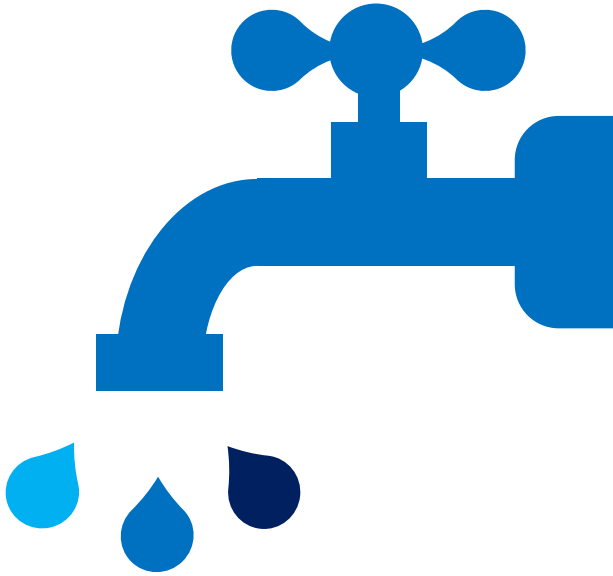
FStream:

Managing Flash Streams in the File System



Eunhee Rho, Kanchan Joshi, Seung-Uk Shin, Nitesh Jagadeesh Shetty,
Joo-Young Hwang, Sangyeun Cho, Daniel DG Lee, Jaeheon Jeong

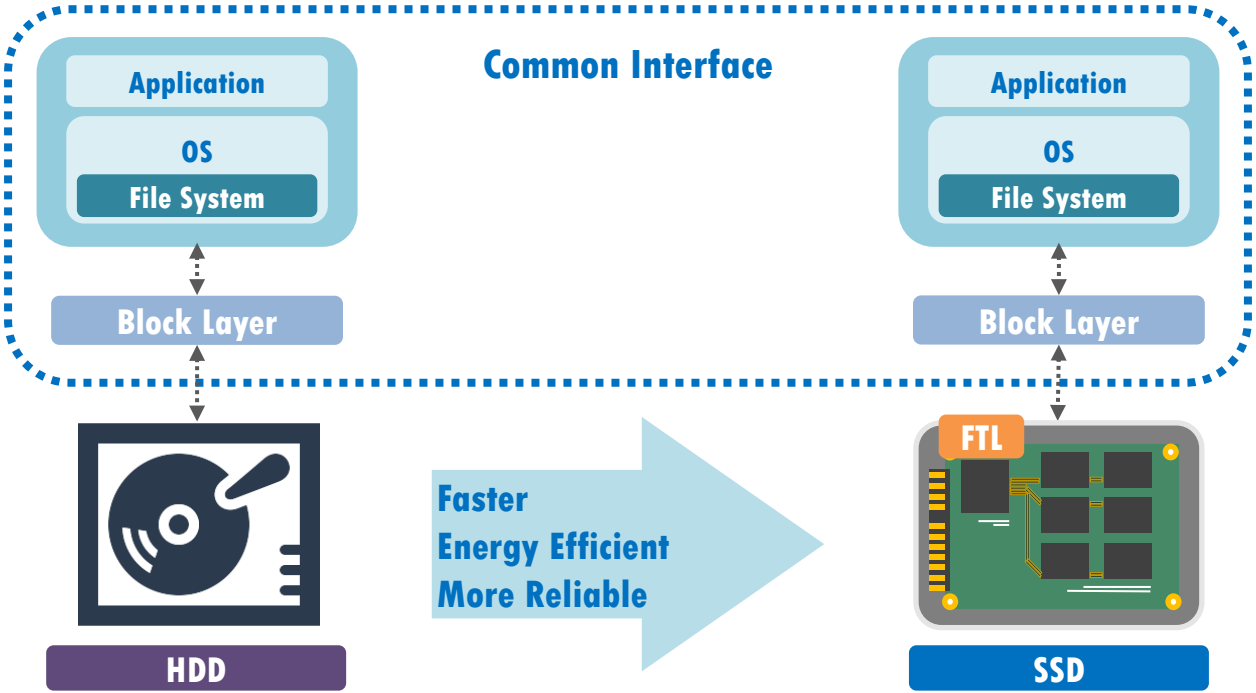
Memory Division,
Samsung Electronics Co., Ltd.

