Viyojit: Decoupling Battery and DRAM Capacities for Battery-Backed DRAM

Rajat Kateja

Anirudh Badam*, Sriram Govindan*, Bikash Sharma+, Greg Ganger#
#CMU, *Microsoft, +Facebook (work done while at Microsoft)
Overview

• Problem
  • Excellent performance with BB-DRAM
  • But, need large batteries for high capacity
• Observation
  • Typical battery used << Worst case
• Idea: Provision battery for typical requirement
• Challenge: Durability guarantee
• Solution: Viyojit, bound dirty data in BB-DRAM
• 89% smaller battery, only 7-25% throughput loss
High performance storage with BB-DRAM

Widely used high performance NVM technology
Poor battery scaling limits BB-DRAM capacity

Challenging to provision high capacity BB-DRAM

Projected growth:
- DRAM: ~100,000x
- Lithium: <4x
Typical battery required is much smaller

- Need to flush only dirty data
- Typical workloads dirty only a small fraction of dataset

Microsoft production MapReduce like workload

Can provision small batteries
Viyojit ensures durability with small batteries

- 10 Clean BB-DRAM pages
- Battery for 2 Dirty pages
- Write protect all clean pages

(a) Write page 0

(b) Write page 2

(c) Write page 9

- Make page 0 writable
- Make page 2 writable
- Make page 0 clean
  - Write protect
- Make page 9 writable

Write protect pages to track and bound dirty data
Proactively write-back infrequently updated data
Viyojit writes-back infrequently updated pages

- Target Least Recently Updated (LRU) page
  - Motivated by Least Recently Used policy
- Consider only write accesses
  - Page always stays in memory
- Identify target page using dirty bit
  - Periodically read and reset dirty bit

Write-back Least Recently Updated pages
Viyojit avoids blocking applications on write-backs

- Background write-back thread
- Adapt aggressiveness based on “dirty pressure”
  - Expected number of new dirty pages
- Count new dirty pages in each period
- Predict new dirty pages in next period
  - Exponentially decaying average

Leave slack to absorb dirty pages w/o demand SSD write
Evaluation

<table>
<thead>
<tr>
<th>In-memory key-value store (Redis)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software persistent memory library (Intel’s PMDK)</td>
</tr>
</tbody>
</table>

**Yahoo! Cloud Serving Benchmarks (YCSB): NoSQL workload**
- 10 million ops, 16 Threads

**BB-DRAM** (battery capacity ≡ database size) or **Viyojit** (battery capacity ≡ fraction of the database size)

- 20 cores; 140 GB DRAM; 280 GB SSD

**Metrics:**
- Throughput
- Latency (average and tail)
- SSD write traffic
Effect of smaller battery on application throughput

Lower is better

Throughput loss relative to 100% battery (%)

battery capacity

- Red: 11%
- Blue: 23%
- Green: 46%

Low overhead (7-25%) with 89% smaller battery
Effect of smaller battery on average and tail latency

YCSB-A (50% writes)

Write traps to track updates lead to high tail latency
Takeaways from Viyojit’s results

- **Key takeaway:** Big reduction in battery; low overheads
  - 89% smaller battery, only 7-25% throughput loss

- **Other results:**
  - Overheads decrease as battery capacity increases
  - Overheads decrease as write skew increases
  - Overheads decrease as BB-DRAM size increases
Conclusion

• Excellent performance with BB-DRAM
  • But, battery scaling is problematic

• Typically, small battery can suffice
  • If an upper bound could be ensured

• Viyojit bounds number of dirty BB-DRAM pages

• 89% smaller battery, only 7-25% throughput loss