Efficient Online Index Maintenance for SSD-based Information Retrieval Systems

Ruixuan Li, Xuefan Chen, Chengzhou Li, Xiwu Gu, Kunmei Wen

Huazhong University of Science and Technology
Wuhan, China

SSD is emerging …

- Comparing to CPU and memory, speed of hard disks improve slowly and many applications become disk-bound
- SSD is emerging and widely used in many applications
  - Extraordinary high I/O performance, at least 10 times faster than hard disks
  - Growing size, already receives 1TB
  - Other merits such as light-weight, low power consumption, no noise, etc

IR systems start running on SSD

- IR systems start running on SSD now. However, most of the existing indexing approaches are based on hard disks
  - Keep each term's postings sequential in different ways
  - These optimization may not be effective on SSD
- The data access in SSD is far more different to conventional hard disks
  - erase-before-write
  - limited number of erases

Related Work

- Flash Memory Based SSD
  - Use flash memories as storage device
  - Deploy FTL to mange the data accesses inside to emulate SSDs as hard disks
  - Random writes problem, write-amplification effect
- SSD Applications
  - Conventional applications on new devices: DBMS [Lee and Moon, 2007], P2P [Kim and Ramachandran, 2009]

Related Work

- Online index maintenance: inverted index
- In-place update [Tomasic et al., 1994]
- Merge-based Update
  - Remerge [Lester et al., 2004] & No merge
  - Multiple partition strategies [N. Lester et al, 2008]

Contributions

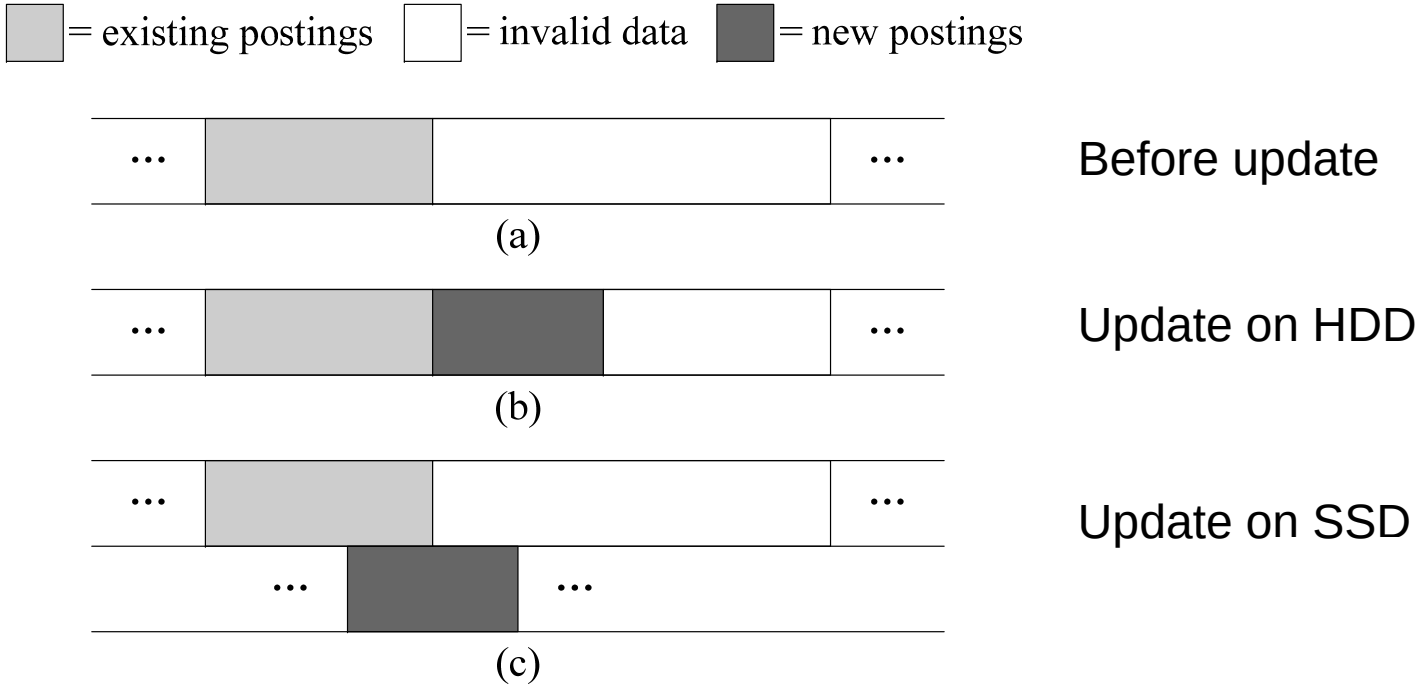
- Analyze existing indexing approaches on SSD
- Propose Hybrid Merge, a new SSD-based online index maintenance method
- Evaluate the proposed strategy and the results show that the Hybrid Merge approach is more suitable to SSD

