中国科学院软件研究所学术年会 2025 第十届
暨重点实验室科技活动周

学术论文

Beaver:一种基于PM-DRAM的高性能崩溃一致性文件系统缓存方法

潘庆霖, 齐冀*, 何家泰, 张珩, 于佳耕, 武延军

中国科学院软件研究所

panqinglin2020@iscas.ac.cn

ACM SIGMETRICS 2025
Best Student Paper.

Motivation

Background

- File systems gain high performance from the memory cache.
- Read/Write requests are cached in the memory cache and asynchronously readahead/writeback from/into disk.

Issues

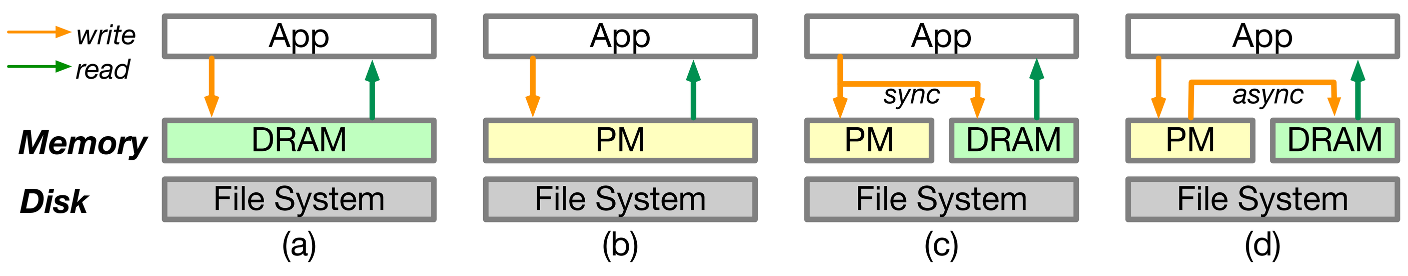
- The file system cache lacks crash-consistency due to its unorder writeback.
- Existing approaches introducing PM(Persistent Memory) for crash-consistency hurts read or write performance.

Ad-hoc workarounds

- Double write buffer in MySQL.
- Full page write in PostgreSQL.

Problem: How to build a crash-consistent file system cache that achieves both high read and write performance?

Limitations of SOTAs



Original DRAM page cache without crash-consistency

PM-only cache without high read performance from DRAM

P2CACHE[ATC'23]: State-of-the-art tiered memory with synchronous double-write policy

- The double-write policy fails when I/O size is larger than 256B
- The double-write policy has neither a hardware guarantee nor software detection methods
- Our experiment shows that the failure causes **2x lower** write throughput degradation than single write to PM/DRAM

