The Composite-File File System: Decoupling the One-to-one Mapping of Files and Metadata for Better Performance

Shuanglong Zhang, Helen Catanese, Andy An-I Wang
Computer Science Department, Florida State University

02/23/2016

Overview

- Current state
  - One-to-one mapping of a logical file and its physical metadata and data representations
- Observations
  - Files are accessed in groups
    - Tend to be small and share similar metadata
- Composite-File File System (CFFS)
  - Many logical files can be consolidated into a single ***composite file*** with its shared metadata and representation
  - Up 27% performance gain

Current State

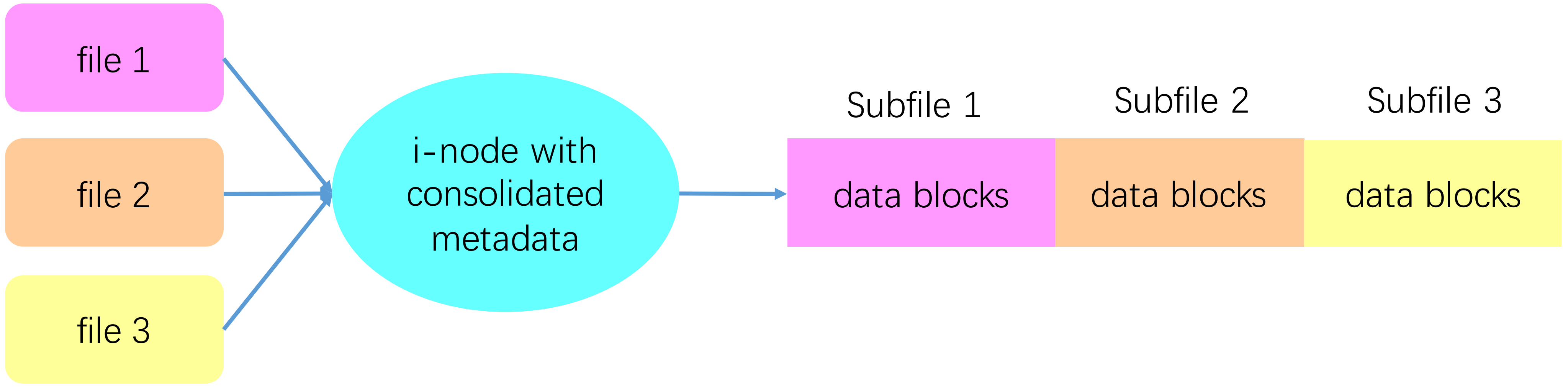
- Each logical file is mapped to its physical metadata and data representations
  - Deep-rooted data structures
  - Natural granularity for many file system mechanisms
    - VFS API, prefetching, etc.

- Suppose we relax this constraint
  - Can we create new optimization opportunities?

Observations

- Frequent access to small files
  - Metadata a major source of overhead for small files (~40%)
- Redundant metadata information
  - Potential opportunities to consolidate
- Files accessed in groups
  - Why physically represent them separately?
- Limitation of prefetching
  - High per-file access overhead (seeking) even with warm cache

A Composite File



Metadata Design Highlights

Upper bits	Lower bits
00	**0**
00	**1**
01	**0**
01	**1**
10	**0**
10	1
11	**0**
11	1

- Modified i-node namespace (highlighted)
  - If number > threshold (e.g, 011)
    - Treat zero extended upper bits as i-node numbers
    - Treat lower bits as subfile numbers
- Directory representation
  - Names are mapped to modified i-node numbers
- Subfiles' metadata stored in extended attributes
  - Permission: First check the permission of the composite file, then the target subfile

Subfile Operations

- Open/close
  - Open the composite file and seek to the offset of target subfile
  - Close the entire composite file
- Add a subfile
  - Append to the end
- Remove a subfile
  - Mark it as freed

