AniFilter: Parallel and Failure-Atomic Cuckoo Filter for Non-Volatile Memories

Hyungjun Oh¹, Bongki Cho¹, Changdae Kim², Heejin Park¹, Jiwon Seo¹

¹ HANYANG UNIVERSITY ² ETRI
Outline

• NVM and AMQs
• Cuckoo Filter
• Optimizations in AniFilter
  • Spillable Buckets
  • Lookahead Eviction
  • Bucket Primacy
• Logging and Recovery
• Evaluation
Non-Volatile Memories

• NVM Characteristics
  • High performance
  • Persistency
  • Byte-addressability

⇒ Best of both DRAM and SSD (almost)
Approximate Membership Queries (AMQs)

- Approximate set data structures
- APIs
  - Insert(x) – inserts key x into the set
  - Lookup(x) – lookup key x and returns true or false
  - Delete(x) – removes key x from the set (optional)
- Small false-positives
  - lookup(x) true when x not in the set
NVM and AMQs

- NVMs are fast, but not as fast as DRAM
  - Read latency is 2~3x slower than DRAM
- DRAM versions of AMQs run slow on NVM

- AMQs’ operations are cheap
  - Insert() and Lookup() need only handful of computation
- Cannot use complicated optimization techniques
Cuckoo Filter

• Fingerprint-based AMQ
• Bucketized Cuckoo Filter

» Each bucket has four slots
» Two hashes ($H_1$, $H_2$) for a bucket index

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>05A</td>
<td>0F2</td>
<td>0E9</td>
<td>000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>600</td>
<td>27D</td>
<td>000</td>
<td>000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2F0</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12D</td>
<td>A15</td>
<td>1EF</td>
<td>AFE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>14A</td>
<td>FA0</td>
<td>5B0</td>
<td>000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1AA</td>
<td>5B0</td>
<td>000</td>
<td>000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cuckoo Filter

- Fingerprint-based AMQ
- Bucketized Cuckoo Filter
  - Each bucket has four slots
  - Two hashes ($H_1$, $H_2$) for a bucket index

```

<table>
<thead>
<tr>
<th>Bucket</th>
<th>05A</th>
<th>0F2</th>
<th>0E9</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>600</td>
<td>27D</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>2</td>
<td>2F0</td>
<td>000</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>3</td>
<td>12D</td>
<td>A15</td>
<td>1EF</td>
<td>AFE</td>
</tr>
<tr>
<td>4</td>
<td>14A</td>
<td>FA0</td>
<td>5B0</td>
<td>000</td>
</tr>
<tr>
<td>5</td>
<td>1AA</td>
<td>5B0</td>
<td>000</td>
<td>000</td>
</tr>
</tbody>
</table>
```
Cuckoo Filter

- Fingerprint-based AMQ
- Bucketized Cuckoo Filter
- Each bucket has four slots
- Two hashes ($H_1$, $H_2$) for a bucket index
- Insertion (without eviction)

$$H_1(x)$$

<table>
<thead>
<tr>
<th>Key x</th>
<th>05A</th>
<th>0F2</th>
<th>0E9</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>600</td>
<td>27D</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>2F0</td>
<td>000</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>2</td>
<td>12D</td>
<td>A15</td>
<td>1EF</td>
<td>AFE</td>
</tr>
<tr>
<td>3</td>
<td>14A</td>
<td>FA0</td>
<td>5B0</td>
<td>000</td>
</tr>
<tr>
<td>4</td>
<td>1AA</td>
<td>5B0</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cuckoo Filter

- Fingerprint-based AMQ
- Bucketized Cuckoo Filter
  ➔ Each bucket has four slots
  ➔ Two hashes ($H_1$, $H_2$) for a bucket index
- Insertion (without eviction)