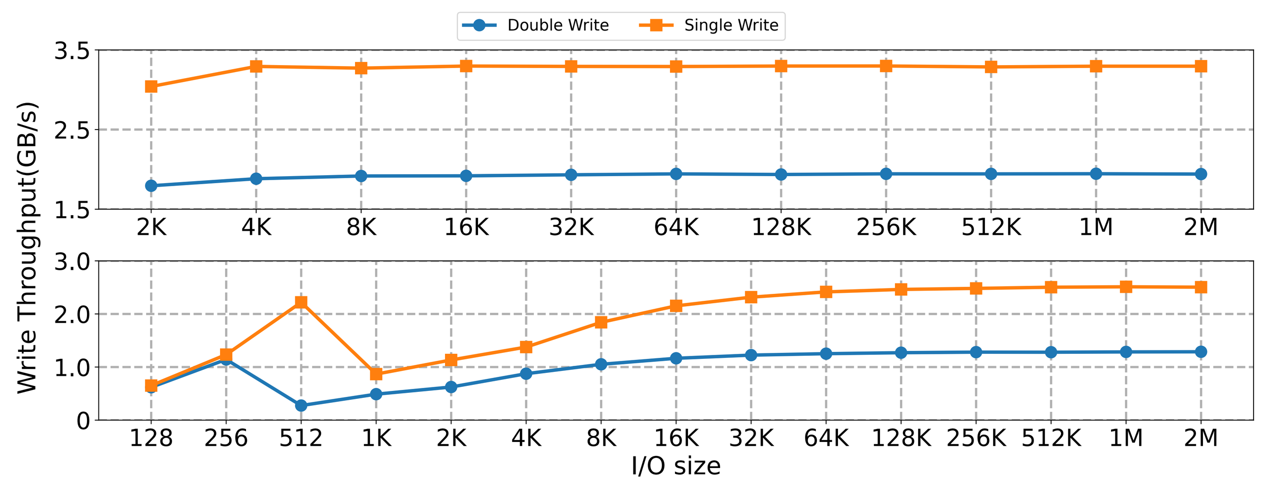


Figure 1. Write policy comparison on P²CACHE with SIMD acceleration on (upper) and off (lower).

	Read Performance	Write Performance	Crash-Consistency
Page Cache	High	High	No
PM-only Cache	Low	High	Yes
P²CACHE	High	Low	Yes
Beaver (this study)	High	High	Yes

Observation and Insight

Our Insight: Writing data to only the PM and asynchronously moving data from PM to DRAM is sufficient to satisfy the application's read requirements.

Preliminary Experiment: A distinct gap exists between write and read

- (Blue lines) Most gaps are longer than 0.1s
- (Red line) Moving a 4KB page from PM to DRAM takes only several micro-seconds

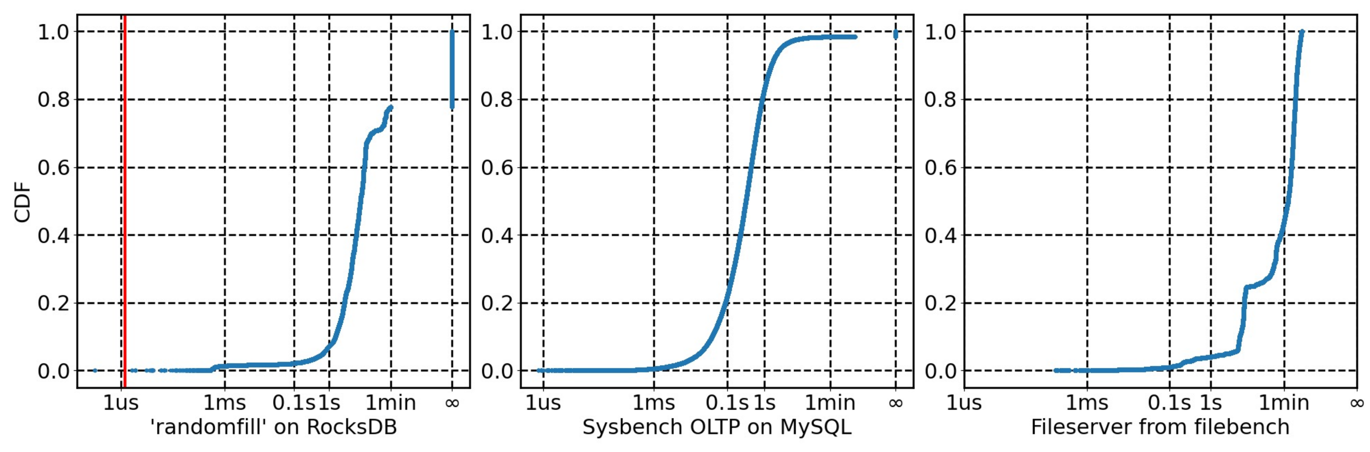


Figure 2. Write/read time interval of page cache when running RocksDB MySQL, and Fileserver.

Analysis

- Applications built upon file systems (e.g., RocksDB and MySQL) coordinate internal cache and file system cache for data caching, caching older data in file system cache in batches.
- In services built on file systems (e.g., file server and video server), writes and reads targeting the same file are often located in different requests.

