Fast On-Line Index Construction by Geometric Partitioning

Nicholas Lester, Alistair Moffat, and Justin Zobel

Proceedings of the 14th ACM International Conference on Information and Knowledge Management. ACM, 2005.

Hybrid Index Maintenance for Growing Text Collections

Stefan Büttcher, Charles LA Clarke, and Brad Lushman

Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2006.

Presented by Matt Chaney
CS 834 - Presentation 2

Inverted Indexes

- Similar to index in book
- Composed of two main parts
  - Vocabulary
  - Set of inverted lists, some with extra info
    - Frequencies
    - Position
- Normally stored on disk as a single, contiguous partition

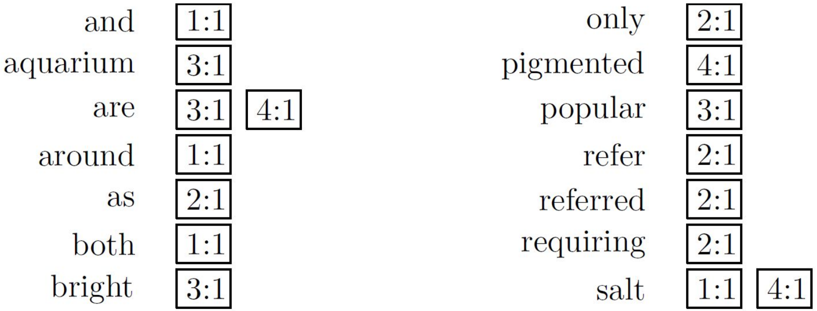


Fig. 5.5. Inverted Index with Counts.
Yang, Christopher C. "Search engines information retrieval in practice." (2010).

Problem

- Adding new documents challenging
- Two basic strategies
  - Off-Line
    - Index creation simple process
    - Lag between document discovery and indexing
  - **On-Line**
    - When delay cannot be tolerated
    - Index always queryable and up to date
    - Adds complexity to system
- On-Line Merge Process
  - In-Memory for new documents
  - On-Disk for existing index

Index Maintenance Algorithms

- Rebuild
  - Simplest
  - Reprocess entire index on disk and switch once complete
  - 10^{10} total items with 10^{8} in-memory requires 17.1 trillion operations

- Remerge
  - Avoids reprocessing already sorted lists
  - Uses *bufferloads* to append new items to each on-disk inverted list
  - Read sequentially and written to new location while inserting in-memory postings
  - 10^{10} total items with 10^{8} in-memory require 0.8 trillion operations

Index Maintenance Algorithms

- In-place Update
  - Write in-memory postings to end of existing lists
  - When on-disk full move to new location
  - Can employ disk over-allocation to avoid excessive file moves
  - 10^{10} total items with 10^8 in-memory require 0.3 trillion operations, *but...*
  - Metric is misleading
    - Memory overhead - 60% allocated and not used
    - Only counts *data movement* - Each term generates disc op ➔ +1 trillion more ops

- Multiple Partitions
  - Create discontiguous fragments as inverted lists grow
  - Similar to In-Place without reprocessing old lists

Geometric Partitioning

- Controlled number of partitions

- Two approaches
  - Fixed partition size, *r*
  - Fixed number of partitions, *p*

- Blends the Remerge and Multiple Partitions approach
  - Better construction time than Remerge
  - Better query processing time than Multiple Partitions

- New documents in smaller partitions
- Old documents in larger partitions

Geometric Partitioning

- Controlled size, static *r*
  - Ex. partition 3 is 2x size of partition 2, which is 2x size of partition 1
  - Most recent documents go into smallest partition
- Leads to a hierarchical merging scheme

