The Impact of Solid State Drive on Search Engine Cache Management

Jiancong Tong
Ph.D. candidate at Nankai University
Visiting student at University of Melbourne
lingfenghx@gmail.com

with
Gang Wang, Xiaoguang Liu (Nankai University)
and
Jianguo Wang, Eric Lo, Man Lung Yiu (Hong Kong Polytechnic University)

Monash University
May 12, 2014
1. Background and Motivation

2. Research Questions and Answers
   - RQ1: What is the impact of SSD on buffer management?
   - RQ2: How could we deal with that?
Background and Motivation

Hard Disk Drive

How does HDD work? [Garcia et al., 2000]

Random read latency of HDD

- Seek time
- Rotational latency
- Transfer time

Caching technology is used to reduce the latency.
What is a Cache?

Small, fast memory used to improve average access time to large, slow storage media.

Exploits locality: both spacial and temporal.

Almost everything is a cache in computer architecture...
What is a Cache? (Cont.)

- **Cache Hit**: the requested data is found in the memory
- **Cache Miss**: the requested data is not found in the memory

\[
\text{Hit ratio} = \frac{\#\text{Hits}}{\#\text{Memory accesses}} \quad \text{Miss ratio} = \frac{\#\text{Misses}}{\#\text{Memory accesses}}
\]

\[\text{Hit ratio} + \text{Miss ratio} = 1\]
What is Solid State Drive?

- SSD: **New Faster** \((10 \sim 100x)\) HDD with compatible interface

- Strong technical merits: [Chen et al., 2009]
  1. Lower power consumption
  2. More compact size
  3. Better shock resistance
  4. Extraordinarily faster random data access
The Rise of SSD

Words of Pioneer (by Jim Gray, 2006)
Tape is dead; Disk is tape; Flash is disk; RAM locality is King.

SSD in Large-Scale System Architectures
- **Google** 2008 (or later)
- **Baidu** 2008
- Facebook 2010
- Myspace 2010
- Oracle 2011
- Microsoft Azure 2012

Trends
Flash memory based SSD is replacing and is going to completely replace HDD as the major storage medium!
Existing caching policies were originally designed for HDD

- HDD: Very slow random read (compare to sequential read)
- Cache design principle: minimize random read

However...

```
\[ \text{4KB Sequential Read} \quad \text{4KB Random Read} \]
```

[11.5x]
```
\[ \begin{array}{c}
\text{SSD} \\
\hline
\text{HDD} \\
\end{array} \]
```

```
\[ \text{Latency of per block (4KB) read (ms)} \]
```

\[ \text{165x} \]

What now?

[Tong et al., 2013]
## Research Questions

### RQ1: What is the impact of SSD on buffer management?
- Are the existing cache techniques designed for HDD-based search engine still good for SSD-based search engine?
- What measure(s) should be used to define a ‘good’ cache policy in this case?
- If the performance of caching is improved or degraded, what does that mean to the entire system?

### RQ2: How could we deal with that?
- What to do if the efficiency of the entire system is affected by such impact?
- Can we propose better cache policies for SSD-based systems?
1. Background and Motivation

2. Research Questions and Answers
   - RQ1: What is the impact of SSD on buffer management?
   - RQ2: How could we deal with that?
Large-scale experimental study

RQ1 can be answered by evaluating the effectiveness of existing caching policies on an SSD-based search engine.

**Settings**

- **Datasets**
  - Web documents: 12,000,000 (∼100GB)
  - Queries: 1,000,000

- **Device**
  - SSD: ADATA 256GB SSD
  - HDD: Seagate 3TB 7200rpm

- **System**: Apache Lucene

- **Measure**: Query time (NOT hit ratio)
Overview of search engine architecture

[Wang et al., 2013]
Overview of search engine architecture

[Wang et al., 2013]
Index Server and List (Index) Cache

Posting Lists (Inverted Lists)

- Clayton (2, 4, 5, 10, 100, 107, ..., 1,000,000)
- Monash (1, 2, 3, 23, ...)

Index Server

- Index Cache (in memory)
- Inverted Index (on disk)
Which lists should be kept in the cache?

- High Frequency First?
- High Frequency/Size First?

Metrics to consider
Frequency/Size is a better metric (for HDD).

The reason

On HDD

- Random read is $130 \sim 170\times$ slower than sequential read
  $\implies$ List access time is (almost) a constant!

- It is \#access that dominates, not |access data|
  $\implies$ Short lists are favored (more lists can be held)

- High benefit gained when size is taken into account
Differences

On SSD

- Random read is $130 \sim 170\times$ slower than sequential read
  \[\implies\text{List access time is (almost) a constant!}\]
- Random read is only $2 \sim 10\times$ slower than sequential read
  \[\implies\text{List access time varies a lot!}\]

- Long lists have slower access time
  \[\implies\text{Admitting too many short lists offers less benefit}\]
How does SSD change the story? (Cont.)