Beaver's challenges

Challenge1: The workload can be skewed

- Hot data is created later but read more frequently

Challenge2: Asynchronous data movement causes PM bandwidth interference

- Foreground write operation: user buffer -> PM
- Background data movement: PM -> DRAM
- Interference can lead to severe performance degradation
  - 12.1 to 27.4% in write performance
  - 26.6 to 63.7% in read performance

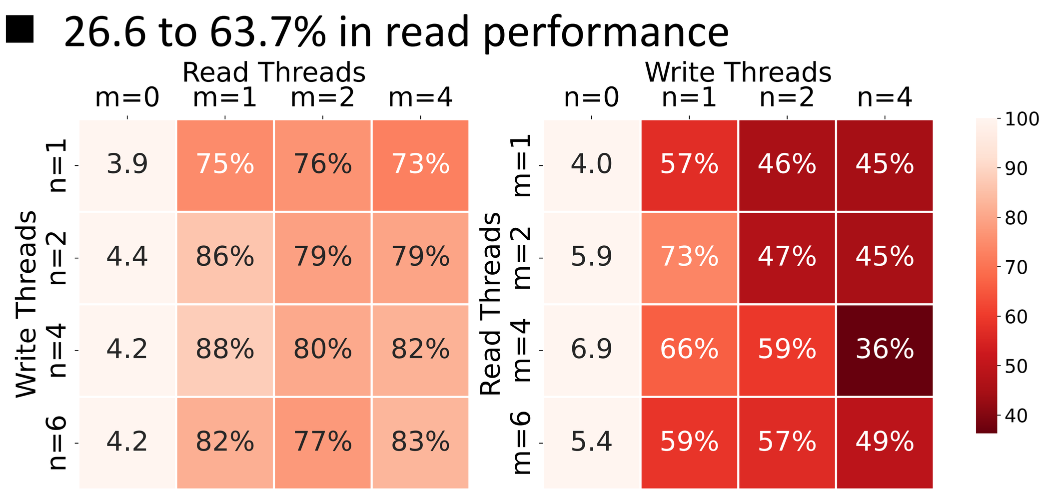
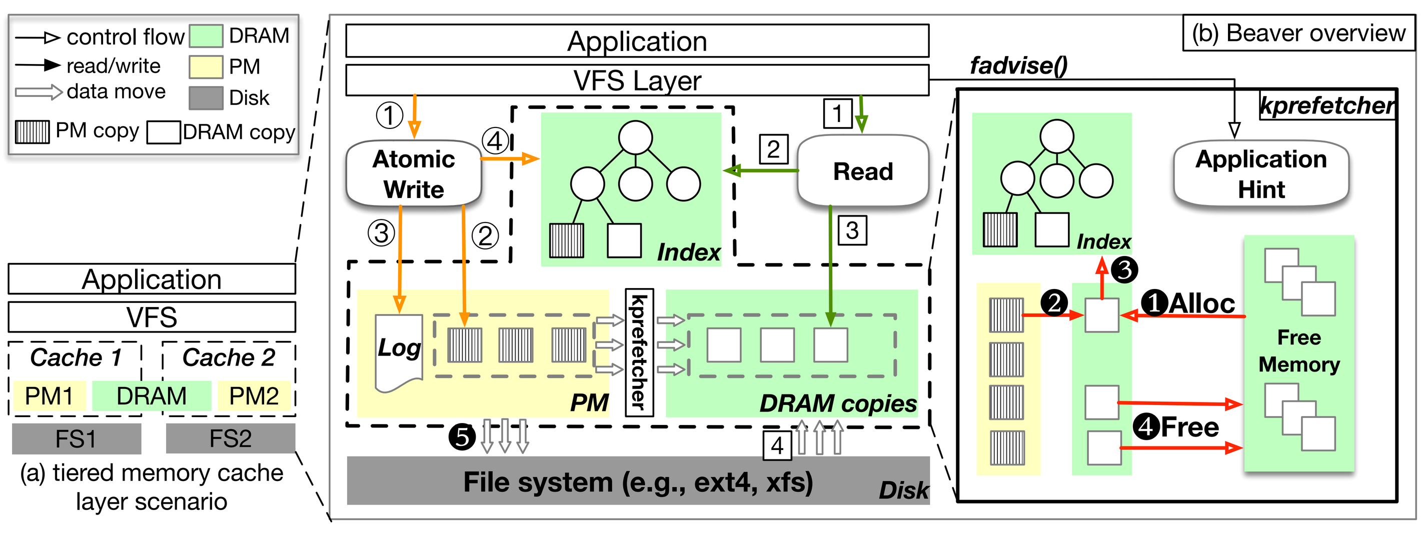


