Boosting Random Write Performance for Enterprise Flash Storage Systems

Tao Xie and Janak Koshia

Computer Science Department
San Diego State University
Agenda

- Introduction
- Related Work and Motivation
- EPO (Element-level Parallel Optimization)
- Performance Evaluation
- Conclusion
Samsung’s K9XXG08UXM series NAND-Flash
Flash Translation Layer

- Mapping between Logical address space to Physical address space

- Wear-leveling

- Garbage collecting
Why Random Write Is an Issue?

- Out-of-place update
- Time consuming erase and garbage collection
- Unit difference between RW and erase operation
- Complicated FTL (Flash Translation Layer) logic
Random Write Performance of SSD

- Depends mainly on two factors
  - Architecture of an SSD
  - Type of workload
Related Work

- Add non-volatile RAM (NVRAM)

- DFTL (Demand based Flash Translation Layer) – FLT has to be changed

- BPLRU (Block Padding Least Recently Used) – Amplifies read and write requests
Motivation

- Exploiting the architecture of an SSD
  - Element-level Concurrency
  - Die-level parallelism
  - Plane-level interleaving
Architectural of an SSD
Important assumptions

- Workload contains only write requests
- Each request is one page size
- LBA of each request is page aligned
The EPO Strategy

E.g. 1, 2, 3, 2, 4, 3
Experimental Setup

- Simulator DiskSim 4.0 and SSD add-on by Microsoft
  - Command line tool
    - disksim <parfile> <outfile> <tracetype> <tracefile> <synthgen> [ par override [ ...]

- Traces
  - Real-world traces (OLTP)
    - TPC-C
    - Financial1
    - Financial2

- Comparing results with other three schemes
  - No cache
  - LRU (Least Recently Used)
  - BPLRU (Block Padding Least Recently Used)
Parameters Tested

- Configuration
  - Varying cache size
  - Varying page size
  - Varying number of elements
# Real World Trace Statistics

<table>
<thead>
<tr>
<th>Workloads</th>
<th>Financial1</th>
<th>Financial2</th>
<th>TPC-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of writes</td>
<td>2,000,000</td>
<td>650,000</td>
<td>2,000,000</td>
</tr>
<tr>
<td>Mean write size (KB)</td>
<td>3.9</td>
<td>2.9</td>
<td>10.2</td>
</tr>
<tr>
<td>Write per second</td>
<td>62.76</td>
<td>10.52</td>
<td>4337.83</td>
</tr>
<tr>
<td>Write size range (KB)</td>
<td>0.5-3148.5</td>
<td>0.5-256.5</td>
<td>0.5-1024</td>
</tr>
</tbody>
</table>
## Simulation Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value (Default) — (Varied)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write buffer capacity (MB)</td>
<td>(8) — (4, 8, 16, 32)</td>
</tr>
<tr>
<td>Number of elements</td>
<td>(48) — (16, 32, 48, 64)</td>
</tr>
<tr>
<td>Number of planes in an element</td>
<td>(8)</td>
</tr>
<tr>
<td>Page size (KB)</td>
<td>(4) — (1, 2, 4)</td>
</tr>
<tr>
<td>Flash block size (page)</td>
<td>(64)</td>
</tr>
<tr>
<td>Element capacity (GB)</td>
<td>(4)</td>
</tr>
<tr>
<td>Flash SSD capacity (GB)</td>
<td>(192) — (64, 128, 192, 256)</td>
</tr>
<tr>
<td>Block erase latency (µs)</td>
<td>(1500)</td>
</tr>
<tr>
<td>Page read latency (µs)</td>
<td>(25)</td>
</tr>
<tr>
<td>Page write latency (µs)</td>
<td>(200)</td>
</tr>
<tr>
<td>Chip transfer latency per byte (µs)</td>
<td>(0.025)</td>
</tr>
</tbody>
</table>
Impacts of Cache Size

TPC-C
Continue…

Financial1

![Graph 1: Mean response time vs Cache size (MB)]

![Graph 2: Throughput (MB/S) vs Cache size (MB)]
Impacts of Page Size

TPC-C

![Graph showing the impacts of page size on TPC-C](image)

![Graph showing the throughput of TPC-C](image)
Continue...

**Financial1**

![Graph 1: Mean response time vs. Page size (KB)]

- **NoCache**
- **LRU**
- **BPLRU**
- **EPO**

![Graph 2: Throughput (MB/S) vs. Page size (KB)]

- **NoCache**
- **LRU**
- **BPLRU**
- **EPO**
Scalability

TPC-C

![Graph showing TPC-C results with different cache strategies: NoCache, LRU, BPLRU, and EPO. The graphs display mean response time and throughput for varying numbers of elements.]
Continue...

- Financial1

![Graph showing Mean response time (ms) and Throughput (MB/S) for NoCache, LRU, BPLRU, and EPO in Financial1 datasets.](image)
Conclusion

- Write buffer is placed after FTL
- Easy to implement the algorithm
- Low cost hardware – small size DRAM
- Trivial change in FTL
- Gives good performance towards random write workload
Acknowledgements

- We thank DiskSim developers who provided an efficient, accurate, highly-configurable disk system simulator for storage research community.
- We would like to thank the researchers from Microsoft Research who developed the SSD model for DiskSim 4.0.
- This work was supported by the US National Science Foundation under grants CNS (CAREER)-0845105 and CNS-0834466.
Thank you
Questions?
Backup slides
Varying cache size

- Synthetic benchmark 1

![Graphs showing mean response time and throughput vs cache size for different cache algorithms.](image)
Continue...

- Synthetic benchmark2

Graphs showing performance metrics (mean response time and throughput) vs. cache size for different cache management policies: Nocahce, LRU, BPLRU, and EPO.
Varying page size

- Synthetic benchmark1

Graphs showing mean response time and throughput for different page sizes and cache algorithms.
Continue...

- Synthetic benchmark2

![Graphs showing mean response time and throughput for different page sizes for Nocahce, LRU, BPLRU, and EPO across two axes.](image-url)
Varying number of elements

- Synthetic benchmark 1

![Graph showing mean response time and throughput](image_url)
Continue...

- Synthetic benchmark2

![Graph showing mean response time and throughput for different benchmarks.

- X-axis: Number of elements
- Y-axis: Mean response time (ms) for left graph, Throughput (MB/s) for right graph
- Legends: Nocahce, LRU, BPLRU, EPO

- The graphs illustrate performance improvements with increasing number of elements.