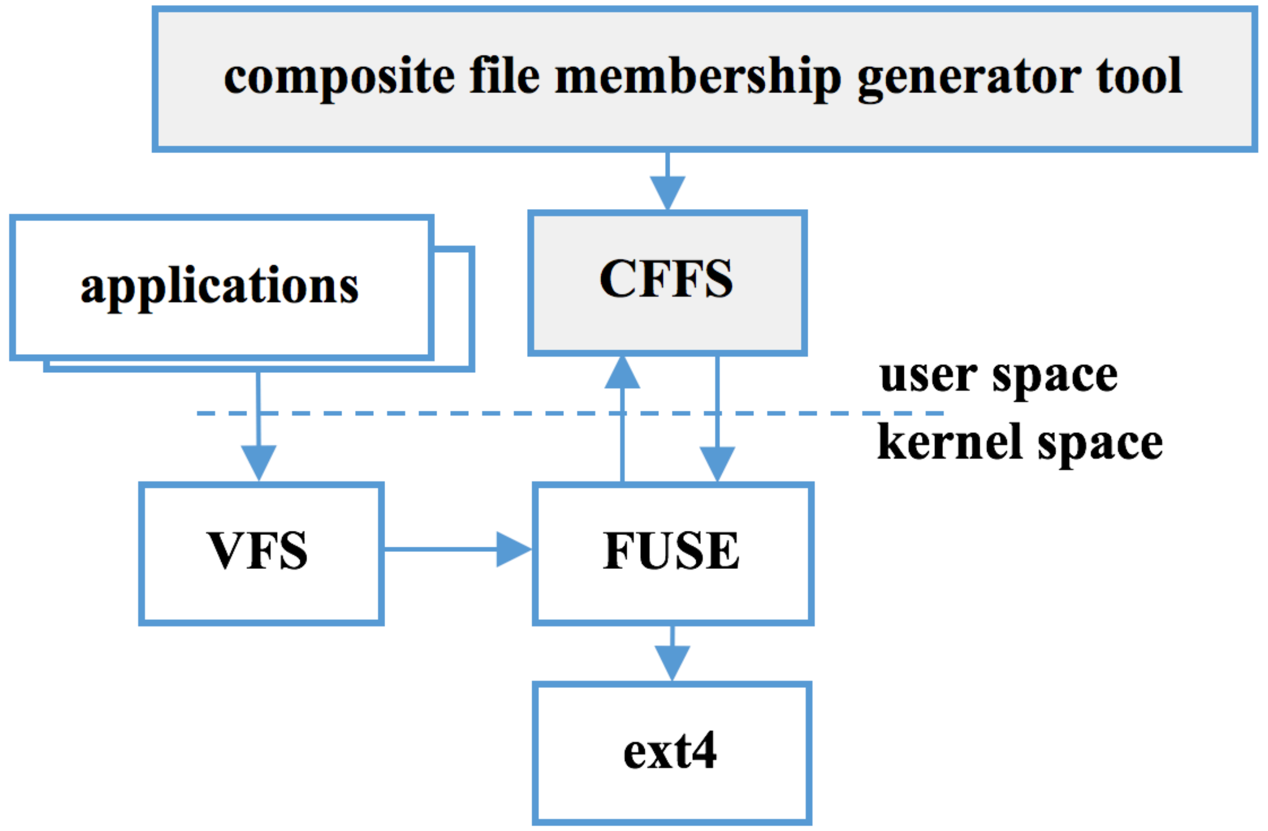
Subfile Operations (cont.)