Figure 3. The mutual impact of memory interference of PM read/write tasks.

Beaver's workflow



✓ **Atomic Write: Synchronously write PM with crash-consistency**

✓ **Read: Asynchronously accelerate read via PM-DRAM data moving**

✓ **Kprefetcher: new daemon for data moving**

Solution1: Collaborative recognition + fadvise() support

- Collaboratively recognizing hot data when accessing read/write requests
- Applications are encouraged to hint at complicated hot data patterns by a fadvise() call

Solution 2: Lightweight interference mitigation mechanism

- Two atomic values: pending_wr and finished_wr
- Foreground write: update two values sequentially
  - Before writing: increase pending_wr
  - After writing: increase finished_wr
- Background data movement: tracking gap between two values
  - The gap is large => the PM workload is heavy
  - Limit moving speed when the PM workload is heavy

Implementation and Evaluation

Implementation

- A kernel module (on Linux 6.1) that works between the VFS layer and disk filesystems.

Baselines

- (DRAM) Default page cache on disk filesystem
- (PM) NOVA filesystem representing as a crash-consistent PM-only cache
- (PM) Ext4 filesystem with DAX feature on
- Operates on PM without crash-consistency

Workloads

- Microbenchmarks by FIO
- Real-world applications: RocksDB and MinIO

Key Results

- Better `insert` throughput than SOTAs
- Similar or better `read` throughput than SOTAs

