JLFS: Journaling the Log-Structured Filesystem for Proactive Cleaning in Flash Storage

Andromachi Hatzieleftheriou and Stergios V. Anastasiadis
{ahatziel, stergios}@cs.uoi.gr
Dept. of Computer Science, University of Ioannina, Greece

1. Flash Memory

- Attractive features
  - Relatively cheap writes
  - Batch processing of multiple requests before reaching LFS
  - Flash reads avoided during garbage collection
  - Possible to adjust hotness boundary to workload
  - Selectively journal data based on a write size threshold

- Idiosyncrasies
  - Workload-dependent performance

- Low-level modifications
  - Flash translation layer

- Flash-aware fs
  - Semantic information
  - No functionality duplication
  - Retroactive decisions

- Journal responsibility
  - Proactively clean permanent state from frequently updated data and metadata

- Idea
  - Categorize pages into hot/cold based on cache timers
  - Respectively transfer data to journal or to LFS partition

- LFS partition
  - Contains mainly valid data → cold pages

- Journal Partition
  - Valid blocks at the front and clean at the rest → hot pages

2. Garbage Collection

- Log-structured approach typically used
  - Avoid random writes
  - Need for recycling of invalid pages

- FTL approach:
  - Separate hot/cold pages based on update frequency
  - Possible misclassification of access frequency
  - Bookkeeping overhead

3. Design Goals

1. Avoid function duplication between fs and firmware
2. Minimize write traffic without compromising persistence
3. Minimize data relocation traffic
4. Identify access characteristics using existing cache mechanisms

Fs level approach:
- Use semantic knowledge and existing system operations to manage flash idiosyncrasies

4. Proposed Architecture

- JLFS - Journaled Log-structured File System
  - Combine journaling functionality with LFS
  - Flash storage management
    - Two distinct partitions; LFS and journal

5. JLFS Concepts

- Journal responsibility
  - Proactively clean permanent state from frequently updated data and metadata

- Idea
  - Categorize pages into hot/cold based on cache timers

- LFS partition
  - Contains mainly valid data → cold pages

- Journal Partition
  - Valid blocks at the front and clean at the rest → hot pages

6. JLFS Features

1. Relatively cheap writes
2. Batching of multiple requests before reaching LFS
3. Flash reads avoided during garbage collection
4. Possible to adjust hotness boundary to workload
5. Selectively journal data based on a write size threshold
  - Avoid traffic duplication in case of sequential requests
6. Journal data updates at subpage granularity
  - Reduce journal traffic in case of small writes
7. Reduced recovery time

7. References