Outline

- Introduction
- Analysis of existing methods
- Hybrid Merge
- Evaluation
- Conclusion

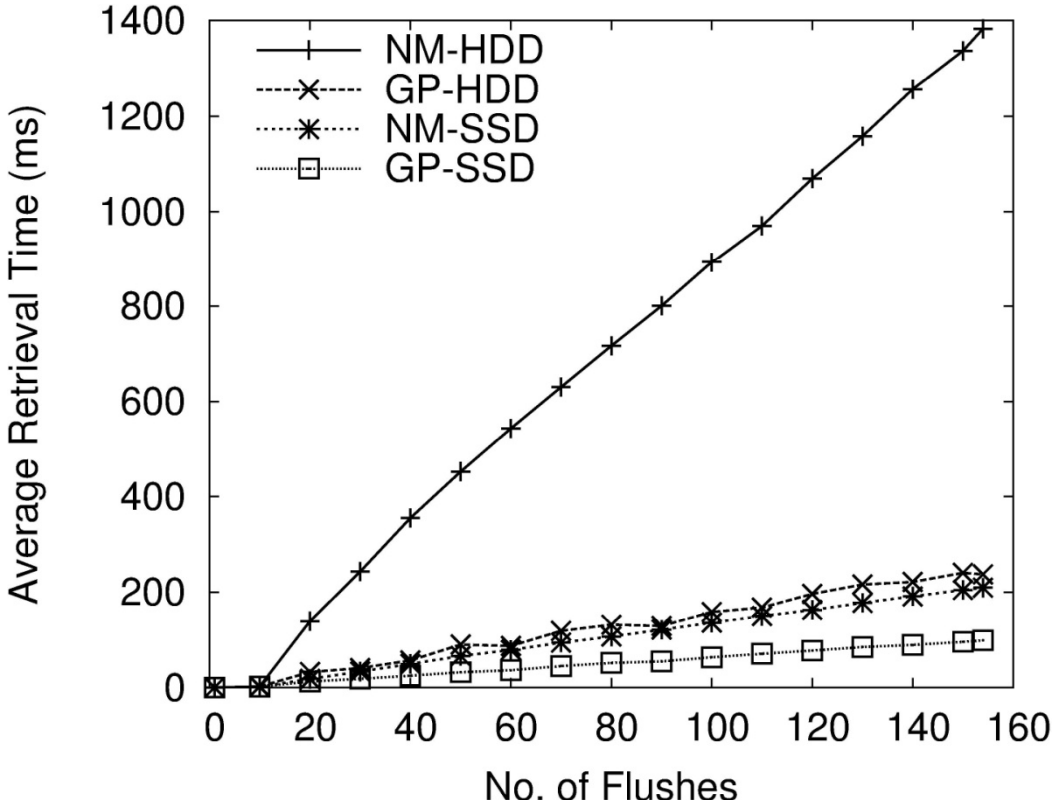
Analysis of existing methods

- In-place Update
  - In-place update leads to random write requests, which is harmful to SSDs and future lower the indexing speed
  - Out-of-place write perform by FTL makes In-place update almost impossible to achieve on SSD