Table of Contents

- Flash-based SSDs
- Garbage Collection & WAF
- Multi-stream
- FStream
- Workload Analysis & Experimental Results
- Conclusion

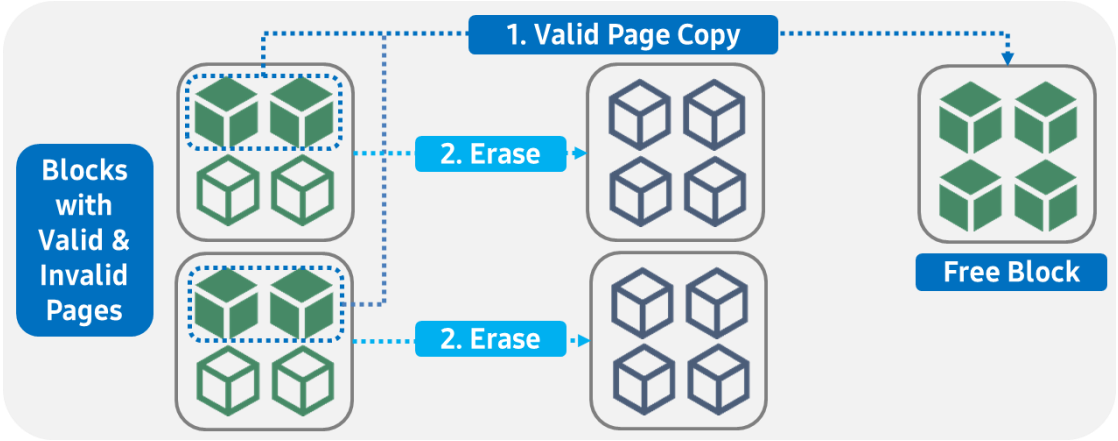
Flash-based Solid State Drives

- Replacement of HDDs
  - Flash Translation Layer (FTL) allows SSDs to maintain traditional block interface



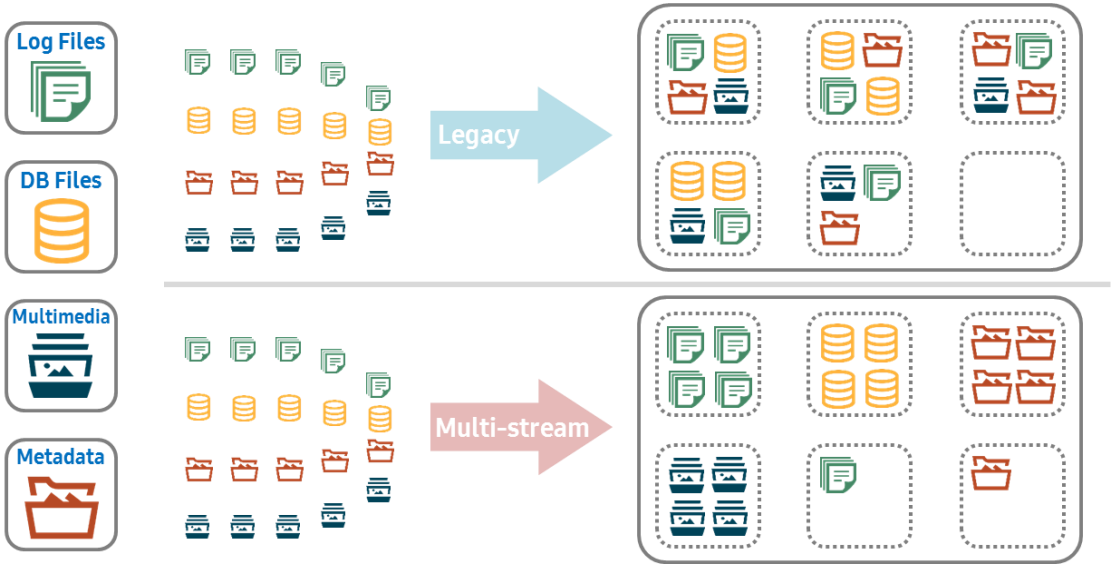
Garbage Collection & WAF

- **Garbage Collection (GC) Overheads**
  - Reclaiming space for empty blocks requires **valid page copy**
  - Media write **amplified** due to garbage collection
  - Shortens **SSD lifetime** and hampers **performance**
- **Write Amplification Factor (WAF)**
  - Ratio of the actual media writes to the user I/O



