Project 2
F2FS-optimized FTL

Joohyung Park(joohyungpark@csl.skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
Project Goal

• Do not implement extra features of FTL
  • Ex) Trim, NCQ, DRAM buffering

• Focus on
  • Page allocation policy
  • Mapping unit and policy
  • GC management policy
  • Valid copy control

• Fit your own FTL into given I/O pattern for
  • Less memory footprint, better performance
  • I/O pattern: SQLite 35K tuples inserted onto F2FS
    • An fsync per an insertion
Given I/O Pattern
Write Upper Part

In-place updates
Write Lower Part: Data
Write Lower Part: Meta

Append only, not in-place updates
Testing Process

• Make sure the Jasmine to be included in /dev/sd*
  • lsblk

• Make a partition to test
  • fdisk /dev/sd*
  • ‘n’ for a new partition creation
  • ‘w’ to commit the modification

• Run the given script
  • ./F2FS.sh $DEVICE_MINOR $PARTITION_PATH
Results

- `.sqlite3.db`
  - All I/O events on Jasmine
- `*.*read / *.*write`
  - Prefix d = data, m = meta
  - nanosec | starting | length
- `*.info`
  - Total events num | total amount of I/O in sectors
- `*.png`
  - Visualized results
- `*.*runtime`
  - Elapsed time info
Grading Policies

• Document Grade \times \frac{\text{dram}_\text{org}}{\text{dram}_\text{yours}} \times \frac{\text{wrt}_\text{org}}{\text{wrt}_\text{yours}} \times \frac{\text{rrt}_\text{org}}{\text{rrt}_\text{yours}}

• Document requirements
  • DRAM layout
  • All of custom defined structures
  • All of global variables
  • Objective/usage of layout, structures, and variables
  • What is the problem?
  • Why should you argue your approach?
  • Expected gain / disadvantage of your approach
  • Specification to be uploaded