<table>
<thead>
<tr>
<th>Key x</th>
<th>05A</th>
<th>0F2</th>
<th>0E9</th>
<th>AC0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>600</td>
<td>27D</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>2</td>
<td>2F0</td>
<td>000</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>3</td>
<td>12D</td>
<td>A15</td>
<td>1EF</td>
<td>AFE</td>
</tr>
<tr>
<td>4</td>
<td>14A</td>
<td>FA0</td>
<td>5B0</td>
<td>000</td>
</tr>
<tr>
<td>5</td>
<td>1AA</td>
<td>5B0</td>
<td>000</td>
<td>000</td>
</tr>
</tbody>
</table>
Cuckoo Filter

- Fingerprint-based AMQ
- Bucketized Cuckoo Filter
  - Each bucket has four slots
  - Two hashes \(H_1, H_2\) for a bucket index
- Insertion (with eviction)

<table>
<thead>
<tr>
<th>Key x</th>
<th>(H_1(x))</th>
<th>(H_2(x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>05A</td>
<td>0F2</td>
<td>0E9</td>
</tr>
<tr>
<td>600</td>
<td>27D</td>
<td>000</td>
</tr>
<tr>
<td>2F0</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>12D</td>
<td>A15</td>
<td>1EF</td>
</tr>
<tr>
<td>14A</td>
<td>FA0</td>
<td>5B0</td>
</tr>
<tr>
<td>1AA</td>
<td>5B0</td>
<td>000</td>
</tr>
</tbody>
</table>
Cuckoo Filter

- Fingerprint-based AMQ
- Bucketized Cuckoo Filter
  ➔ Each bucket has four slots
  ➔ Two hashes ($H_1$, $H_2$) for a bucket index
- Insertion (with eviction)
Cuckoo Filter

- Fingerprint-based AMQ
- Bucketized Cuckoo Filter
  - Each bucket has four slots
  - Two hashes ($H_1, H_2$) for a bucket index
- Insertion (with eviction)
Cuckoo Filter

- Fingerprint-based AMQ
- Bucketized Cuckoo Filter

- Each bucket has four slots
- Two hashes ($H_1, H_2$) for a bucket index
- Lookup operation

\[
\begin{array}{cccc}
05A & 0F2 & 0E9 & AC0 \\
600 & 27D & 000 & 000 \\
2F0 & 000 & 000 & 000 \\
12D & A15 & 1EF & AFE \\
14A & FA0 & 5B0 & 000 \\
1AA & 5B0 & 000 & 000 \\
\end{array}
\]
Cuckoo Filter

- Fingerprint-based AMQ
- Bucketized Cuckoo Filter
  ➔ Each bucket has four slots
  ➔ Two hashes ($H_1$, $H_2$) for a bucket index
- Lookup operation

<table>
<thead>
<tr>
<th>Key x</th>
<th>$H_1(x)$</th>
<th>$H_2(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>05A</td>
<td>0F2</td>
<td>0E9 AC0</td>
</tr>
<tr>
<td>600</td>
<td>27D</td>
<td>000 000</td>
</tr>
<tr>
<td>2F0</td>
<td>000</td>
<td>000 000</td>
</tr>
<tr>
<td>12D A15</td>
<td>1EF AFE</td>
<td></td>
</tr>
<tr>
<td>14A</td>
<td>FA0</td>
<td>5B0 000</td>
</tr>
<tr>
<td>1AA</td>
<td>5B0</td>
<td>000 000</td>
</tr>
</tbody>
</table>
Cuckoo Filter Issue

1) Eviction overhead in high load factors (>75%)
   • Worse in NVM with higher latency

2) Failure-atomicity issue
   • Typical setting: 4 slots, 12 bit fingerprints
   • A bucket is 6 bytes
   • NVM’s atomic write unit: 8 byte
Cuckoo Filter Issue

1) Eviction overhead in high load factors (>75%)
   • Worse in NVM with higher latency

2) Failure-atomicity issue
   • Typical setting: 4 slots, 12 bit fingerprints
   • A bucket is 6 bytes
   • NVM’s atomic write unit: 8 byte
AniFilter*

• Cuckoo Filter optimized for NVM
• Optimization techniques
  • Spillable Buckets
  • Lookahead Evictions
  • Bucket Primacy
• Failure-atomic with minimal logging