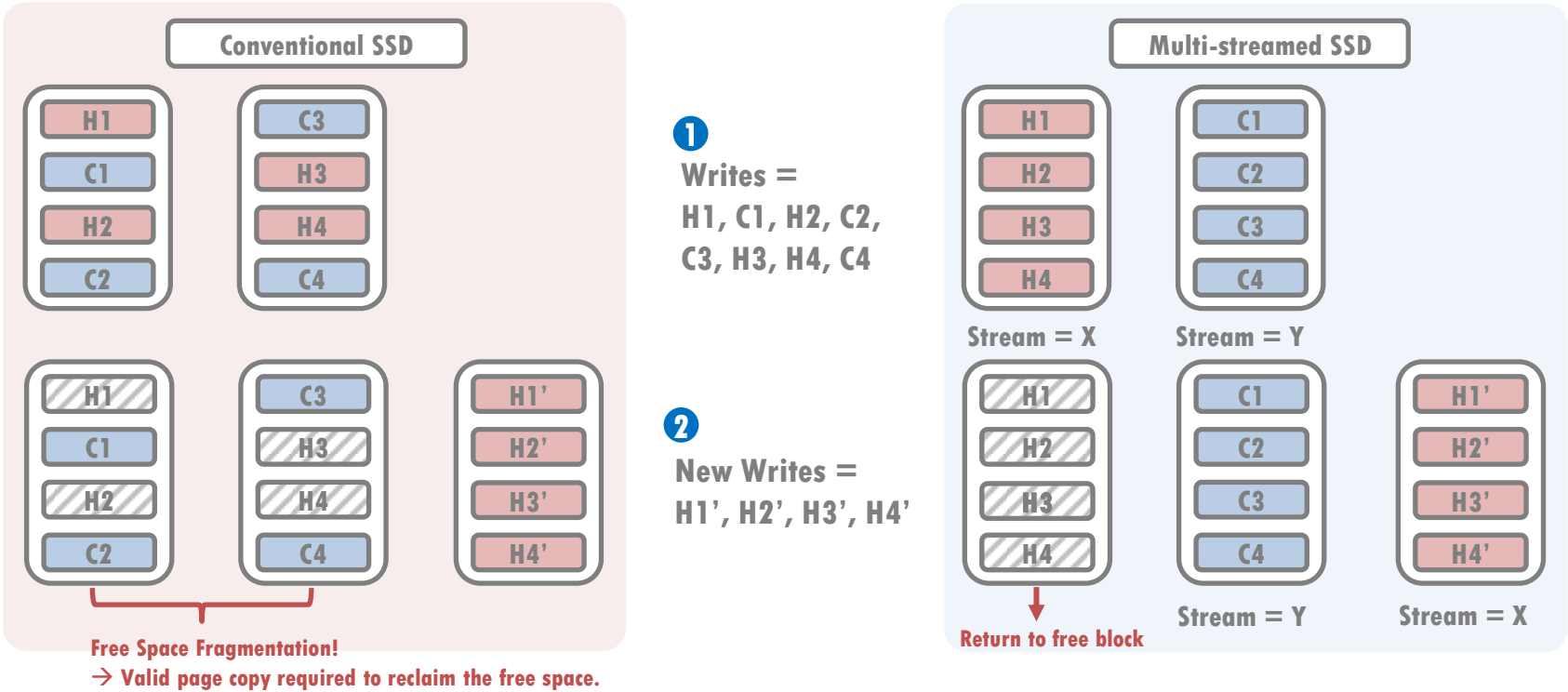
Multi-stream

- Managing data placement on a SSD with streams
  - Mapping data to separate stream by their **life expectancy**
- Standardization status
  - T10 (SCSI) standard & NVME 1.3 “directives”



