An Efficient SSD-based Hybrid Storage Architecture for Large-scale Search Engines

Ruixuan Li, Chengzhou Li, Weijun Xiao, Hai Jin, Heng He, Xiwu Gu, Kunmei Wen and Zhiyong Xu

School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, China
Introduction

- Modern information retrieval
  - **Characteristics**: mass data and large query requests
  - **Solutions**: caching and distributed file system
Introduction

Solid State Disk (SSD) VS. Hard Disk Drive (HDD)

The desired technical merits:
- Low power consumption
- Shock resistance
- Compact size
- High random I/O performance

<table>
<thead>
<tr>
<th>Type</th>
<th>Capacity (GB)</th>
<th>Read (MB/S)</th>
<th>Write (MB/S)</th>
<th>Unit Price (RMB/GB)</th>
<th>P. C. (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDD</td>
<td>500</td>
<td>100</td>
<td>80</td>
<td>0.56</td>
<td>17~20</td>
</tr>
<tr>
<td>SSD</td>
<td>200</td>
<td>285</td>
<td>275</td>
<td>12.5</td>
<td>2</td>
</tr>
<tr>
<td>RAM</td>
<td>5</td>
<td>6700</td>
<td>5700</td>
<td>92.5</td>
<td>15</td>
</tr>
</tbody>
</table>

Ruixuan Li, Chengzhou Li, Weijun Xiao, et al.
School of Computer Science and Technology
Huazhong University of Science and Technology
However, some potential issues may complicate the full adoption of SSD.

The potential technical deficiencies: erase-before-write, limited life span, asymmetrical I/O performance, high cost.
Observations

- It is not suitable to store all the data on SSD
- It is necessary to design SSD adaptive applications
- Performance, cost and reliability should synthetically be taken into consideration in large-scale search engines

SSD-based Hybrid Storage Architecture for Large-scale Search Engines
SSD-based Hybrid Storage Architecture

- The SSD-based hybrid storage architecture for large-scale search engines, with memory as the first-level cache and SSD as the second-level cache

Three main issues: data selection, data placement, data replacement.
Problem Description and Consideration

- Problem description (a data management problem for a special two-level cache)
  - L1 cache: memory
  - L2 cache: SSD (special I/O characteristics)

- Problem consideration
  - It is reasonable to place read intensive data in memory or SSD and write-intensive data on HDD
  - An advanced algorithm is needed to identify which part of the inverted list is worth caching
  - It is a challenge to determine whether the data evicted from memory should be flushed to SSD
Motivation

☐ Improve I/O performance
  - Decrease the access on HDD
  - Ensure high I/O performance on SSD

☐ Reduce costs
  - Smaller memory with larger SSD (use ordinary servers, rather than high performance servers)

☐ Improve reliability
  - More reliable than the scenario replacing HDD with SSD completely
  - Extend SSD Lifetimes
Related Work

- Related research work
  - Caching in search engines
  - SSD-based buffer management
  - SSD-based hybrid storage architecture

- The most valuable research to our work
  - CFLRU: Clean First LRU (CASE’2006)
  - FlashLite (ICDCS’2009)
  - SSD buffer pool extensions for DB (PVLDB’2010)
  - Allocating inverted index into flash memory for search engines (WWW’2011, the first research work on SSD and IR)

To the best of our knowledge, we are the first to analyze this problem and propose an efficient design for the application of search engines.
Contributions

- We analyze the I/O patterns of search engines and carefully choose the data to be cached in memory or SSD, which is a data selection policy.
- We propose an improved log-based cache data management policy, which is a data placement policy.
- We propose appropriate data replacement policies for SSD, which is a data replacement policy.
Outline

- The I/O patterns of search engines
- Log-based data management policies
  - Data selection
  - Data placement
  - Data replacement
- Experiments and evaluation
  - Hit ratio comparison
  - Retrieval performance comparison
  - Retrieval cost comparison
  - SSD validation
- Conclusion and future work
The I/O patterns of search engines

- Four characteristics in I/O patterns: dominant read, locality, skipped read, and random read

Ruixuan Li, Chengzhou Li, Weijun Xiao, et al.
School of Computer Science and Technology
Huazhong University of Science and Technology
Log-based data management

- We divide the data on SSD into three states logically, respectively normal state, replaceable state and free state.

Three main operations: Read, Log Write, Overwrite
Outline

- The I/O patterns of search engines
- Log-based data management policies
  - Data selection
  - Data placement
  - Data replacement
- Experiments and evaluation
  - Hit ratio comparison
  - Retrieval performance comparison
  - Retrieval cost comparison
  - SSD validation
- Conclusion and future work
Data management: data selection

- How to select the data to be cached in memory or SSD?