Analysis of existing methods

- Merge-based Update
  - Massive extra writes (nearly 5 times more than needed) to keep postings sequential, which can potentially shorten SSD's lifespan
  - These effort is much less important on SSD than on hard disks



Outline

- Introduction
- Analysis of existing methods
- Hybrid Merge
- Evaluation
- Conclusion

Design objectives

- Avoid random disk access
- Decrease total amount of data to be written on SSD under the constraint above
- Maintain balance between index maintenance and query performance under the above constraint

Basic ideas

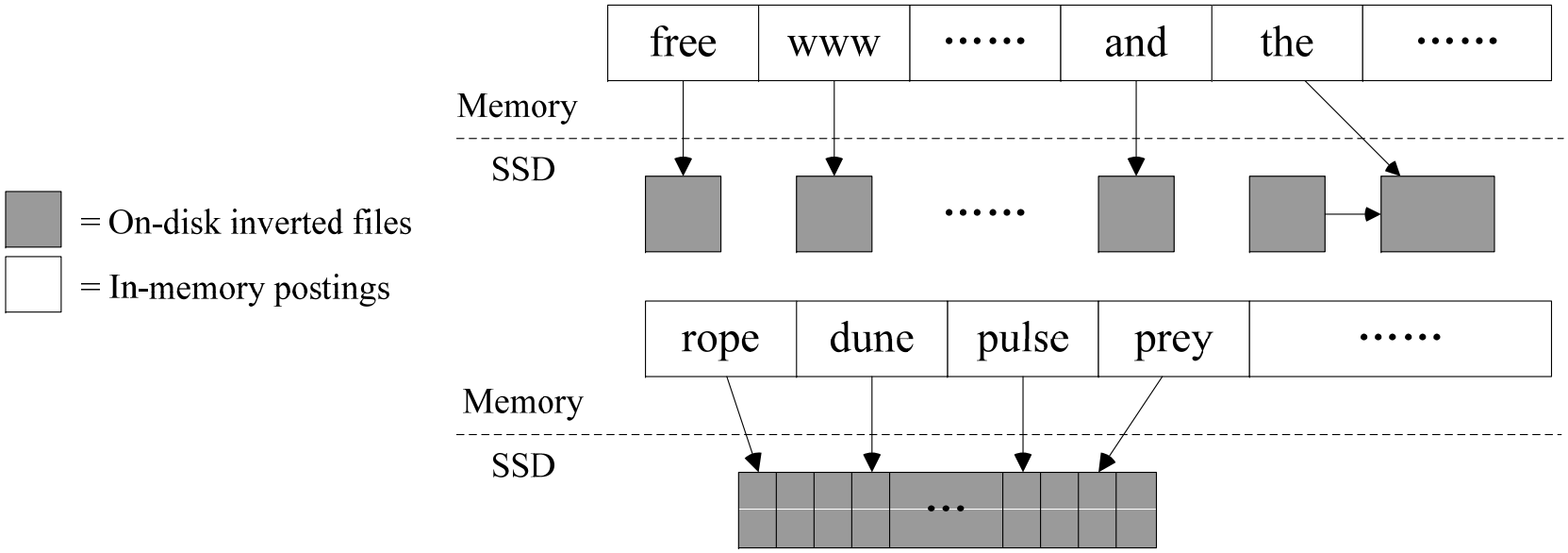
- The basic ideas about our design:
  - Avoid in-place updates to eliminate random writes during indexing
  - Categorize terms into short and long according to the size of their postings
  - For two types of terms we apply different merge-based methods respectively