Multi-stream Cont'd

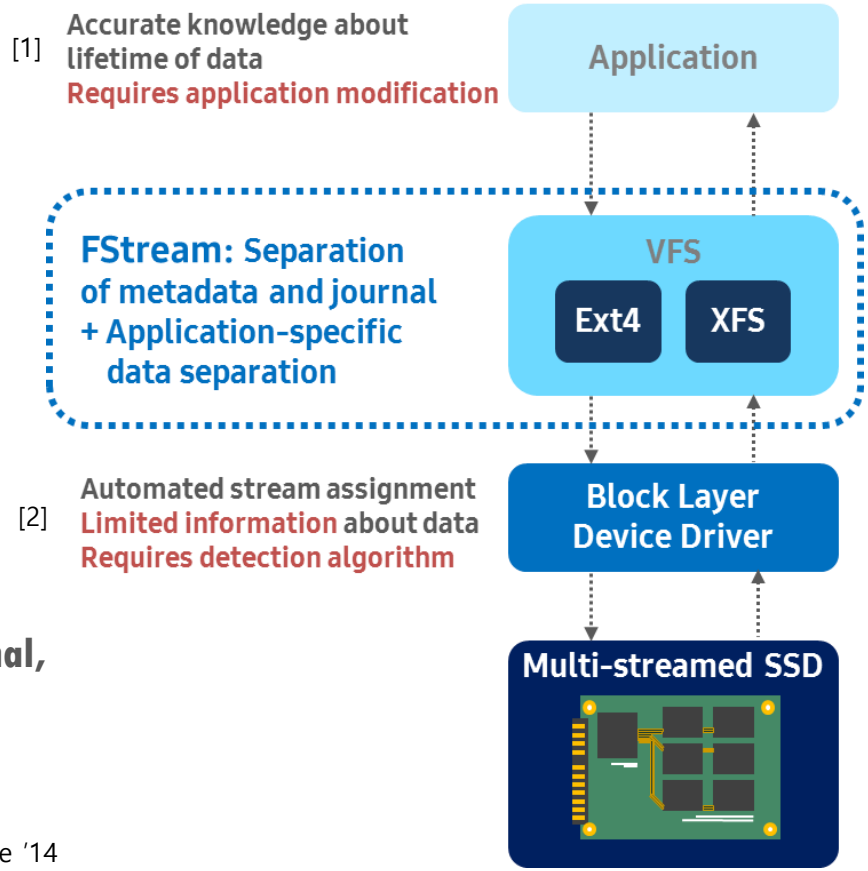
- Data Placement Comparison



FStream

- **Motivation**
  - We need easier, general method of stream assignment.
  - Block device layer has limited information about data lifetime.
  - File system metadata has different lifetime from user data, need be separated.

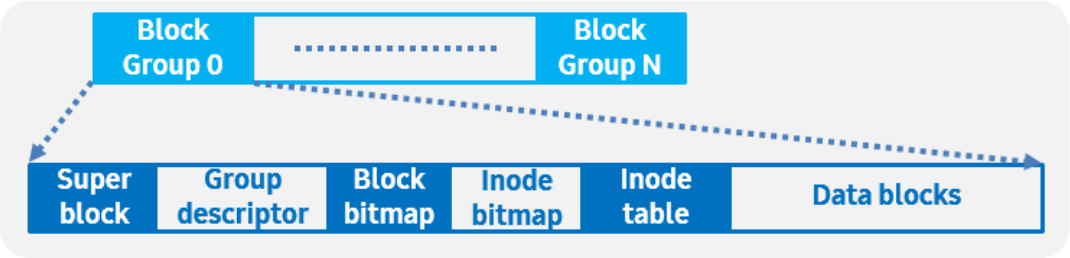
- **Our Approach**
  - File system level stream assignment.
  - Separate streams for file system metadata, journal, and user data.
  - Implemented FStream in existing file systems.



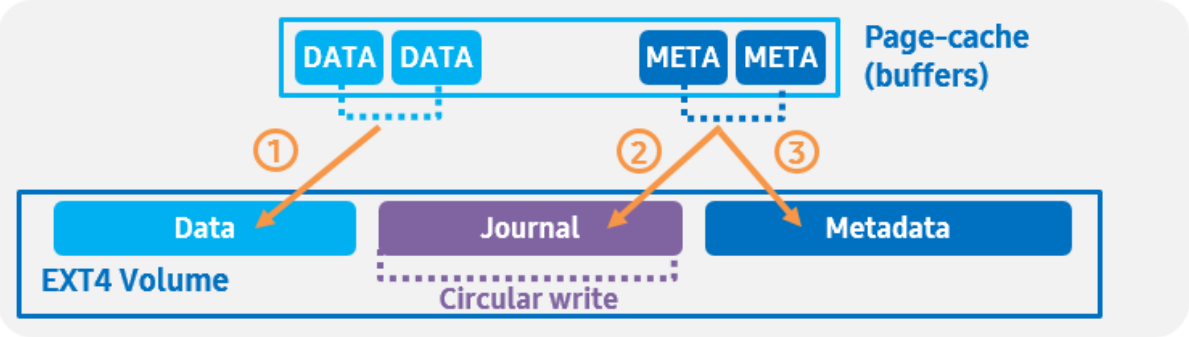
[1] Kang, JU et al., "The Multi-streamed Solid-State Drive", HotStorage '14
[2] Yang, Jingpei et al., "AutoStream: automatic stream management for multi-streamed SSDs", SYSTOR '17

Ext4

- EXT4 metadata and journaling
  - EXT4 on-disk layout: block groups with data and metadata related to it



