stat = GETREG(APB_INT_STS);
88
89          if (intr_stat & (INTR_TIMER_1 | INTR_TIMER_2 | INTR_TIMER_3))
90          {
91                  g_timer_interrupt_count++;
92
93                  if(rand() % 100 == 0)
94                  {
95                          uart_printf("IRQ_Interrupt: Power-Off!");
96                          uart_printf("Shutdown Jasmine board right now");
97                          while(1);
98
99                  }
100
101                 CLEAR_TIMER_INTR(TIMER_CH1);
102                 CLEAR_TIMER_INTR(TIMER_CH2);
103                 CLEAR_TIMER_INTR(TIMER_CH3);
104                 SETREG(APB_INT_STS, INTR_TIMER_1 | INTR_TIMER_2 | INTR_TIMER_3);
105         }
```

Debugging Tips (cont'd)

- Also make fault_injection to flash wrapper function
- Modify ftl_test.c to verify read after recovery
  - Current ftl_test.c issues write with a value from rand() and store it its own DRAM area, which is not valid any more (DRAM is volatile)
- Discuss it with your teammate and contact me if it's not enough
- Step by step approach