FlashTier: A Lightweight, Consistent and Durable Storage Cache

Mohit Saxena, Michael M. Swift and Yiyang Zhang

University of Wisconsin-Madison

EuroSys’12
Outline

• Introduction
• Motivation
• System Design
• Evaluation
• Related Work
• Conclusion
### Flash

<table>
<thead>
<tr>
<th>Device</th>
<th>Access Latency</th>
<th>Capacity</th>
<th>Price $/GB</th>
<th>Endurance Erase Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read</td>
<td>Write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAM</td>
<td>50 ns</td>
<td>50 ns</td>
<td>8-16 GB</td>
<td>$15</td>
</tr>
<tr>
<td>Flash</td>
<td>40-100 μs</td>
<td>60-200 μs</td>
<td>TB</td>
<td>$3</td>
</tr>
<tr>
<td>Disk</td>
<td>500-5000 μs</td>
<td>500-5000 μs</td>
<td>TB</td>
<td>$0.3</td>
</tr>
</tbody>
</table>

- Flash’s price and performance are between DRAM and disk.
- write endurance: a single MLC flash cell can only be erased 10,000 times.
**Introduction**

- Flash’s 2 characteristics:

1. Flash does not support *in-place writes*.

2. Flash devices use *address mapping* to translate block addresses.
Introduction

• Caches have at least 3 different behaviours that distinguish them from general-purpose storage

1. Data in a cache may be present elsewhere in the system.

2. A cache stores data from a separate address space.

3. A cache must ensure it never returns stale data.
Introduction

OverView

• FlashTier’s 3 features

1. A unified address space
2. Cache consistency guarantees
3. Reduce the cost of garbage collection

• Results: fewer memory usages, erase cycles, recovery time.
Motivation

• Address Space Density.

• Persistence and Cache Consistency.

• Wear Management.
Motivation

- Address Space Density.

Motivate a change in how mapping information is stored.
Motivation

- Persistence and Cache Consistency.
  - large caches and poor disk performance result in long cache warming periods.
  - require storing cache metadata

- Wear Management.
  - garbage collection is often a contributor to wear.
• FlashTier is a block-level caching system.

• A *cache manager* interposes above the disk device driver.

---

**Figure 2. FlashTier Data Path:** A cache manager forwards block read/write requests to disk and solid-state cache.
System Design

- Unified Address Space
- Consistent Cache Interface
- Free Space Management
- Cache Manager
System Design

< General purpose SSD >

1. Look-up
2. Request (Hit)
2. Request (Miss)

LBA-d to LBA-s

SSD address space (LBA-s)

LBA-s to PA (Physical Address)

SSD

Cache manager

HDD address space (LBA-d)

HDD

< Solid State Cache >

Cache manager

1. Request
2. Result (Hit/Miss)
3. If missed

LBA-d to LBA-s

SSC

Same as LBA-d

LBA-d to PA

HDD

HDD address space (LBA-d)
System Design

- Unified Address Space
  - Sparse Mapping
    (google sparse hash map)
  - keep entire mapping in memory.

  hybrid FTL   4KB Page

  256KB Block
- Block State

SSC maintains:

1. the state of all flash blocks for GC.

2. usage statistics to guide wear-leveling and eviction store in out of band area (access by physical address only).

3. a reverse map for fast Address Translation
System Design

• Consistent Cache Interface

- interface

  1. persist cached data across reboot or crash

  2. never return stale data because of an inconsistent mapping.
System Design

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>write-dirty</td>
<td>Insert new block or update existing block with dirty data.</td>
</tr>
<tr>
<td>write-clean</td>
<td>Insert new block or update existing block with clean data.</td>
</tr>
<tr>
<td>read</td>
<td>Read block if present or return error.</td>
</tr>
<tr>
<td>evict</td>
<td>Evict block immediately.</td>
</tr>
<tr>
<td>clean</td>
<td>Allow future eviction of block.</td>
</tr>
<tr>
<td>exists</td>
<td>Test for presence of dirty blocks.</td>
</tr>
</tbody>
</table>

1. Write-through policy

Choose victim (all clean | invalid) and erase only when remaining free blocks are insufficient.