- Read/write operation
  - Read from the starting offset of the subfile, bounded by subfile size
  - Write from the starting offset of the subfile, bounded by subfile size
    - May move to the end if there is not enough space
- Space compaction
  - When half of the space of a composite file is marked as freed

Ways to Form Composite Files

- *Directory-based* consolidation
  - Groups all files in one directory
- *Embedded-reference-based* consolidation
  - Groups files based on the extracted references (e.g., URLs)
- *Frequency-mining-based* consolidation
  - Based on variants of Apriori Algorithm
    - Frequently encountered file request sequences must contain frequently encountered subsequences

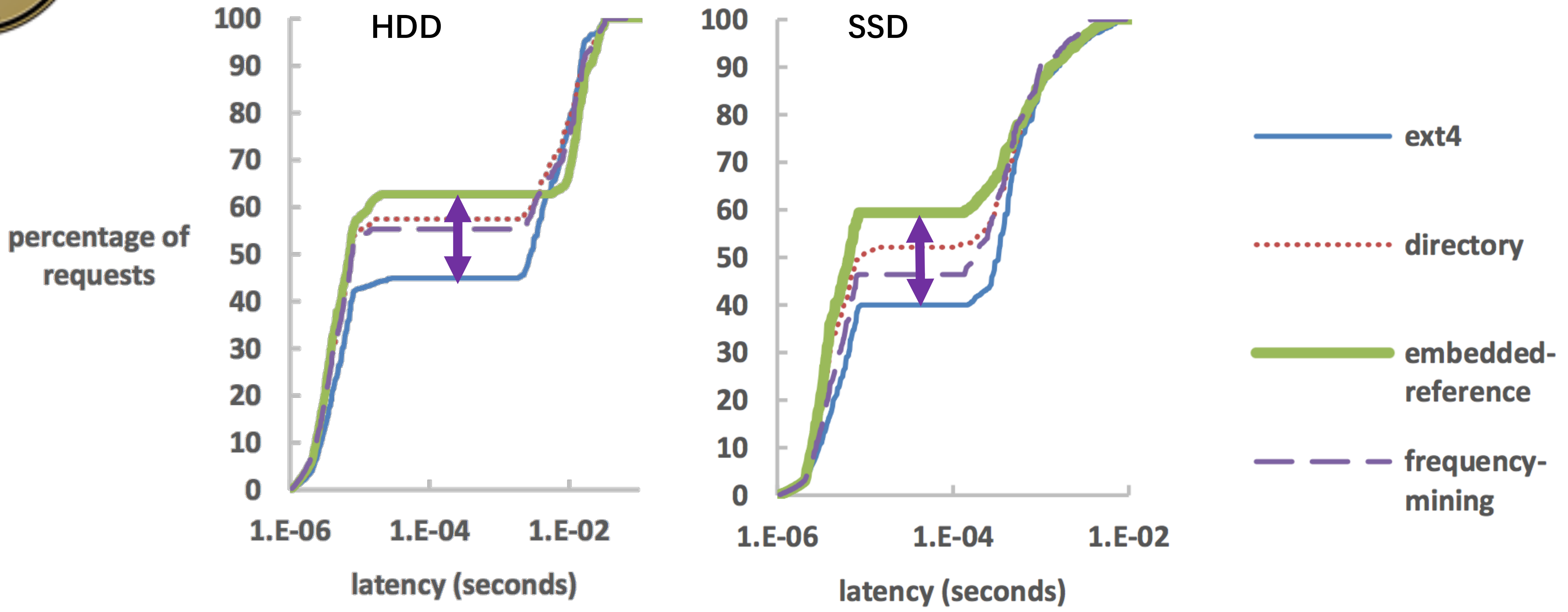
CFFS Components



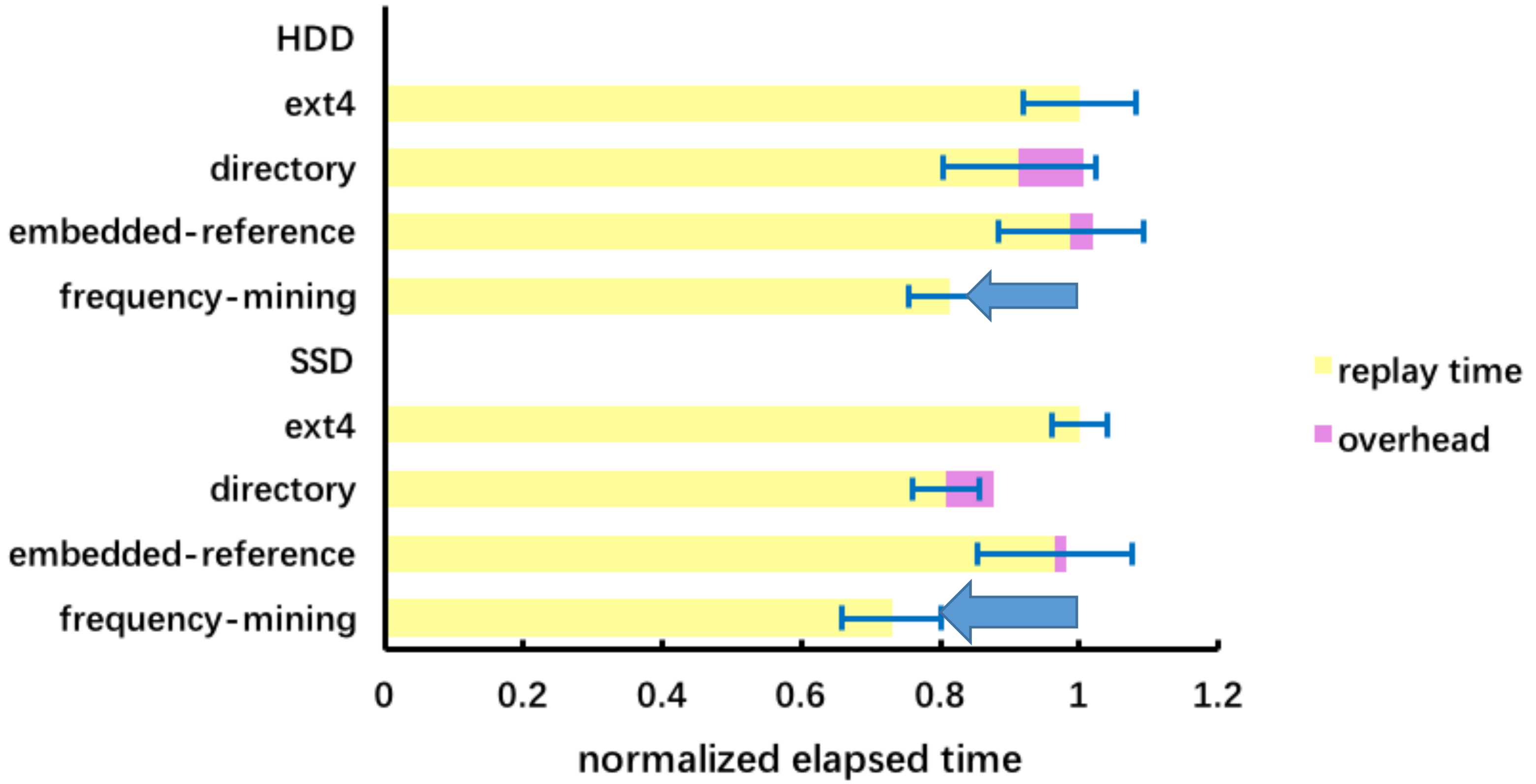
Empirical Evaluation

- Prototyped CFFS via FUSE+ext4
  - Use hardlinks to map multiple file names to the composite file i-node
  - Use extended attributes to store consolidated metadata
- FUSE+CFFS+ext4 vs. FUSE+ext4
- Workloads
  - 3-month long web server trace, 11-day long software development trace
  - Zero-think time replays

Web Server Latency



Software Development Trace Replays



Conclusion

- CFFS decouples the one-to-one mapping of files and metadata
  - Increases throughput up to 27%
  - Reduces latency up to 20%
- The CFFS approach is promising