  - EXT4 journal: write ordering in 'data=ordered' mode



Ext4Stream

- Mount options

Journal-stream
- Separate journal writes

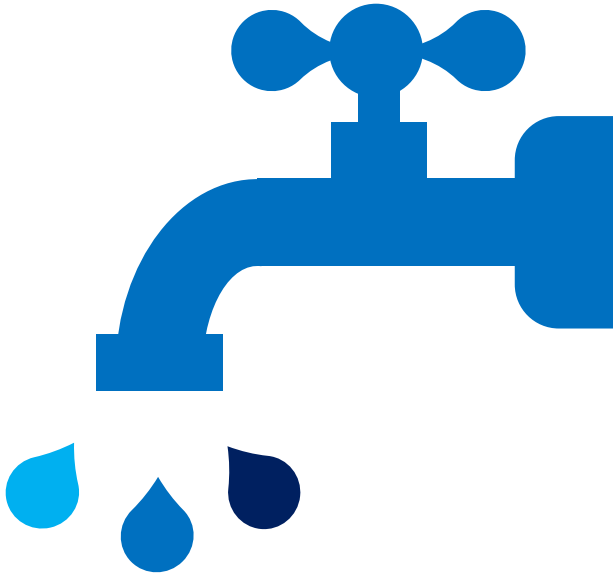
Inode-stream
- Separate inode writes

Dir-stream
- Separate directory blocks

Misc-stream
- Inode/block bitmap and group-descriptor

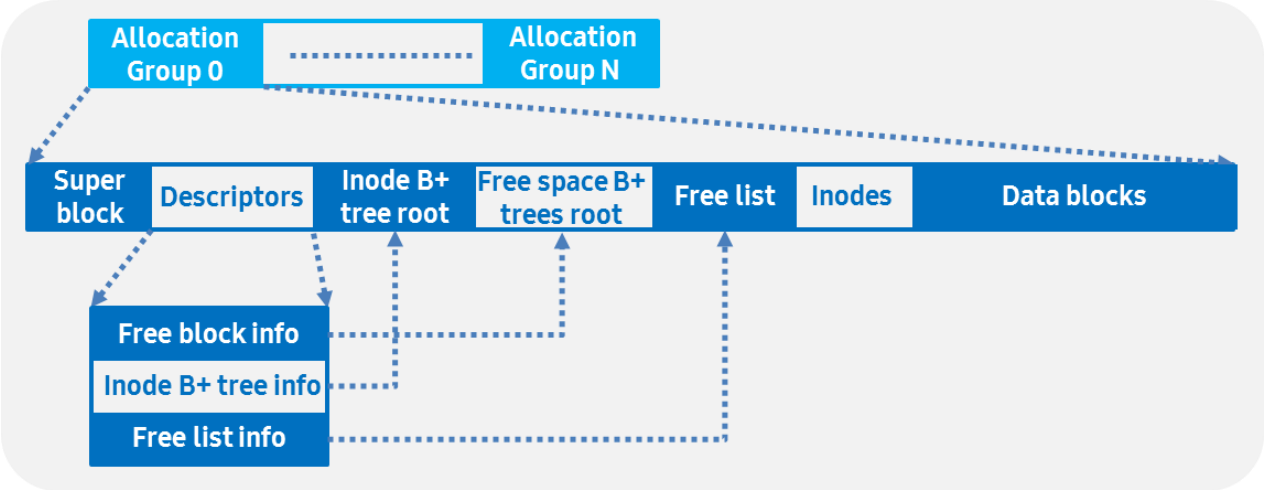
Fname-stream
- Distinct stream to file(s) with specific names

Extn-stream
- File-extension based stream



XFS

- XFS metadata and journaling
  - Parallel metadata operations, metadata buffering (page cache not used)
  - Mixture of logical and physical journaling
  - Minimum inode update size is a chunk of 64 inodes.