Choose victim (all clean | invalid) and erase only when remaining free blocks are insufficient.
### System Design

| write-dirty | Insert new block or update existing block with dirty data. |
| write-clean | Insert new block or update existing block with clean data. |
| read        | Read block if present or return error. |
| evict       | Evict block immediately. |
| clean       | Allow future eviction of block. |
| exists      | Test for presence of dirty blocks. |

2. Write-back policy

Choose victim (all clean | invalid) and erase only when remaining free blocks are insufficient (Silent-evict)
## System Design

| write-dirty | Insert new block or update existing block with dirty data. |
| write-clean | Insert new block or update existing block with clean data. |
| read        | Read block if present or return error. |
| evict       | Evict block immediately. |
| clean       | Allow future eviction of block. |
| exists      | Test for presence of dirty blocks. |

2. Write-back policy

- **Clean:** SSC mapping info preserved until silent-evict occurs
- **Evict:** SSC mapping info vanishes immediately

Choose victim (all clean | invalid) and erase only when remaining free blocks are insufficient (Silent-evict)
## System Design

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>write-dirty</td>
<td>Insert new block or update existing block with dirty data.</td>
</tr>
<tr>
<td>write-clean</td>
<td>Insert new block or update existing block with clean data.</td>
</tr>
<tr>
<td>read</td>
<td>Read block if present or return error.</td>
</tr>
<tr>
<td>evict</td>
<td>Evict block immediately.</td>
</tr>
<tr>
<td>clean</td>
<td>Allow future eviction of block.</td>
</tr>
<tr>
<td>exists</td>
<td>Test for presence of dirty blocks.</td>
</tr>
</tbody>
</table>

2. **Write-back policy**

Dirty block table re-construction with ‘exist’ query
During crash recovery

---

**Exists**

- Dirty block table
- Cache manager

- LBA-d range
- Dirty data status within queried range

**SSC**
System Design

- Consistent Cache Interface
  - Persistence
    1. Logging
    2. Checkpoints
System Design

- Free hybrid
  - SSC with fixed log space: Evicted data blocks are recycled for use as data GC blocks.
  - SSC-V with variable log space: Evicted data blocks can be recycled for use as data blocks or increase the fraction of log blocks.

1. SE-Util: create erased data block but not log block

2. SE-Merge: increase log block
System Design

• Cache Manager Based on Facebook’s FlashCache

write-through:

write-clean

write-back:

1. write-dirty

2. maintain in-memory table of dirty blocks
System Design

- Implementation
  - The Cache Manager (based on FlashCache)
  - An SSC Function Emulator (based on FlashSim)
  - An SSC timing simulator

2 Basic Configuration of the Simulator

1. SSC(SE-Util)
2. SSC-R(SE-Merge)
Evaluation

• What are the benefits of providing a sparse unified cache address space for FlashTier?

• What is the cost of providing cache consistency and recovery guarantees in FlashTier?

• What are the benefits of silent eviction for free space management and write performance in FlashTier?
Evaluation

- Emulation parameters & Workloads Characteristics

<table>
<thead>
<tr>
<th></th>
<th>Page read/write</th>
<th>Block erase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>65/85 $\mu$s</td>
<td>1000 $\mu$s</td>
</tr>
<tr>
<td>Bus control delay</td>
<td>2 $\mu$s</td>
<td>Control delay</td>
</tr>
<tr>
<td>Flash planes</td>
<td>10</td>
<td>Erase block/plane</td>
</tr>
<tr>
<td>Pages/erase block</td>
<td>64</td>
<td>Page size</td>
</tr>
<tr>
<td>Seq. Read</td>
<td>585 MB/sec</td>
<td>Rand. Read</td>
</tr>
<tr>
<td>Seq. Write</td>
<td>124 MB/sec</td>
<td>Rand. Write</td>
</tr>
</tbody>
</table>

Table 2. Emulation parameters.

<table>
<thead>
<tr>
<th>Workload</th>
<th>Range</th>
<th>Unique Blocks</th>
<th>Total Ops.</th>
<th>% Writes</th>
</tr>
</thead>
<tbody>
<tr>
<td>homes</td>
<td>532 GB</td>
<td>1,684,407</td>
<td>17,836,701</td>
<td>95.9</td>
</tr>
<tr>
<td>mail</td>
<td>277 GB</td>
<td>15,136,141</td>
<td>462,082,021</td>
<td>88.5</td>
</tr>
<tr>
<td>usr</td>
<td>530 GB</td>
<td>99,450,142</td>
<td>116,060,427</td>
<td>5.9</td>
</tr>
<tr>
<td>proj</td>
<td>816 GB</td>
<td>107,509,907</td>
<td>311,253,714</td>
<td>14.2</td>
</tr>
</tbody>
</table>