Hybrid merge strategy

- For short terms, an indexing method with infrequent merge event is applied
  - Short terms are relatively infrequent in queries
  - Even the postings are separated into several inverted files, SSD can still ensure the speed of fetching them
- For long terms, “active” merge method is applied
  - They appear frequently in the query log
  - Speed of random read still cannot equal that of sequential read with large size of data
- As a result, many extra writes are saved and only the critical part of postings are maintained by frequently merging

No merge + Immediate merge

- Partitioning the in-memory index into blocks
- No merge for short terms with less postings
- Immediate merge for long terms with large size of postings



Selective memory flush

- Furthermore, we need to prevent merge event of the long term's postings from being triggered frequently
- Selective in-memory postings flush:
  - Flush fix-sized postings instead of the whole in-memory index
  - Flush the short term's postings as possible as we can
- Parameters
  - Posting Size S_p : the least postings size of the long term
  - Flushed Postings P_f : the least size of flushed postings for one flush from in-memory index to SSD

Selective memory flush

- 1. Parse documents into postings
- 2. When the memory is full
  - If the size of all short terms' postings can reach P_f, then flush them into SSD
  - Or flush the postings of long terms, from the one with the most postings until the size reaches P_f
- 3. Keep parsing

Outline

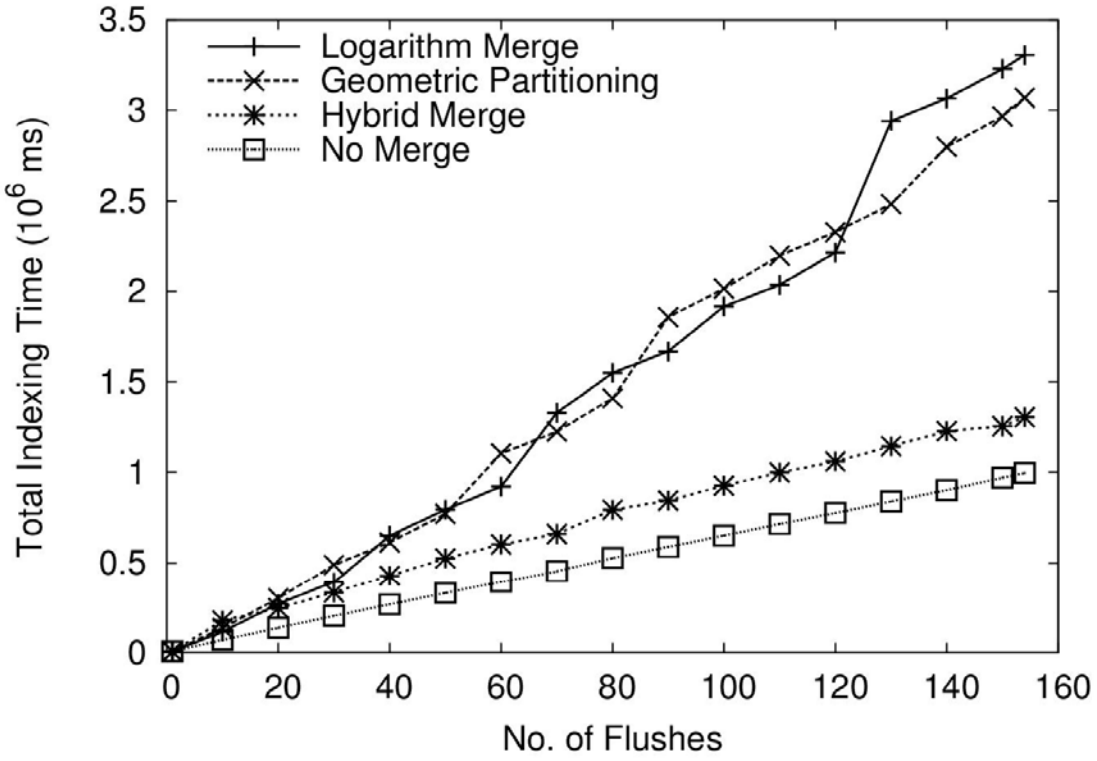
- Introduction
- Analysis of existing methods
- Hybrid Merge
- Evaluation
- Conclusion