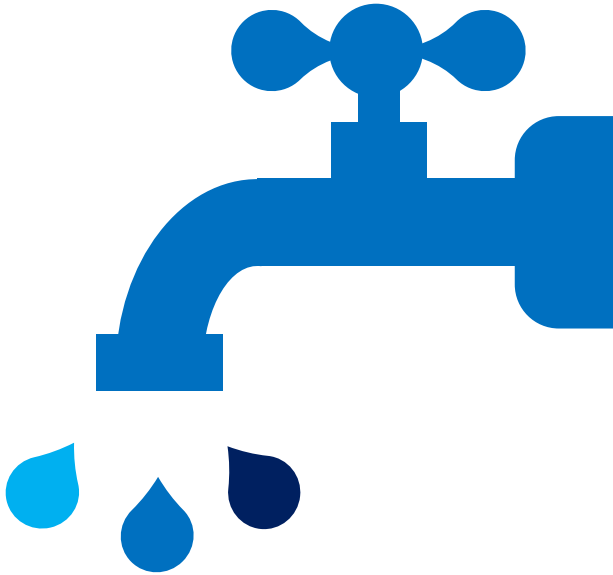
XFStream

- Mount options

Log-stream
  - Separate journal writes

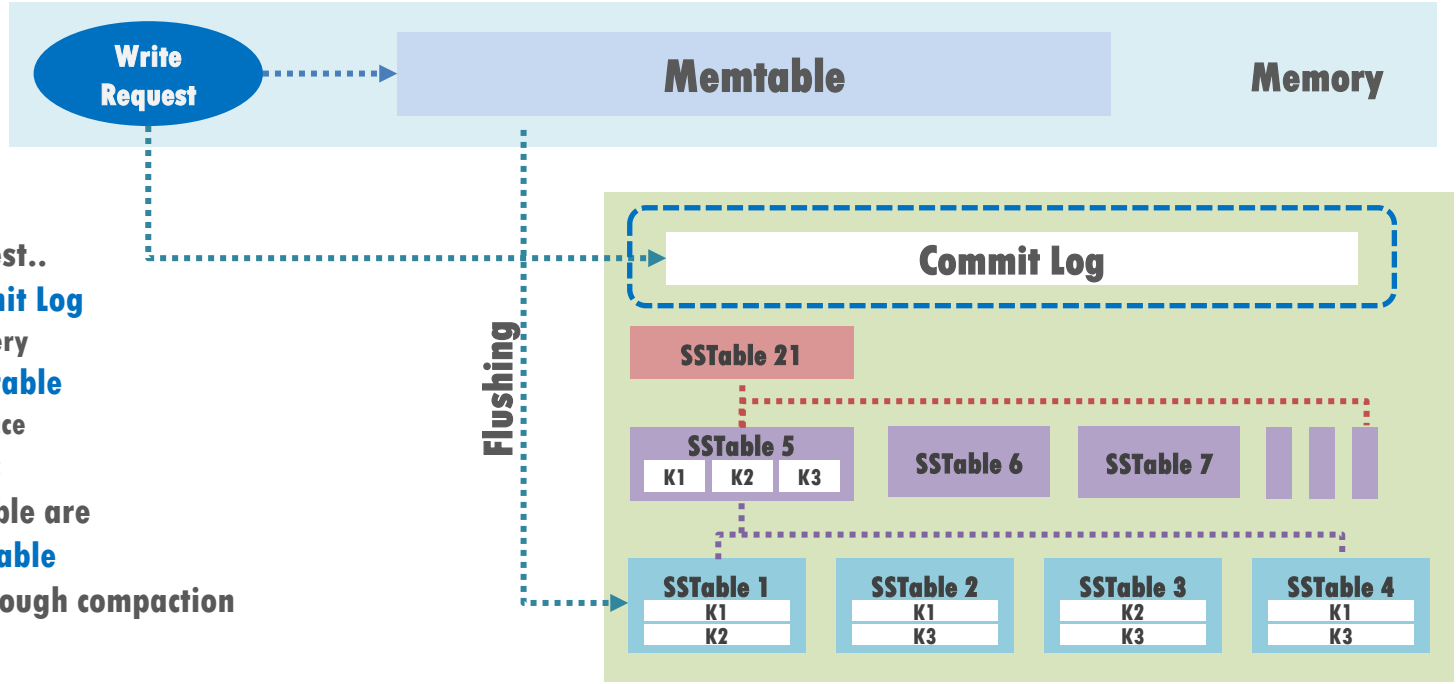
Inode-stream
  - Separate inode writes

Fname-stream
  - Distinct stream to file(s) with specific names



Application Specific Data Separation

- Stream for Cassandra's commit log file.
  - **Fname_stream** option: commitlog-*



Upon Write Request..

1) Writes to **Commit Log**
   - For error recovery
2) Writes to **Memtable**
   - In memory space
3) Returns success
4) Data in Memtable are flushed to **SSTable**
5) SSTables go through compaction

Experimental Setup

OS:

- Linux kernel v4.5 with io-streamid support

System:

- Dell PowerEdge R720 server with 32 cores and 32GB memory

SSD:

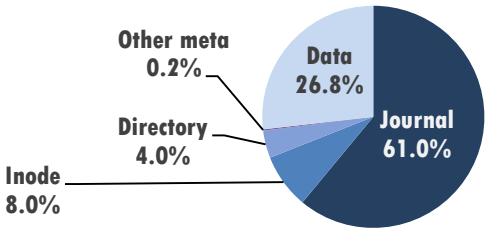
- Samsung PM963 480GB with streams support