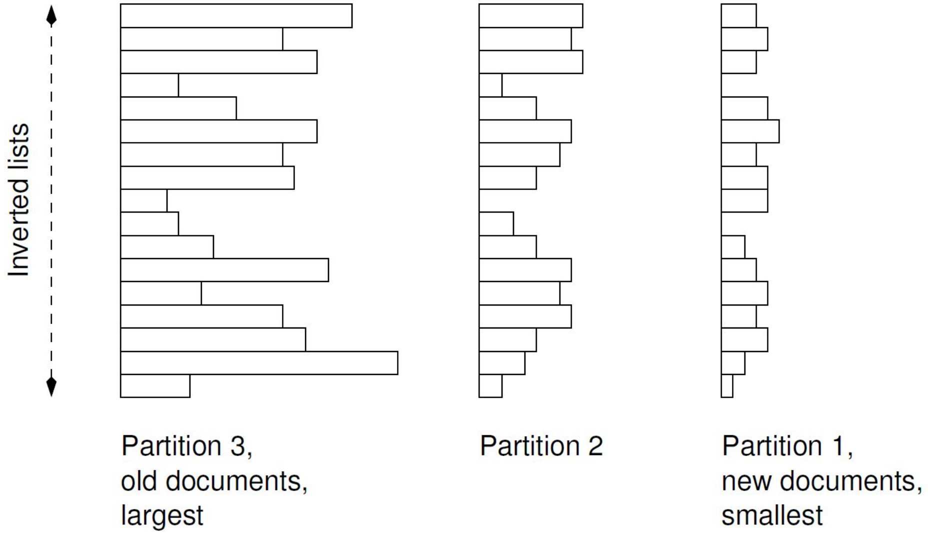
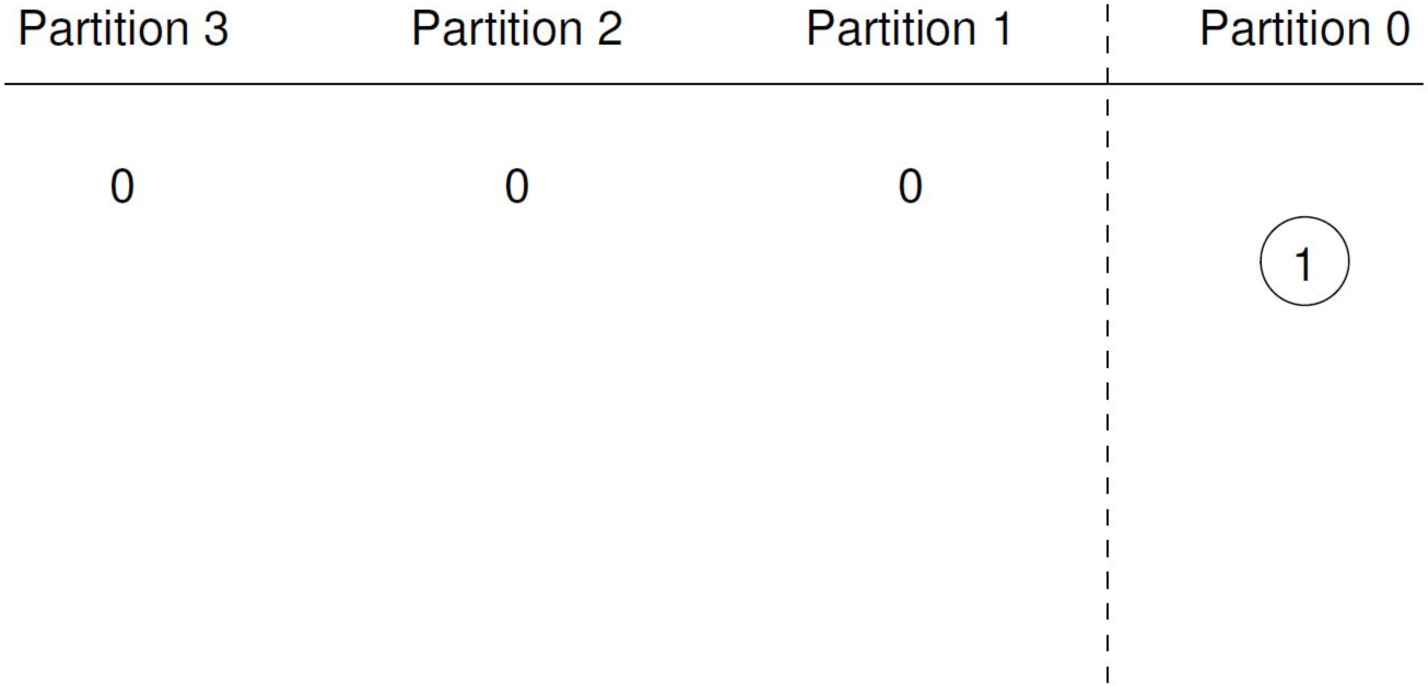


Fig. 1. Geometric Partitioning example.
Lester, Nicholas, Alistair Moffat, and Justin Zobel. "Fast on-line index construction by geometric partitioning." Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, 2005.