Evaluation

- Experimental environment
  - PC: Intel Dual E2180 2.0GHz CPU, 2GB of RAM, one 320GB sized SATA hard disk, one 40GB sized Intel X25-M SSD
  - Text Collection: 106G Wikipedia Static HTML Dumps
  - Query log: AOL query log
- Compare with
  - Geometric Partitioning with $r = 3$ (GP)
  - Logarithmic Merge (LM)
  - No Merge (NM)
- Parameters:
  - S_p = 1MB, P_f = 40MB

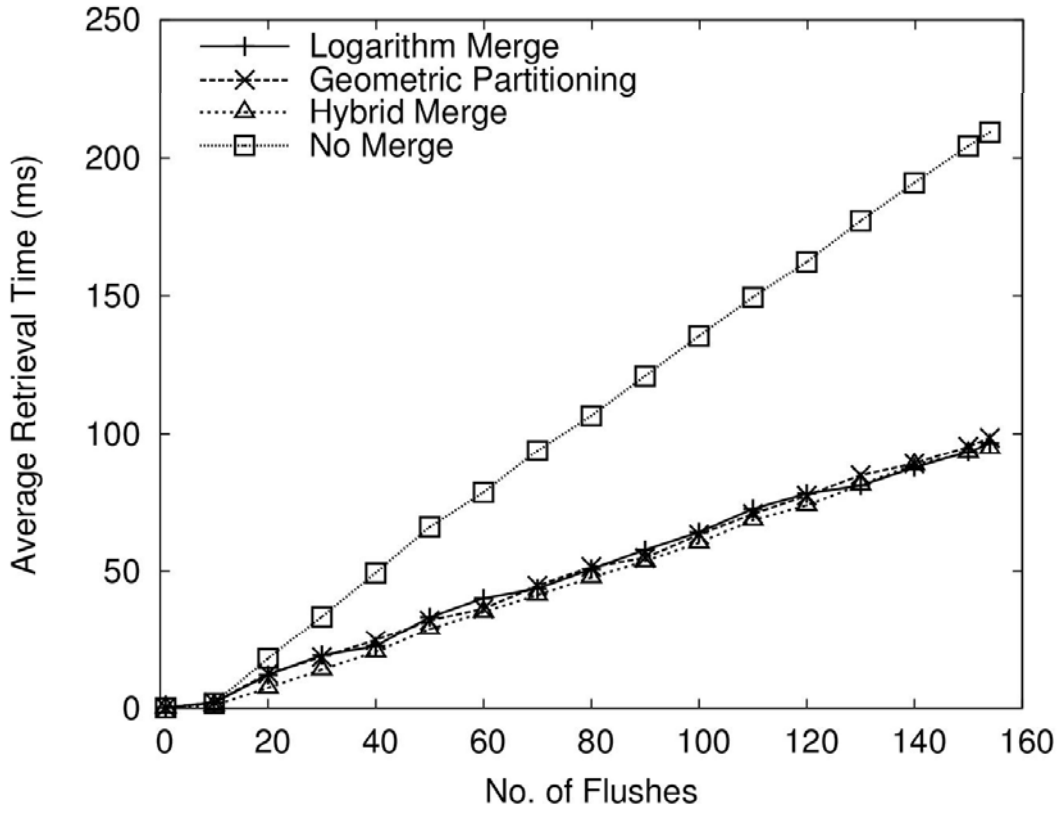
Indexing performance

- LM and GP cost nearly 3.3 and 3.1 times higher time than NM, while Hybrid Merge (HM) only costs 1.3 times higher time



