Erasing Belady’s Limitations: In Search of Flash Cache Offline Optimality

Yue Cheng¹,², Fred Douglis², Philip Shilane², Michael Trachtman², Grant Wallace², Peter Desnoyers³, Kai Li⁴

¹Virginia Tech, ²EMC Corporation, ³Northeastern University, ⁴Princeton University

Motivation

• Using SSDs for an HDD cache
  • SSD cache w/ DRAM & HDD
  • Goal: balance performance against endurance
• Nitro [ATC’14], CacheDedup [FAST’16]
• RIPQ [FAST’15]
• Pannier [Middleware’15]

• Are we doing well?
  • Comparison against an offline “best case”
  • But what is the offline optimal for flash cache?

Background: Offline Optimality

• Belady’s MIN: A simple offline caching alg. when the next access is known
  • Evicts the item that will be used furthest in the future
  • Yields the optimal read hit ratio (RHR)
  • Ignorant of minimizing erasures/block/day (EPBPD)
• MIN is not able to provide the optimal erasures in the context of flash caching
• MIN inserts items that won’t actually be read

Offline Flash Caching

• Objectives
  • Minimize erasures s.t. maximal RHR
  • Never insert items if it does not increase RHR
  • Other objectives possible as discussed in the paper
• True optimal vs. Heuristic
• Complexity of true optimal?
• Approximation is the focus of this work

Container-optimized Heuristic

1. On a read miss, check if the block will be read before evicted
2. Pack a block into in-RAM write buffer and insert it to MIN PQ
3. Disperse the write buffer into containers and write them to the flash cache
4. On a read hit, read the block from the flash cache
5. At garbage collection, copy-forward (CF) the valid block to the write buffer by checking if it will be reread before evicted

Comparing Algorithms

<table>
<thead>
<tr>
<th>Policy</th>
<th>Description</th>
<th>O</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>Least recently used</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>RIPQ+</td>
<td>Static web content</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Pannier</td>
<td>Handles divergent containers</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>MIN</td>
<td>Don’t insert data w/ furthest next ref</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MIN+</td>
<td>Don’t insert data evicted w/o read</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>C</td>
<td>Our container-optimized heuristic</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Policy # of Blocks % of Cache Size

<table>
<thead>
<tr>
<th>Policy</th>
<th>50% Cache Size</th>
<th>25% Cache Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>121, 119</td>
<td>43, 42</td>
</tr>
<tr>
<td>RIPQ+</td>
<td>121, 119</td>
<td>43, 42</td>
</tr>
<tr>
<td>Pannier</td>
<td>121, 119</td>
<td>43, 42</td>
</tr>
<tr>
<td>MIN</td>
<td>121, 119</td>
<td>43, 42</td>
</tr>
<tr>
<td>MIN+</td>
<td>121, 119</td>
<td>43, 42</td>
</tr>
<tr>
<td>C</td>
<td>121, 119</td>
<td>43, 42</td>
</tr>
</tbody>
</table>

Evaluating Optimization Techniques

• R₁: Omit insertions w/o reread
• TRIM: Notify GC to omit dead blocks
• CFR: Avoid wasted CF blocks
• E: Segregate blocks by evict timestamp

Conclusion

• Important to have a baseline for the offline optimal considering both RHR and endurance
• Additional optimizations may be possible to move our heuristic to the true optimal

 Instantiate Heuristic

EPBPD w/ identical optimizations