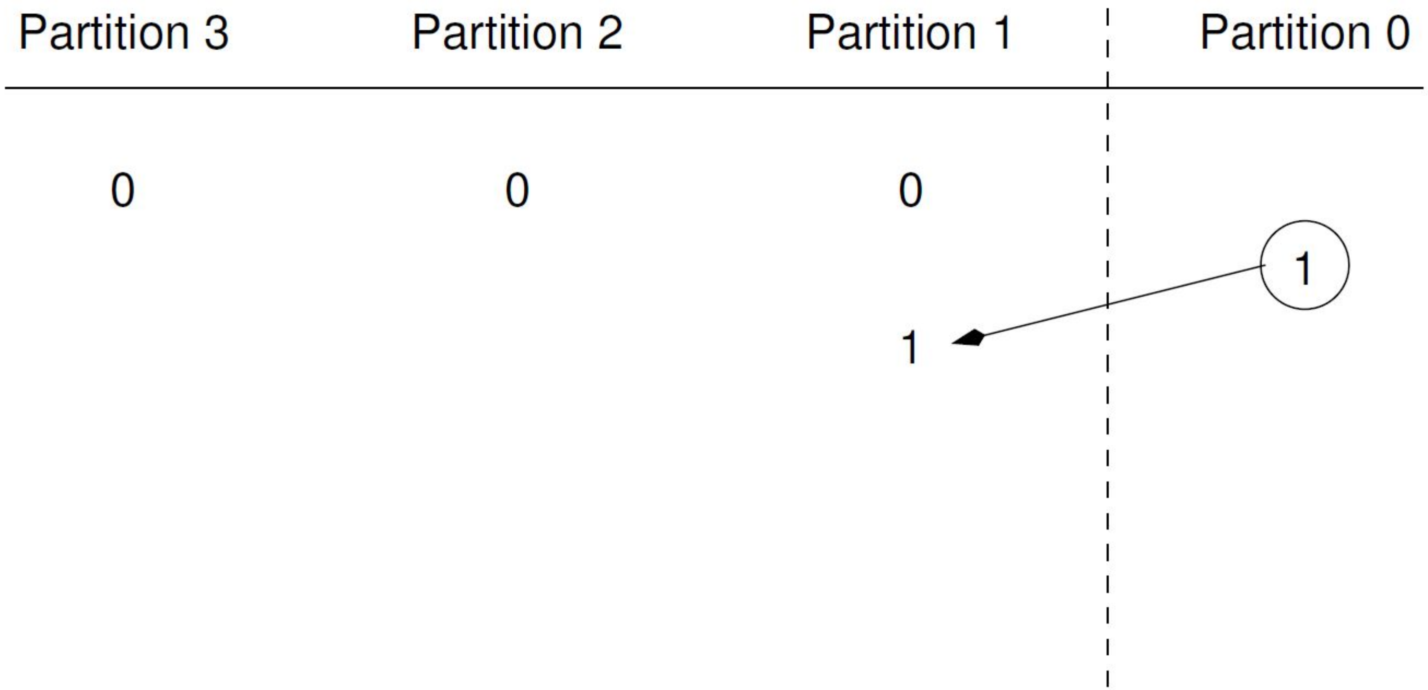
Hierarchical Merging, r = 3

Partition 3	Partition 2	Partition 1	Partition 0
0	0	0	

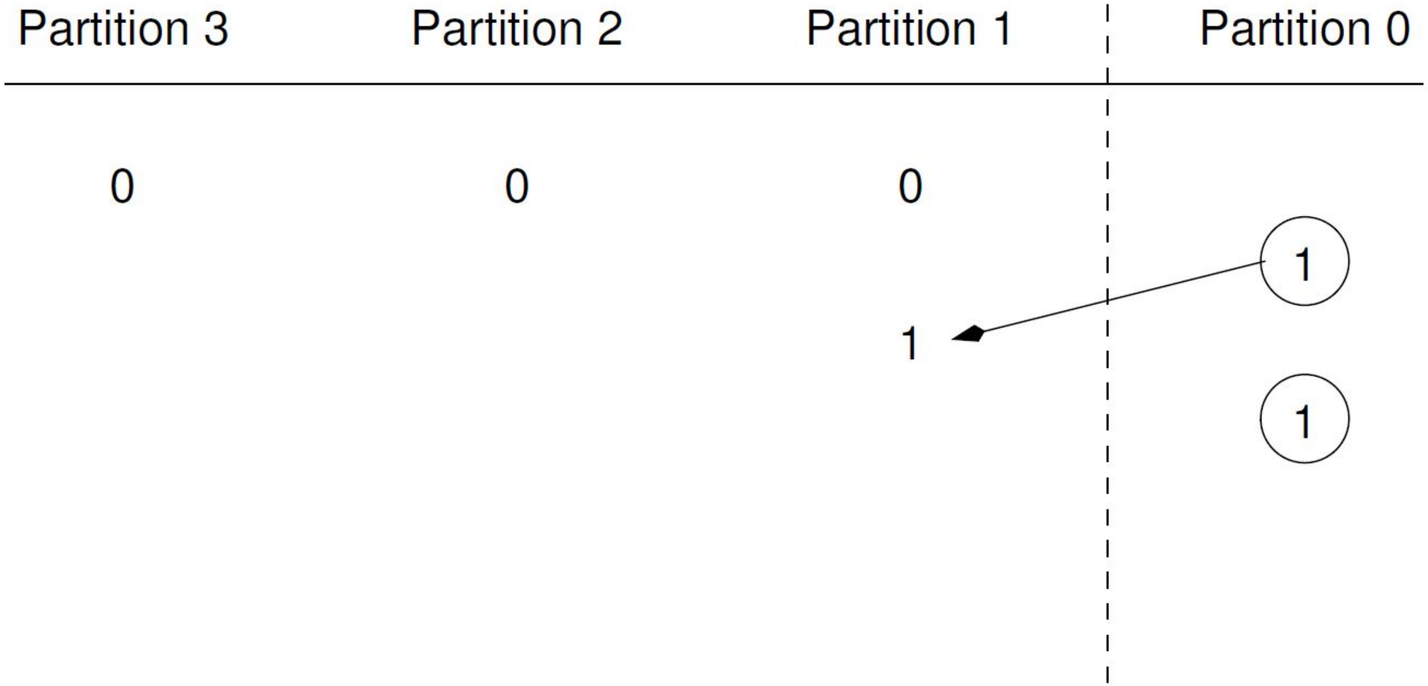
Hierarchical Merging, r = 3