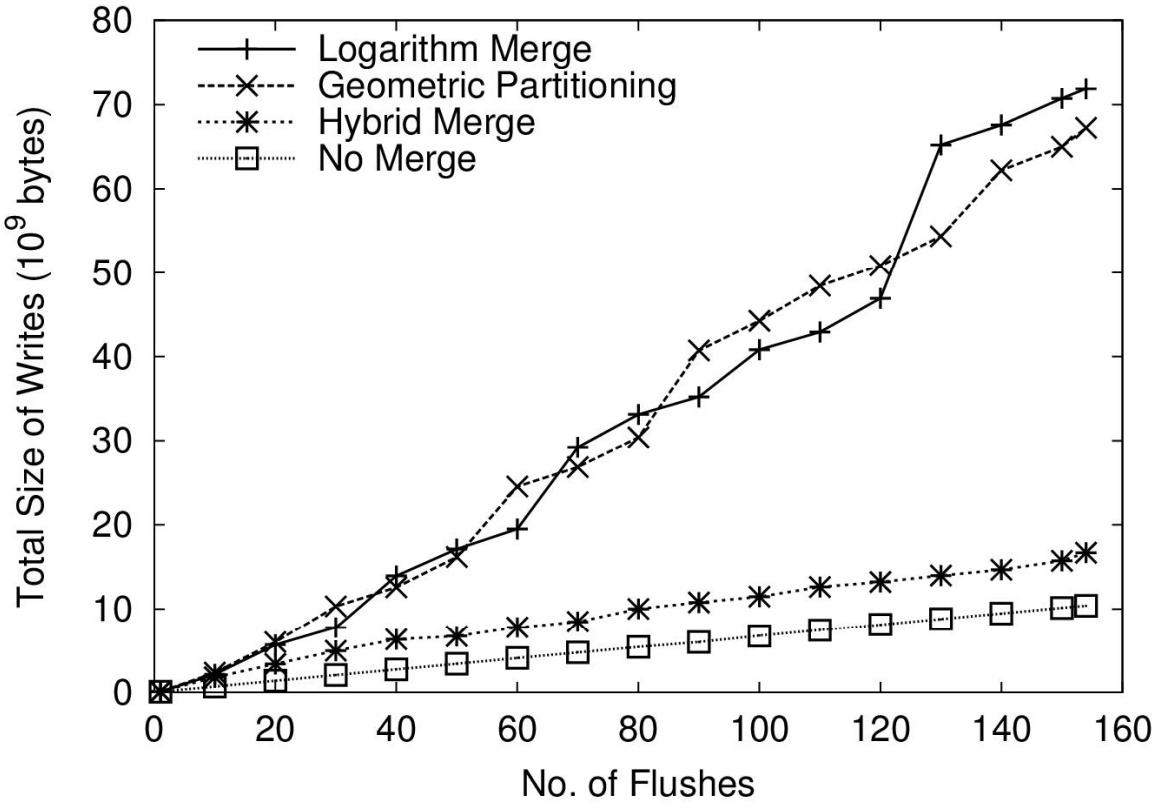
Query performance

- HM requires only about 98.0% and 96.0% of time to complete a query request compare to LM and GP, NM is the worst with no doubts