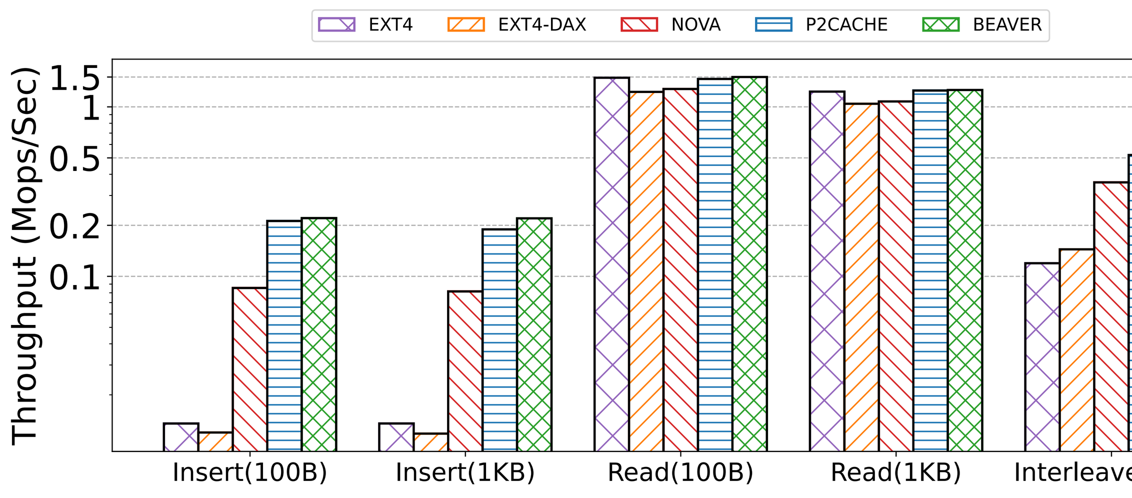


Figure 4. Performance comparison of using RocksDB

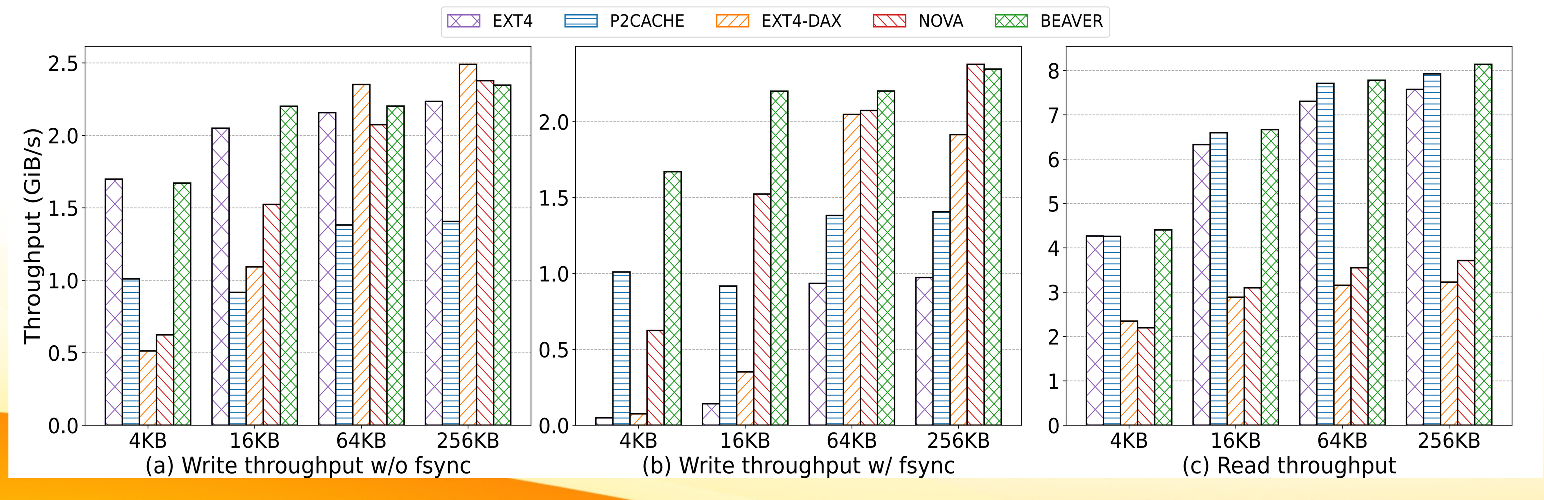


Figure 5. End-to-end performance comparison between Beaver and state-of-the-art cache systems.

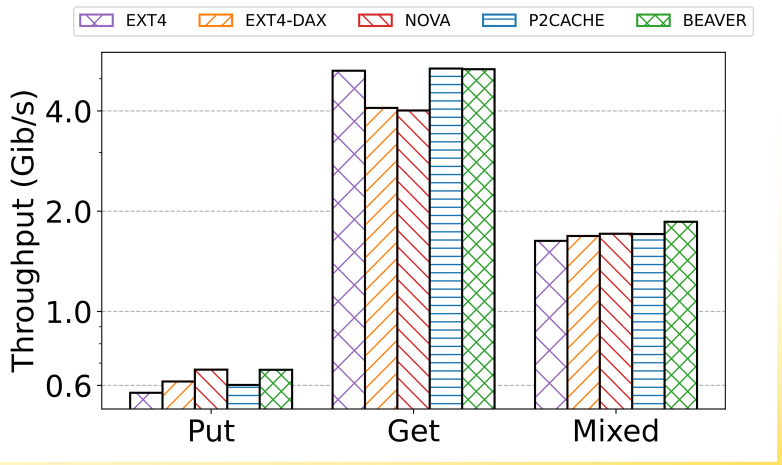


Figure 6. Performance comparison of using MinIO