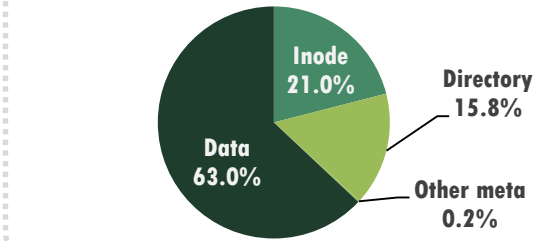
Benchmarks:

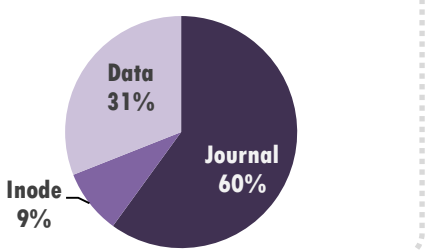
- Filebench: Varmail & Fileserver
- YCSB on Cassandra

Filebench Workload Analysis

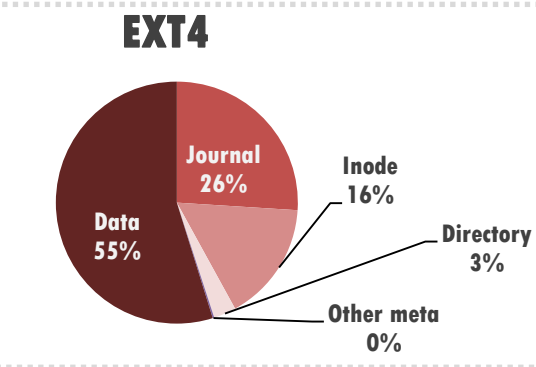
- Varmail

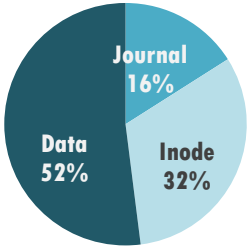






- Fileserver





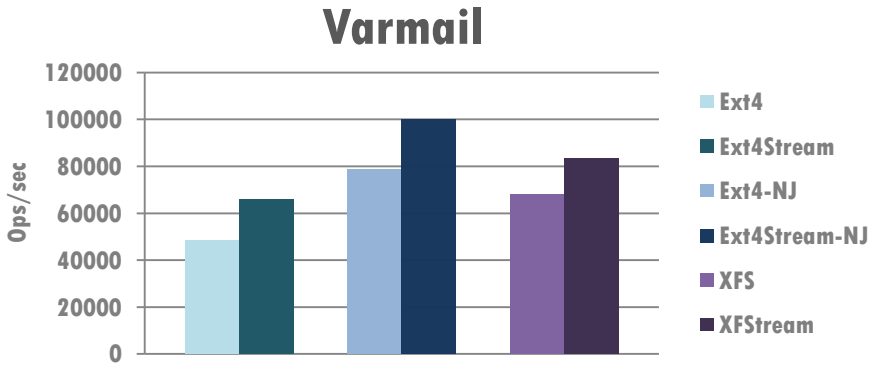
XFS writes more inodes (random writes) than Ext4.

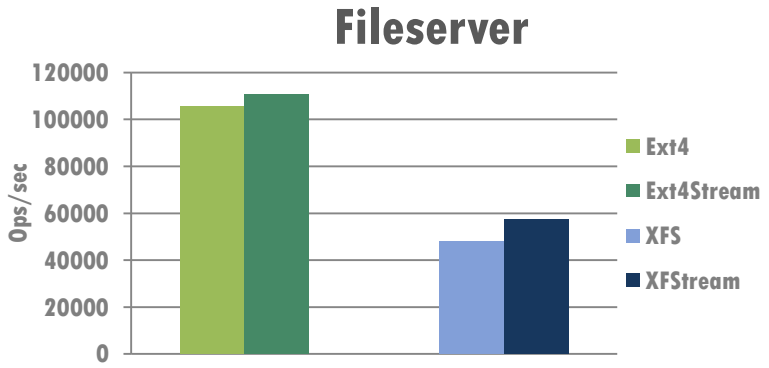
Workload Conditions

- Filled 80% of disk before test
- Number of test files: 900,000 (14GB)
- Varmail : fsync-intensive
- Runtime: 2 hours