The term access frequency and inverted size distribution

**EV**: efficiency value, select data according to EV

\[ EV = \frac{Freq}{Sc} \]

Ruixuan Li, Chengzhou Li, Weijun Xiao, et al.
School of Computer Science and Technology
Huazhong University of Science and Technology
Data management: data placement

- How to place the cached data on SSD?

Data placement policy:
- Partially store
- Logic block based placement
- Assemble and clip
Data management: data replacement (1)

How to replace the cached data on SSD?

(a) Random Write

(b) Sequential Write

result cache replacement
Data management: data replacement (2)

- How to replace the cached data on SSD?

![Diagram](image)

**Inverted List Cache Replacement**

**Working Region**

**Replace First Region, W=5**

**Normal State**

**Replaceable State**
Outline

- The I/O patterns of search engines
- Log-based data management policies
  - Data selection
  - Data placement
  - Data replacement
- Experiments and evaluation
  - Hit ratio comparison
  - Retrieval performance comparison
  - Retrieval cost comparison
  - SSD validation
- Conclusion and future work
Experiments and evaluation

<table>
<thead>
<tr>
<th>Test-platform Environment</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IR Tool</td>
<td>Lucene 3.0.0</td>
</tr>
<tr>
<td>Data Set</td>
<td>enwiki-20090805-pages-articles.xml</td>
</tr>
<tr>
<td>Query Log</td>
<td>AOL-user-ct-collection</td>
</tr>
<tr>
<td>SSD</td>
<td>Intel SSD 320 Series 40GB</td>
</tr>
<tr>
<td>HDD</td>
<td>WDCWD3200AAJS 180GB</td>
</tr>
<tr>
<td>OS</td>
<td>Windows Server 2003/Ubuntu 10.04</td>
</tr>
<tr>
<td>CPU/RAM</td>
<td>Inter(R) Pentium(R) Dual CPU E2180/2GB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Simulated SSD</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD Simulator</td>
<td>FlashSim (PSU)/DiskSim 3.0</td>
</tr>
<tr>
<td>I/O Trace Analyzer</td>
<td>DiskMon 2.0.1</td>
</tr>
<tr>
<td>FTL</td>
<td>page-mapping</td>
</tr>
<tr>
<td>Page Size</td>
<td>2KB</td>
</tr>
<tr>
<td>Block Size</td>
<td>128KB</td>
</tr>
<tr>
<td>Page Read</td>
<td>32.725μs</td>
</tr>
<tr>
<td>Page Write</td>
<td>101.475μs</td>
</tr>
<tr>
<td>Block Erase</td>
<td>1.5ms</td>
</tr>
</tbody>
</table>

Test-platform
- Hit ratio
- Performance
- Cost

Simulation
- SSD validation
Experiments and evaluation

- Hit ratio comparison

(a) RC, IC and RIC Comparison

(b) LRU, CBLRU and CBSLRU Comparison
Experiments and evaluation

- Retrieval performance comparison

(a) Index Stored on HDD/SSD with 1L Cache

(b) Response Time Comparison

(c) Throughput Comparison
Experiments and evaluation

- Retrieval cost comparison

(a) 1L Cache and 2L Cache Comparison

(b) Comparison in Different Situations

Ruixuan Li, Chengzhou Li, Weijun Xiao, et al.
School of Computer Science and Technology
Huazhong University of Science and Technology
Experiments and evaluation

- SSD validation

(a) Block Erasure Count

(b) Flash Average Access Time

Ruixuan Li, Chengzhou Li, Weijun Xiao, et al.
School of Computer Science and Technology
Huazhong University of Science and Technology
Outline

- The I/O patterns of search engines
- Log-based data management policies
  - Data selection
  - Data placement
  - Data replacement
- Experiments and evaluation
  - Hit ratio comparison
  - Retrieval performance comparison
  - Retrieval cost comparison
  - SSD validation
- Conclusion and future work
Conclusion and future work

- The experimental results demonstrate our design
  - Improve the hit ratio by 13.31%,
  - Improve the performance by 41.05%
  - Reduce the average access time in SSD by 43.83%
  - Reduce block erase operations by 71.52%.

- The future work
  - Consider the dynamic scenario (real-time)
  - Consider the three-level cache scheme, namely results, inverted lists and intersections
  - Consider the data management policy in DFS
Thanks for your attention

Contact information:

- Ruixuan Li
- Huazhong University of Science and Technology
- rxli@hust.edu.cn
- http://idc.hust.edu.cn/~rxli/