* Anis are in the cuckoo family and have communal nests.
Spillable Buckets

• Spill a fingerprint in next 2 buckets
• Only spill in the first slot

<table>
<thead>
<tr>
<th></th>
<th>A0F</th>
<th>D1F</th>
<th>E49</th>
<th>F8A</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000</td>
<td>27D</td>
<td>2A0</td>
<td>F09</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
<td>A15</td>
<td>1EF</td>
<td>AFE</td>
</tr>
<tr>
<td>A4A</td>
<td>DA1</td>
<td>EC0</td>
<td>F02</td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>5B0</td>
<td>B8A</td>
<td>CAC</td>
<td></td>
</tr>
</tbody>
</table>
Spillable Buckets

- Spill a fingerprint in next 2 buckets
- Only spill in the first slot

<table>
<thead>
<tr>
<th>Key x</th>
<th>$H_1(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0F</td>
<td>D1F</td>
</tr>
<tr>
<td>000</td>
<td>27D</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>000</td>
<td>A15</td>
</tr>
<tr>
<td>A4A</td>
<td>DA1</td>
</tr>
<tr>
<td>000</td>
<td>5B0</td>
</tr>
</tbody>
</table>
Spillable Buckets

• Spill a fingerprint in next 2 buckets
• Only spill in the first slot

<table>
<thead>
<tr>
<th>Key x</th>
<th>Spill FP_x</th>
</tr>
</thead>
<tbody>
<tr>
<td>A4A</td>
<td>DA1</td>
</tr>
<tr>
<td>A0F</td>
<td>D1F</td>
</tr>
<tr>
<td>E49</td>
<td>F8A</td>
</tr>
<tr>
<td>000</td>
<td>27D</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>000</td>
<td>A15</td>
</tr>
<tr>
<td>000</td>
<td>1EF</td>
</tr>
<tr>
<td>000</td>
<td>AFE</td>
</tr>
<tr>
<td>A4A</td>
<td>DA1</td>
</tr>
<tr>
<td>A0F</td>
<td>D1F</td>
</tr>
<tr>
<td>E49</td>
<td>F8A</td>
</tr>
<tr>
<td>000</td>
<td>27D</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>000</td>
<td>A15</td>
</tr>
<tr>
<td>000</td>
<td>1EF</td>
</tr>
<tr>
<td>000</td>
<td>AFE</td>
</tr>
</tbody>
</table>

Swapped to encode spill
Spillable Buckets – Theoretic Analysis

- Eviction probability with and without Spillable Buckets
- Probabilistic model to compute \( \text{Prob}(X=k) \)
Spillable Buckets – Theoretic Analysis

• Eviction probability with and without Spillable Buckets
• Probabilistic model to compute Prob(X=k)
Spillable Buckets – Theoretic Analysis

- Eviction probability with and without Spillable Buckets
- Probabilistic model to compute Prob(X=k)
Spillable Buckets – Theoretic Analysis

- Eviction probability with and without Spillable Buckets
- Probabilistic model to compute $\text{Prob}(X=k)$
Spillable Buckets – Theoretic Analysis

- Eviction probability with and without Spillable Buckets
- Probabilistic model to compute $\text{Prob}(X=k)$
Lookahead Eviction

- Evict a fingerprint that does not incur further eviction

<table>
<thead>
<tr>
<th></th>
<th>A0F</th>
<th>D1F</th>
<th>E49</th>
<th>F8A</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>27D</td>
<td>2A0</td>
<td>F09</td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>A15</td>
<td>1EF</td>
<td>AFE</td>
<td></td>
</tr>
<tr>
<td>A4A</td>
<td>DA1</td>
<td>EC0</td>
<td>F02</td>
<td></td>
</tr>
<tr>
<td>080</td>
<td>B8A</td>
<td>5B0</td>
<td>CAC</td>
<td></td>
</tr>
</tbody>
</table>

Occupancy flags

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Bucket Primacy

- Primary bucket (H₁) and secondary bucket (H₂)