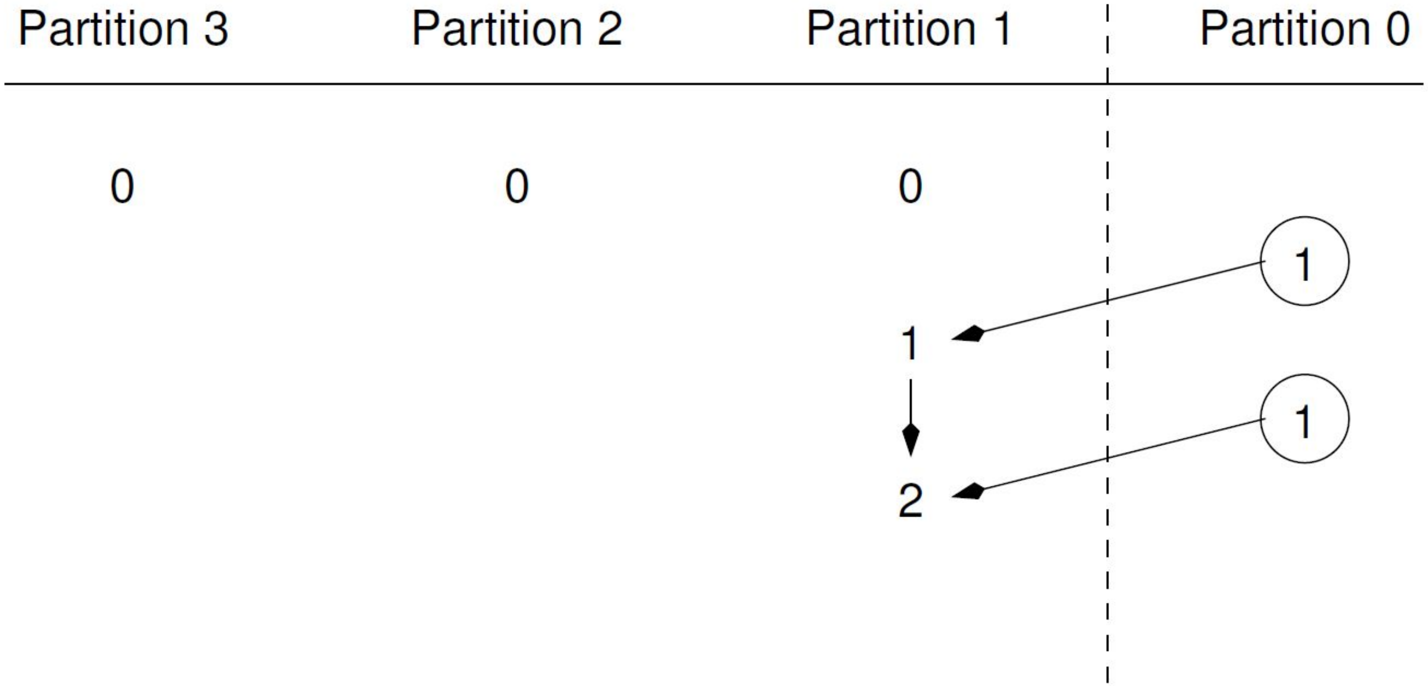
Hierarchical Merging, r = 3



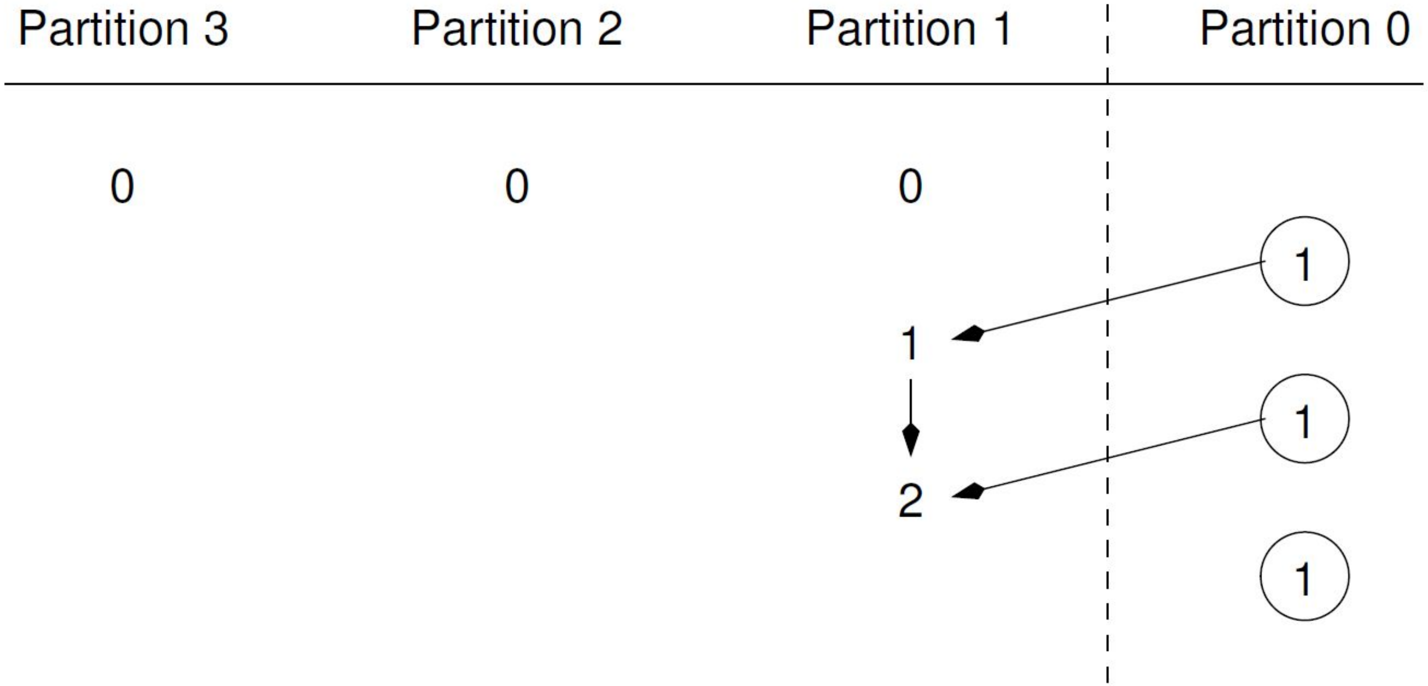
Hierarchical Merging, r = 3



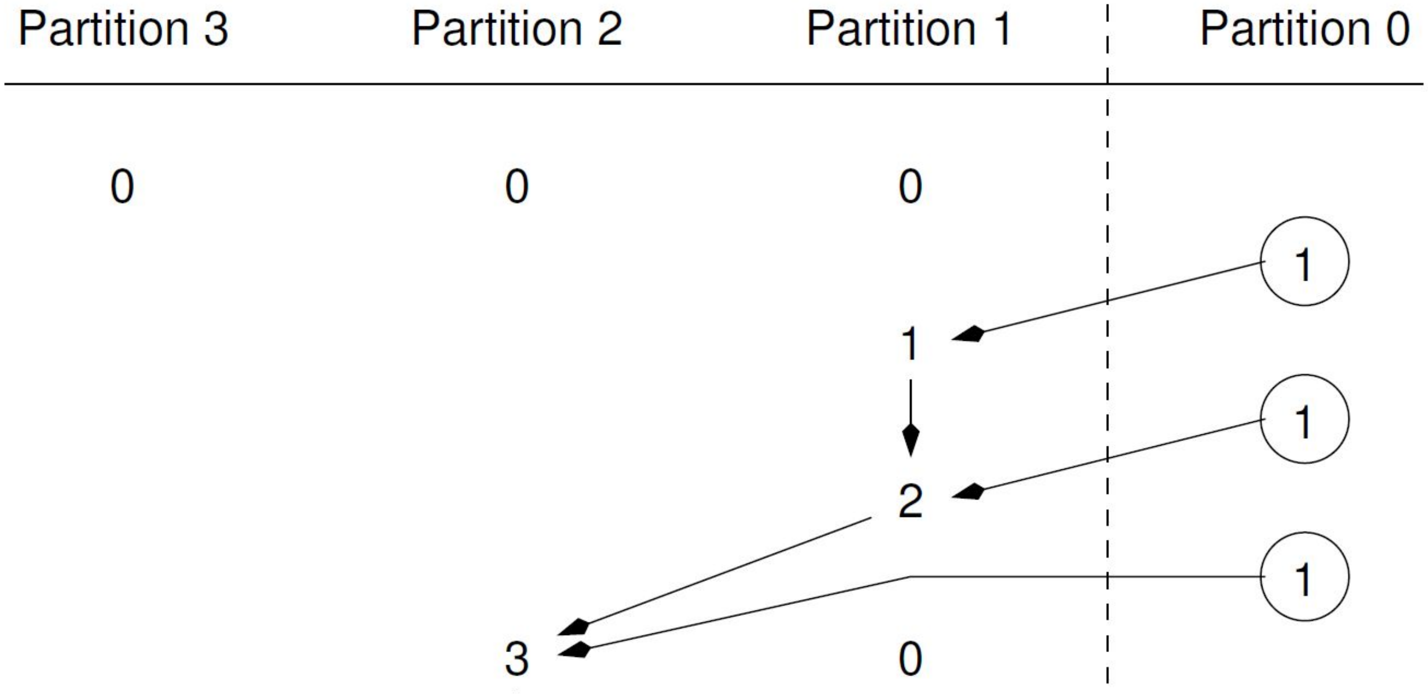