Read access latency of posting lists of varying lengths

Conjecture 1
The best cache policy may have changed
Conjecture 1
The best cache policy may have changed – Confirmed
★ Do not rely on your knowledge acquired in HDD era.
Is Hit Ratio Reliable?

Conjecture 2

Cache hit ratio might not be reliable.

Previously on HDD | Now on SSD
---|---
Reliable | ?
As list access is a constant | NO
So, can measure the effectiveness of the cache | ?

Reliable means: high hit ratio low query time

query time

hit ratio
Is Hit Ratio Reliable? (Cont.)

Static List Cache Policy (on SSD)

Conjecture 2

Cache hit ratio might not be reliable – **Confirmed**

★ Use query time, not hit ratio
Document Server and Its Cache

Query + docID → Snippet Cache

Snippet Cache

(docID → Document Cache)

Document Cache

(document → Documents)

Documents (on disk)

Snippet Cache (in memory)
Document Server and Its Cache (Cont.)

2. Snippet Generation (CPU)

1. Document Retrieval (I/O)
   - (avg. size: 8KB)
   - block1
   - block2
   - Rand. Read
   - Seq. Read

Title: Monash University Clayton campus
URL: www.monash.edu.au/campuses/clayton/
Short Description:
Nov 5, 2012 - Monash University Clayton campus is the largest of the university's eight campuses. Eight out of the university's 10 faculties are represented on...
Evaluation on Document Server

Document retrieval is no longer the bottleneck on document server

Conjecture 3

Is it the same case for the whole system?
The Impact on the Entire System Efficiency

Time break down for query processing (on SSD)

All caches are enabled

- Posting list intersection (54.1%)
- Ranking (22.4%)
- Document access (3.2%)
- Snippet generation (10.5%)
- Others (0.1%)
- Posting list access (9.7%)

Conjecture 3

Is it the same case for the whole system? – **Confirmed**

★ Disk accessing is no longer the bottleneck!
1 Background and Motivation

2 Research Questions and Answers

- RQ1: What is the impact of SSD on buffer management?
- RQ2: How could we deal with that?
The time spent on intersection and ranking should be reduced.

**Solution**

Adopting full-term-ranking-cache (FTRC) [Altingövde et al., 2011] and two-term-intersection-cache (TTIC) [Long and Suel, 2005]
Incorporating PLC+FTRC+TTIC

(a) query latency

(b) cache benefit

FTRC : TTIC : PLC = 20% : 40% : 40%
Can we convert the latencies of random seek and sequential read into a uniform measure?
Latency-Aware Caching (Static): BLOCK [Tong et al., 2013]

For term \( t \), \( B(t) \) represents the I/O cost that can be saved during the whole query evaluating process if \( \ell(t) \) is kept in the cache.

Estimating model and existing static caching policies:

\[
B(t) = C(t) \times \frac{f_q(t)}{\ell(t)}
\]

\[
B(t) = \begin{cases} 
  f_q(t) / \ell(t), & C(t) = 1 \quad \Leftrightarrow \quad \text{QTFDF} \\
  f_q(t), & C(t) = |\ell(t)| \quad \Leftrightarrow \quad \text{QTF} \\
  f_q(t) / \ell(t) \times T(t), & C(t) = T(t) \quad \Leftrightarrow \quad \text{MECH}
\end{cases}
\]

New method: BLOCK

1 access = 1 random seek + several sequential reads

= 1 random seek + a few equivalent random seeks
Experimental Results

- BLOCK outperforms other methods on SSD
  1. BLOCK beats QTFDF (by far the best policy on HDD) by 20%-60%
  2. BLOCK outperforms QTF/MECH (identical on SSD) too

- BLOCK is better even on HDD (though not that significantly)


Thanks for your time!

Questions & Comments?

For more details of RQ1, please refer to:

- J. Wang, E. Lo, M. Yiu, J. Tong, G. Wang, X. Liu, 
  *The impact of Solid State Disk on Search Engine Cache Management*,
  In *SIGIR*, 2013, pp. 693-702.

For more details of RQ2, please refer to:

- J. Tong, G. Wang, X. Liu, 
  *Latency-Aware Strategy for Static List Caching in Flash-based Web Search Engines*,
  In *CIKM*, 2013, pp. 1209-1212.

* Some slides were taken from a presentation (“The Impact of Solid State Drive on Search Engine Cache Management”) of Jianguo Wang.