### Swapped to encode overflow

<table>
<thead>
<tr>
<th>A0F</th>
<th>D1F</th>
<th>F8A</th>
<th>E49</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A0</td>
<td>27D</td>
<td>2A0</td>
<td>F09</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>000</td>
<td>A15</td>
<td>1EF</td>
<td>AFE</td>
</tr>
<tr>
<td>A4A</td>
<td>DA1</td>
<td>F02</td>
<td>EC0</td>
</tr>
<tr>
<td>080</td>
<td>B8A</td>
<td>5B0</td>
<td>CAC</td>
</tr>
</tbody>
</table>

- Buckets previously not overflowed
Logging for Failure Atomicity

- Type-A, -B, -C buckets
- Requires different # of loggings
- Logging example for Type-B buckets

<table>
<thead>
<tr>
<th>Type A</th>
<th>Type B</th>
<th>Type C</th>
<th>Type A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0F</td>
<td>D1F</td>
<td>F8A</td>
<td>E49</td>
</tr>
<tr>
<td>2A0</td>
<td>27D</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>AC7</td>
<td>000</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>009</td>
<td>A15</td>
<td>1EF</td>
<td>AFE</td>
</tr>
</tbody>
</table>
Logging for Failure Atomicity

- Type-A, -B, -C buckets
- Requires different # of loggings
- Logging example for Type-B buckets
- 8-byte logging record

<table>
<thead>
<tr>
<th>32bit</th>
<th>12bit</th>
<th>12bit</th>
<th>4bit</th>
<th>1bit</th>
<th>1bit</th>
<th>2bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>bucket index</td>
<td>FP(insert)</td>
<td>FP(evict)</td>
<td>unaligned FP</td>
<td>spill status</td>
<td>log-splitted bit</td>
<td>Timestamp</td>
</tr>
</tbody>
</table>
Logging for Failure Atomicity

- Type-A, -B, -C buckets
- Requires different # of loggings
- Logging example for Type-B buckets

```
008 EA1 F01 0F0
17
```

FP(insert) FA1
Logging for Failure Atomicity

- Type-A, -B, -C buckets
- Requires different # of loggings
- Logging example for Type-B buckets

```
0 0 8 E A 1
F 0 1
0 F 0
```

FP(insert)  FA 1

```
17 FP(insert) FP(evict) unaligned FP spill status log-split bit Timestamp
```

17
Logging for Failure Atomicity

- Type-A, -B, -C buckets
- Requires different # of loggings
- Logging example for Type-B buckets

```
0 0 8   E A 1   F 0 1   0 F 0
```

FP(insert) [FA1]
Logging for Failure Atomicity

- Type-A, -B, -C buckets
- Requires different # of loggings
- Logging example for Type-B buckets

<table>
<thead>
<tr>
<th>0 0 8</th>
<th>E A 1</th>
<th>F 0 1</th>
<th>0 F 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FP(insert) [FA1]

17  F A 1  E A 1  unaligned FP  spill status  log-splitted bit  Timestamp
Logging for Failure Atomicity

• Type-A, -B, -C buckets
• Requires different # of loggings
• Logging example for Type-B buckets

<table>
<thead>
<tr>
<th>17</th>
<th>0 0 8</th>
<th>E A 1</th>
<th>F 0 1</th>
<th>0 F 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>F A 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FP(insert)
Logging for Failure Atomicity

• Type-A, -B, -C buckets
• Requires different # of loggings
• Logging example for Type-B buckets