Table 3. Workload Characteristics: All requests are sector-aligned and 4,096 bytes.
Evaluation

• System Comparison

1. Performance

2. Memory Consumption

Figure 3. Application Performance: The performance of write-through and write-back FlashTier systems normalized to native write-back performance. We do not include native write-through because it does not implement durability.
Evaluation

• FlashTier Address Space Management

1. Device memory usage (native LBA_s-PA, SSC LBA_d-PA)

2. Host memory usage

<table>
<thead>
<tr>
<th>Workload</th>
<th>Size (GB)</th>
<th>SSD Device (MB)</th>
<th>SSC Device (MB)</th>
<th>SSC-R Device (MB)</th>
<th>Native Host (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>homes</td>
<td>1.6</td>
<td>1.13</td>
<td>1.33</td>
<td>3.07</td>
<td>8.83 0.96</td>
</tr>
<tr>
<td>mail</td>
<td>14.4</td>
<td>10.3</td>
<td>12.1</td>
<td>27.4</td>
<td>79.3 8.66</td>
</tr>
<tr>
<td>usr</td>
<td>94.8</td>
<td>66.8</td>
<td>71.1</td>
<td>174</td>
<td>521 56.9</td>
</tr>
<tr>
<td>proj</td>
<td>102</td>
<td>72.1</td>
<td>78.2</td>
<td>189</td>
<td>564 61.5</td>
</tr>
<tr>
<td>proj-50</td>
<td>205</td>
<td>144</td>
<td>152</td>
<td>374</td>
<td>1,128 123</td>
</tr>
</tbody>
</table>

Table 4. Memory Consumption: Total size of cached data, and host and device memory usage for Native and FlashTier systems for different traces. FTCM: write-back FlashTier Cache Manager.
Evaluation

• FlashTier Consistency

1. Consistency Cost

2. Recovery Time

Figure 4. Consistency Cost: No-consistency system does not provide any consistency guarantees for cached data or metadata. Native-D and FlashTier-D systems only provide consistency for dirty data. FlashTier-C/D provides consistency for both clean and dirty data.
Evaluation

- FlashTier Consistency
  1. Consistency Cost
  2. Recovery Time

**Figure 5.** Recovery Time: Native-FC accounts for only recovering FlashCache cache manager state. Native-SSD accounts for only recovering the SSD mapping.

A 3-5% increase in average response times for these workloads respectively. Overall, the extra cost of consistency for the request response time is less than 26 μs for all workloads with FlashTier.
Evaluation

- FlashTier Silent Eviction

1. Garbage Collection
2. Cache Misses
3. Wear Management
Evaluation

- FlashTier Silent Eviction

<table>
<thead>
<tr>
<th>Workload</th>
<th>Erases</th>
<th>Wear Diff.</th>
<th>Write Amp.</th>
<th>Miss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SSD</td>
<td>SSC</td>
<td>SSC-R</td>
<td>SSD</td>
</tr>
<tr>
<td>homes</td>
<td>878,395</td>
<td>829,356</td>
<td>617,298</td>
<td>3,094</td>
</tr>
<tr>
<td>mail</td>
<td>880,710</td>
<td>637,089</td>
<td>525,954</td>
<td>1,044</td>
</tr>
<tr>
<td>usr</td>
<td>339,198</td>
<td>369,842</td>
<td>325,272</td>
<td>219</td>
</tr>
<tr>
<td>proj</td>
<td>164,807</td>
<td>166,712</td>
<td>164,527</td>
<td>41</td>
</tr>
</tbody>
</table>

Table 5. Wear Distribution: For each workload, the total number of erase operations, the maximum wear difference between blocks, the write amplification, and the cache miss rate is shown for SSD, SSC and SSC-R.
Related Work

• SSD Caches:
  FlachCache, dm-cache, bcache

• Hybrid Systems:
  FushionIO, Synapse (OCZ)

• Informed Caching

• Storage Interfaces
• Flash caching promises an inexpensive boost to storage performance.

• FlashTier provides memory-efficient address space management, improved performance and cache consistency to quickly recover cached data following a crash.

- reflection: Reduce GC in Common SSD Cache.