Filebench: Performance

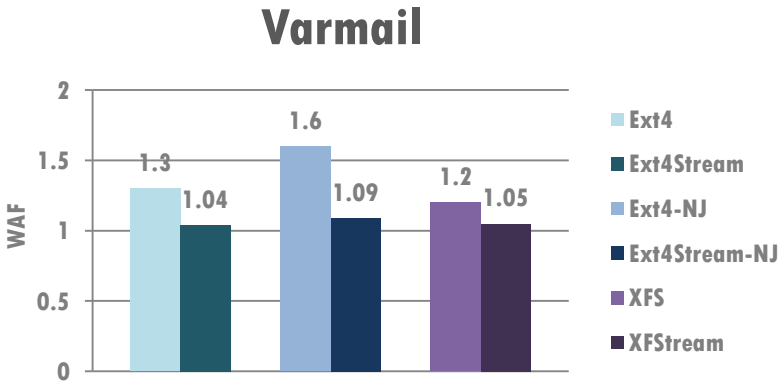
- Fstream achieved 5 ~ 35% performance improvements.

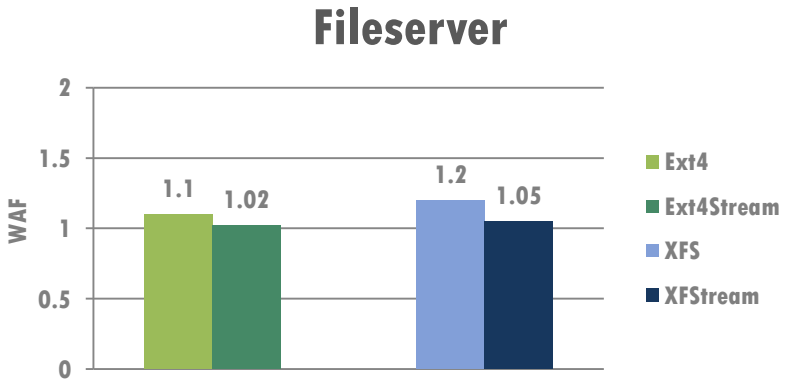




Filebench: WAF

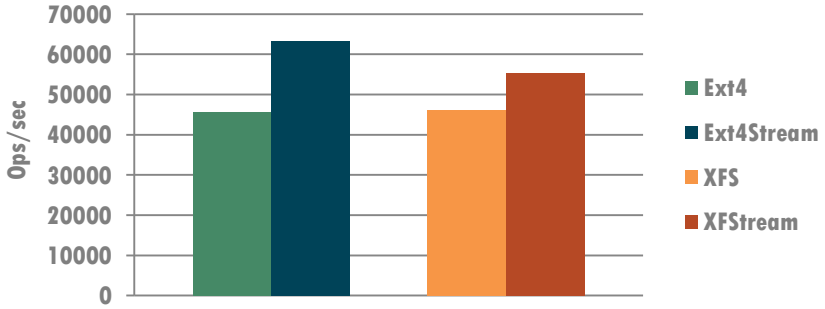
- Fstream achieved WAF of close to one.
- Ext4's WAF < Ext4NJ's WAF
  - Journal is written in a circular fashion, so is invalidated periodically.

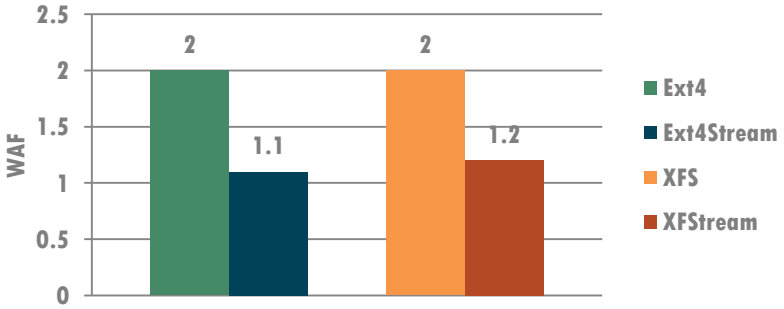




YCSB on Cassandra Results

- Data intensive workload
  - Load phase: 1KB record x 120 million inserts
  - Run phase: 1KB record x 80 million inserts





Conclusion and Acknowledgements

- **SSD Performance & Lifetime**
  - The less FTL garbage collection overheads, the longer SSD lives and the faster SSD performs.
- **Streams: SSD interface for separating data with different lifetimes**
- **FStream: stream assignment in file system**
  - Separate streams for file system metadata, journal, and user data.
  - Provide filename and extension based user data separation.
  - Achieved 5~35% performance improvement and near 1 WAF for filebench.
- **Acknowledgements**
  - We thank Cristian Ungureanu, our shepherd, and anonymous reviewers for their feedbacks.


THANK
YOU