Hierarchical Merging, r = 3



Hierarchical Merging, r = 3



Hierarchical Merging, r = 3



Merge with fixed *r*

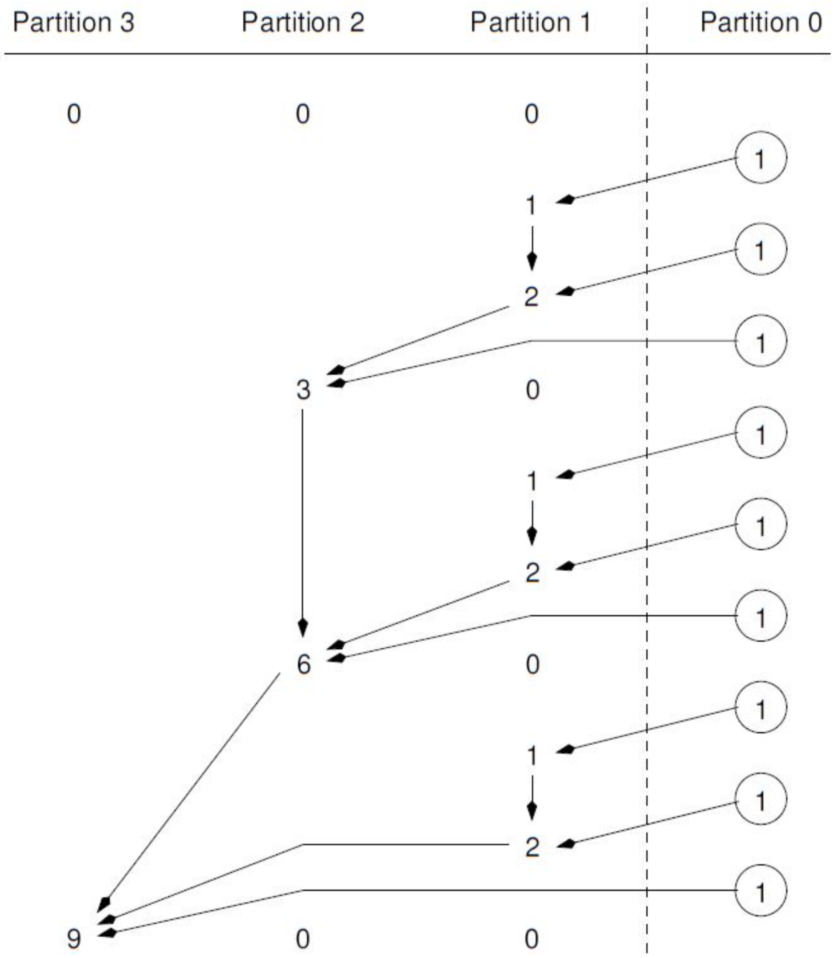


Figure 2. Merge with fixed r = 3.
Lester, Nicholas, Alistair Moffat, and Justin Zobel. "Fast on-line index construction by geometric partitioning." Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, 2005.