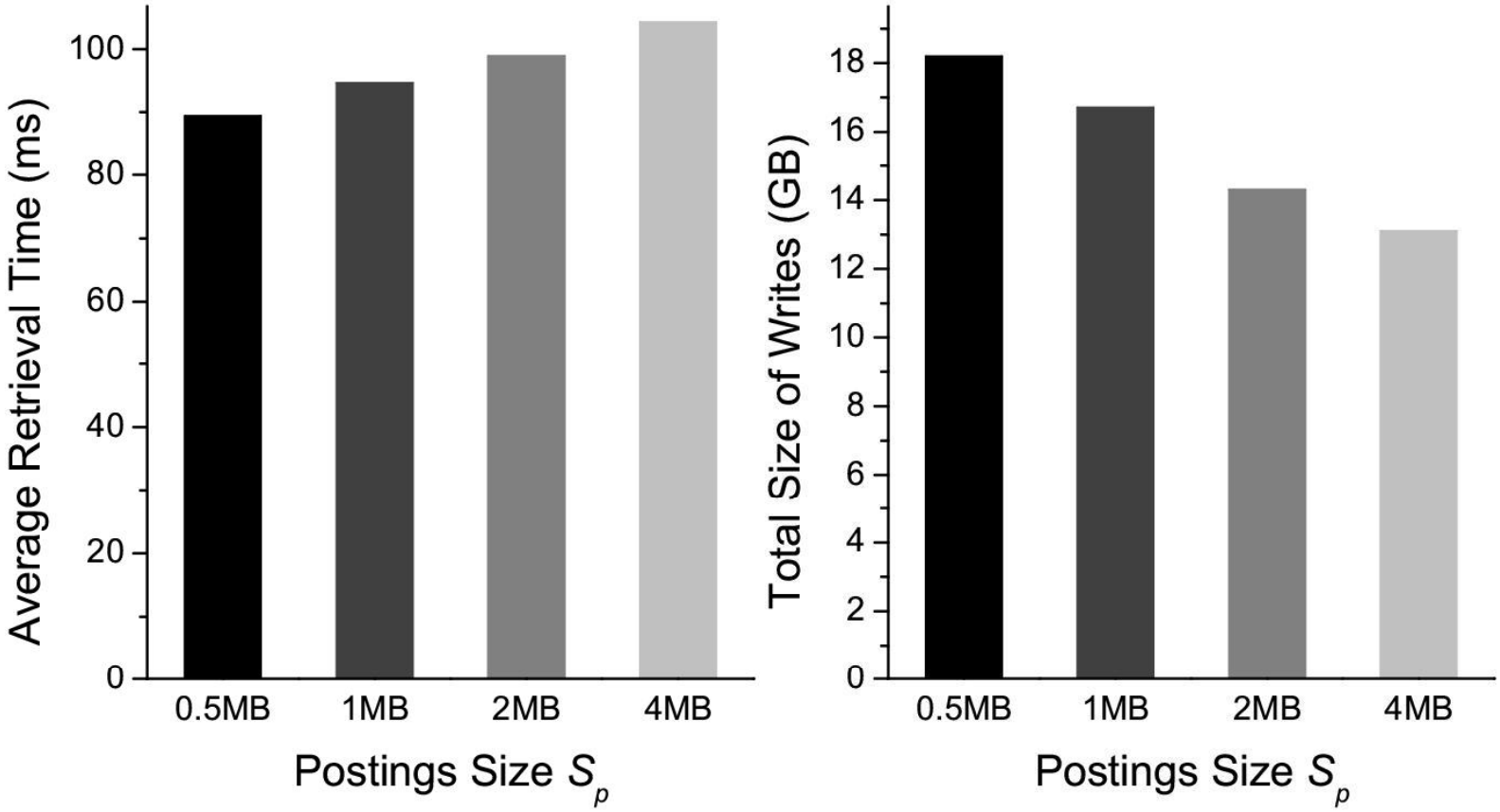
Total size of writes

- HM writes only 1.6 times larger size of data than NM and only writes 23.2% and 24.9% of data compare to LM and GP



