AutoStream: Automatic Stream Management for Multi-stream SSDs

Jingpei Yang, PhD, Rajinikanth Pandurangan, Changho Choi, PhD, Vijay Balakrishnan

Memory Solutions Lab
Samsung Semiconductor
Agenda

• SSD NAND flash characteristics
• Multi-stream
• Autostream: Automatic stream management
  – Multi-Q
  – SFR
• Performance enhancement
• Summary
SSD NAND Flash Characteristics

- **Different IO units**
  - Read/Program: Page, Erase: Block (=multiple of pages)

- **Erase before program**
  - Out-of-place update

- **Unavoidable GC overhead**
  - The higher GC overhead, the larger Write Amplification*(= the lower endurance)

- **Limited number of Program/Erase cycles**

\[ WAF = \frac{\text{amount of data written to NAND Flash}}{\text{amount of data written by host}} \]

To maximize SSD lifetime, need to minimize Write Amplification!
Multi-stream: Minimize Write Amplification

- Store similar lifetime data into the same erase block and reduce WA (GC overhead)
- Provide better endurance and improved performance
  - Host associates each write operation with a stream
  - All data associated with a stream is expected to be invalidated at the same time (e.g., updated, trimmed, unmapped, deallocated)
  - Align NAND block allocation based on application data characteristics (e.g., update frequency)
AutoStream: Automatic Stream Management

- **Multi-stream shows good benefit but requires application and system modification**
  - More challenges in multi-application, multi-tenant environments (e.g., VM or Docker)
- **AutoStream**
  - Make stream detection independent of applications (e.g., in device driver)
  - Cluster data into streams according to data update frequency, recency and sequentiality
  - Minimize stream management overhead in application and systems

---

**Multi-stream**

Applications manage streams

App. & Kernel modification req’d

Stream sync overhead

---

**AutoStream**

Automatic stream management based on data characteristics

No app. & Kernel modification required
AutoStream IO Processing with Minimal Overhead

READ I/O bypass AutoStream

WRITE I/O just one table look up
AutoStream Implementation

AutoStream controller

Multi-Q queue update

SFR table update

Submission queue

AutoStream module

Multi-stream SSD

Multi-stream SSD

Device driver

Block layer

File system

OS kernel

application

Write <sLBA, sz>

Write <sLBA, sz, sID>

1 <sLBA, sz>

2 <sLBA>

3 <sID>
Multi-Q Algorithm Basics

- Divide a whole SSD space into the same size chunks
  - 480GB SSD, 1MB chunk size -> 480,000 chunks
- Track statistics for each chunk
  - access time, access count, expiry time, etc.
  - Expiry time
    - *hottest chunk’s lifetime := current time – last access time*
    - *Other chunk’s expiry time:= current time + hottest chunk’s lifetime*

<table>
<thead>
<tr>
<th>chunk id</th>
<th>......</th>
<th>c</th>
<th>c</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>u</th>
<th>w</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>access time</td>
<td>......</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>access count</td>
<td>......</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Access time 11: Hottest chunk = c
Chunk c’s lifetime = 11 – 5 = 6

Access time 12: chunk d expiry time = 18 (12+6)
Multi-Q Update (Promotion & Demotion)

Submission Q

- Multi-Q thread processes each entry

Promotion

| a | b | c |

Demotion

| e | f | g |

Chunk e’s expiry time has passed (recency*)

Chunk a’s access count is bigger than Q1’s access count threshold (frequency)

Q1 cold

Q2

... Q7

Q8 hot

* Recency considers the last updated time
SFR - Sequentiality Frequency Recency Algorithm

AutoStream controller

\[ \langle sLBA, sz \rangle \]

Sequential write?

- yes
  - \( sID := \text{prev\_sID} \)
  - Get \( sID \) from stream table
  - Update \( \text{prev\_sID} \)
  - Put \( sLBA \) to submission queue

- no
  - Get \( sID \) from stream table
  - Update \( \text{prev\_sID} \)
  - Put \( sLBA \) to submission queue

Submission Q

SFR thread processes each entry

- Increase \( \text{access\_cnt} \)
- Calculate \( \text{recency\_weight} := \text{pow}(2, (\text{curr\_time} - \text{last\_access\_time})/\text{decay\_period}) \)
- \( \text{access\_cnt} := \text{access\_cnt}/\text{recency\_weight} \)
- \( sID := \log(\text{access\_cnt}) \)

Stream table update (Frequency, Recency)
Docker Environment Performance Measurement

- Running 2 MySQL & 2 Cassandra instances simultaneously

<table>
<thead>
<tr>
<th>Database</th>
<th>Size/Workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL TPC-C</td>
<td>800 warehouse TPC-C: 30 connection</td>
</tr>
<tr>
<td>Cassandra -Stress</td>
<td>1KB record, 100 million entries r/w: 50/50</td>
</tr>
</tbody>
</table>

MySQL average tpmC

- Legacy: 42%
- SFR: 39%
- MQ: 6%

Cassandra average TPS

- Legacy: 21000
- SFR: 22500
- MQ: 23000
Summary

• **AutoStream**
  – With no application and system modification, improve SSD lifetime and performance

• **AutoStream with minimal overhead**
  – Works well under different workloads for diverse applications on various system environments
  – Up to 60% WAF reduction
  – Up to 237% performance improvement

• **Future work**
  – Optimize resource utilization and performance to fit into devices
Thank You!