FP(insert)  FA1

17  0  0  8  EA1  F 0 1  0  F 0

17  FA1  EA1  E  0  0  0 0
Recovery

1. Check log record and read $FP_i$, $FP_e$, and bucket $B$

2. Test if $FP_i$ and $FP_e$ are in $B$
   
   a) $FP_e \in B$ and $FP_i \in B$
   
   b) $FP_e \notin B$ and $FP_i \notin B$
   
   c) $FP_e \in B$ and $FP_i \notin B$
   
   d) $FP_e \notin B$ and $FP_i \notin B$

3. For (a), (b), (c) insertion is incomplete
   
   For (d) examine meta-data in log record

$\Rightarrow$ Example recovery process for (d)
Recovery – Example

Write a log record

1

0 0 8
E A 1
F 0 1
0 F 0

17
FA 1
EA 1
E
0
0
0 0
Recovery

1. Write a log record

2. Update high 16 bit
Recovery

1. Write a log record

2. Update high 16 bit

3. Crash
Write a log record

1. Write a log record

2. Update high 16 bit

3. Crash

4. Recovery
Recovery

1. Write a log record

2. Update high 16 bit

3. Crash

4. Recovery
Recovery

1. Write a log record

2. Update high 16 bit

3. Crash

4. Recovery
Recovery

1. Write a log record
2. Update high 16 bit
3. Crash
4. Recovery

FP₁ > FP₂ ≠ spill status bit
⇒ Incomplete update detected
Recovery

1. Write a log record

2. Update high 16 bit

3. Crash

4. Recovery

FP₁ > FP₂ ≠ spill status bit

→ Incomplete update detected
Parallel Implementation

• Synchronization
• Logging
Parallel Implementation – Sync

- Shared lock for buckets
- Holding locks for both buckets
- Lock ordering – for spill, try lock

8 byte boundaries
Parallel Implementation – Sync

- Shared lock for buckets
- Holding locks for both buckets
- Lock ordering – for spill, try lock
Parallel Implementation – Sync

- Shared lock for buckets
- Holding locks for both buckets
- Lock ordering – for spill, try lock

➔ No deadlock, yet may have livelock

1) Single insertion and multiple lookups
   - Bounded wait time for lookups
2) Multiple insertions
   - Extremely unlikely (\(< 10^{-28}\))
Parallel Implementation – Logging

• Thread-local log entries per thread
• Additional log write for commit mark
  – To determine write order between 2 threads accessing a same bucket
**Evaluation**

- System setting

<table>
<thead>
<tr>
<th></th>
<th>Intel Optane DC Persistent Memory*</th>
<th>Quartz Emulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Xeon Gold 5215M 2.5GHz (10 cores)</td>
<td>Xeon E5-2620 2.4GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>384 GB DRAM + 1512 GB NVM</td>
<td>96 GB DRAM</td>
</tr>
<tr>
<td>OS</td>
<td>Linux Kernel 4.18.0</td>
<td>Linux Kernel 4.8.12</td>
</tr>
</tbody>
</table>

*Used AppDirect mode
Evaluation

• Evaluated Filters

<table>
<thead>
<tr>
<th>Filter</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cuckoo Filter</td>
<td>CF</td>
<td>Bucketized Cuckoo Filter</td>
</tr>
<tr>
<td>Morton Filter</td>
<td>MF</td>
<td>DRAM-optimized Cuckoo Filter</td>
</tr>
<tr>
<td>Rank-and-Select Quotient Filter</td>
<td>RSQF</td>
<td>SSD-optimized Quotient Filter</td>
</tr>
<tr>
<td>Bloom Filter</td>
<td>BF</td>
<td>Bitmap-based AMQ</td>
</tr>
</tbody>
</table>

→ Configured to have the same false-positive rates
Parallel Throughput*

AniFilter upto 10.7x faster (2.6x faster on avg) for insertion

*Intel Optane DC PM, 10 threads
Sequential Throughput*

*Intel Optane DC PM
Sequential Throughput*

*Quartz emulation, 75% load factor
Effect of Lookahead Eviction – Occupancy Flags

![Graph showing the effect of lookahead eviction on occupancy flags]

- AF
- CF
- AF (O.F. 1/2)
- AF (No O.F.)
Synergy between Optimizations

- Spillable Buckets and Lookahead Eviction’s Impact on Bucket Primacy
Conclusion

• AniFilter – Optimized Cuckoo Filter for NVM

• Optimizations
  • Spillable Buckets
  • Lookahead Evictions
  • Bucket Primacy

• Logging for Failure-Atomicity

• Evaluation on NVM
Q/A

seojiwon@gmail.com