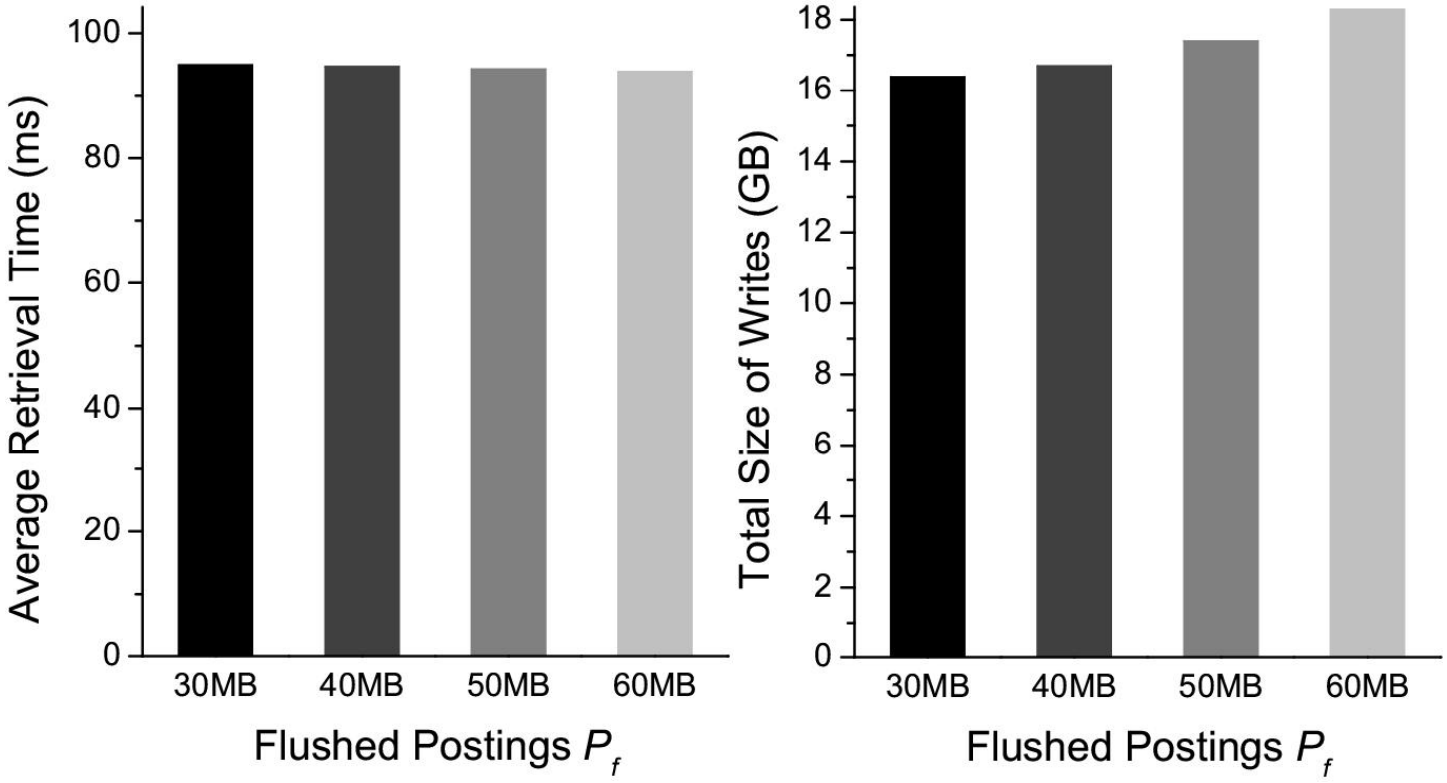
Impact of parameter configuration

- Query performance degrades and the total size of writes drops with the increase of S_p



Impact of parameter configuration

- Query performance drops slightly with the increasing value of P_f while amount of write traffic raises significantly



Outline

- Introduction
- Analysis of existing methods
- Hybrid Merge
- Evaluation
- Conclusion

Conclusions & future work

- Existing hard disks based index maintenance approaches are analyzed and found no longer suitable on SSD
- a new SSD based indexing strategy is proposed
- Hybrid Merge performs more suitable on SSD than conventional way
- Future work:
  - Test our method on various types of SSDs
  - Apply automatic adjustment of the configuration parameters

Thanks for your attention

Contact information:

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