Merge with fixed *p*

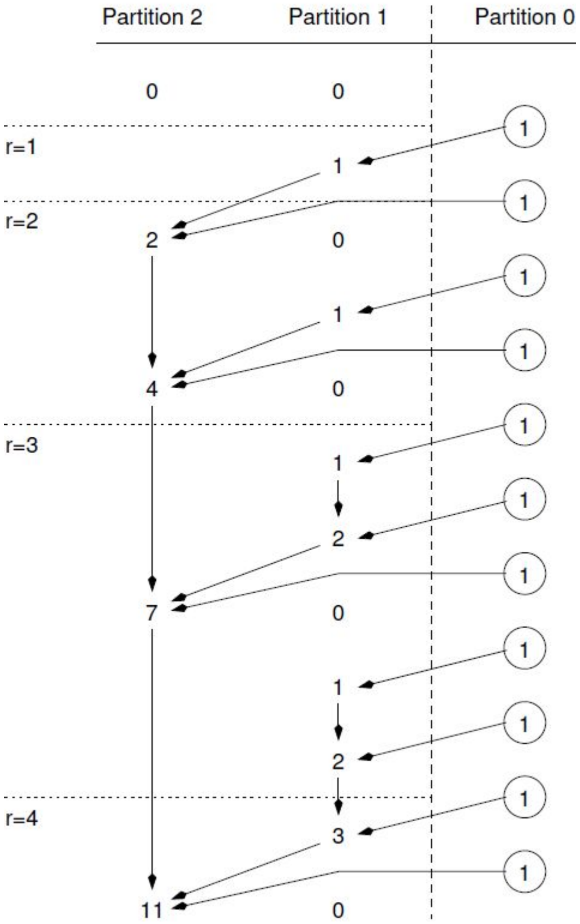


Figure 3. Merge with fixed p = 3.
Lester, Nicholas, Alistair Moffat, and Justin Zobel. "Fast on-line index construction by geometric partitioning." Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, 2005.

Analysis - Construction vs Access

r	Build cost	Access cost
2	0.29	3.6
3	0.31	3.1
4	0.32	2.9
6	0.34	2.6
8	0.35	2.4
12	0.39	2.2
16	0.42	2.0

Table 1. Construction cost versus access cost.
Lester, Nicholas, Alistair Moffat, and Justin Zobel. "Fast on-line index construction by geometric partitioning." Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, 2005.

Analysis - Index Construction Time

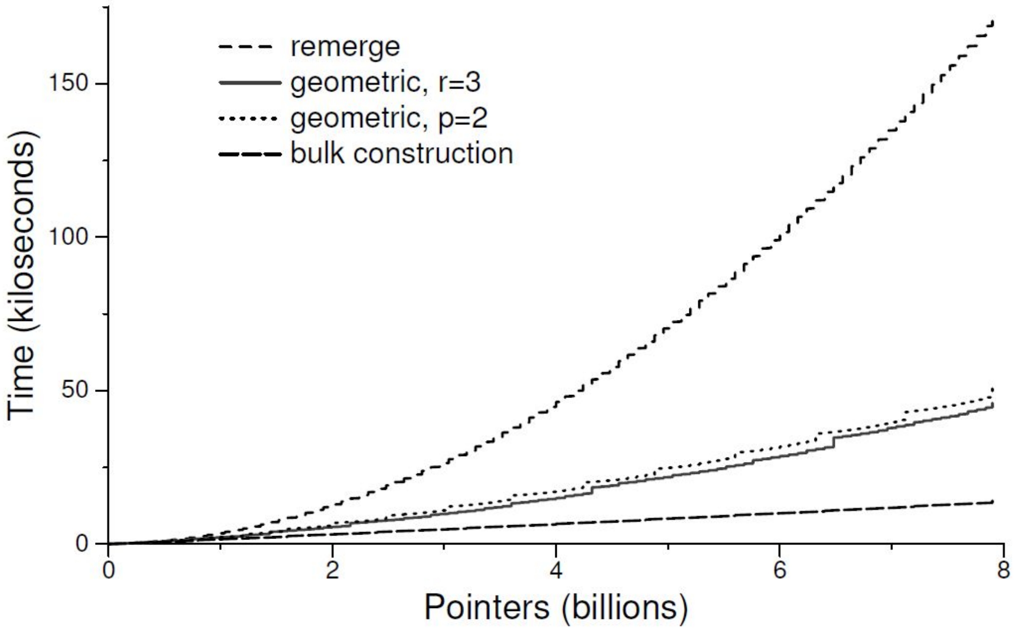
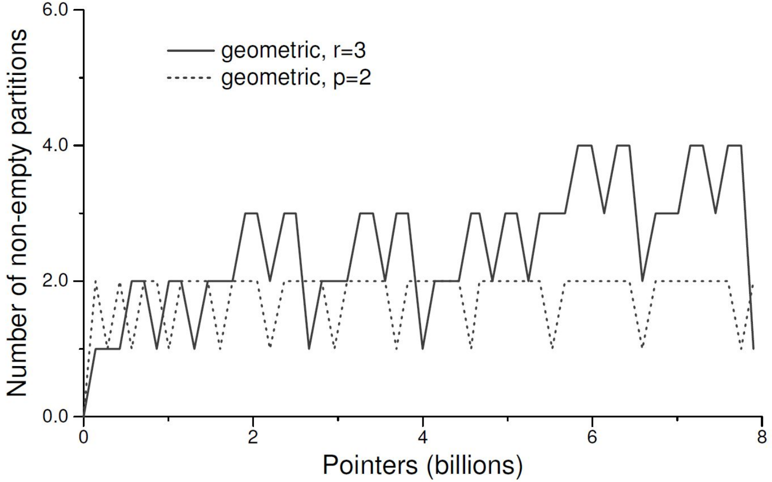


Figure 4. Build time vs number of pointers.
Lester, Nicholas, Alistair Moffat, and Justin Zobel. "Fast on-line index construction by geometric partitioning." Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, 2005.

Analysis - Fixed *r* vs Fixed *p*



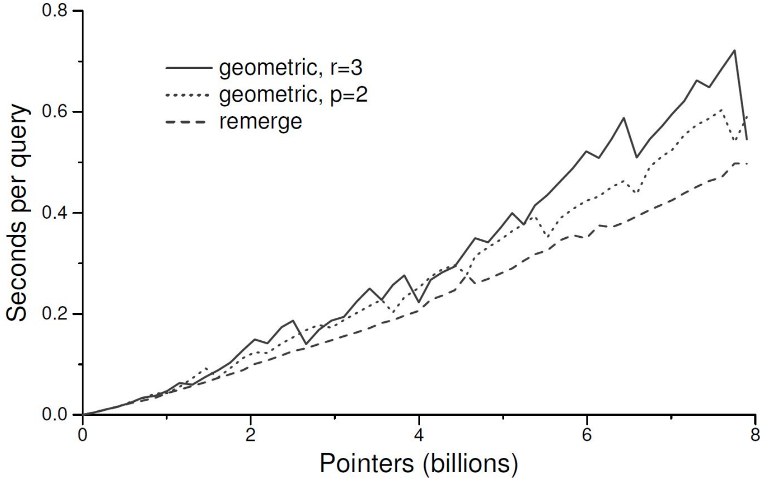


Figure 5. Query time versus number of pointers.
Lester, Nicholas, Alistair Moffat, and Justin Zobel. "Fast on-line index construction by geometric partitioning." Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, 2005.

Conclusions

- Index construction cost significantly reduced versus standard contiguous inverted indexes

- 100GB collection index construction 4x faster than Remerge

- As index grows, speed gain increases
  - Adding last Gig of collection takes
    - 1 hour with Remerge
    - 9 minutes with Geometric Partitioning

- Query time only 20% slower than Remerge

Fast On-Line Index Construction by Geometric Partitioning

Nicholas Lester, Alistair Moffat, and Justin Zobel

Proceedings of the 14th ACM International Conference on Information and Knowledge Management. ACM, 2005.

Hybrid Index Maintenance for Growing Text Collections

Stefan Büttcher, Charles LA Clarke, and Brad Lushman

Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2006.

Presented by Matt Chaney

CS 834 - Presentation 2

In-Place Index Maintenance

- In-memory data appended to on-disk lists without merging
- Generally requires relocating on-disk lists
  - Can be costly with large collections
- Over-allocating can reduce cost

Merge-Based Index Maintenance

- **Immediate Merge** - Rebuild/Remerge
  - Maintains one in-memory and one on-disk index
  - When in-memory full → merge with on-disk
  - Re-processes entire on-disk image
  - Still performs better than In-Place

- **Sqrt Merge**
  - Geometric Partitioning with fixed p = 2

- **Logarithmic Merge**
  - Geometric Partitioning with fixed r

Motivation

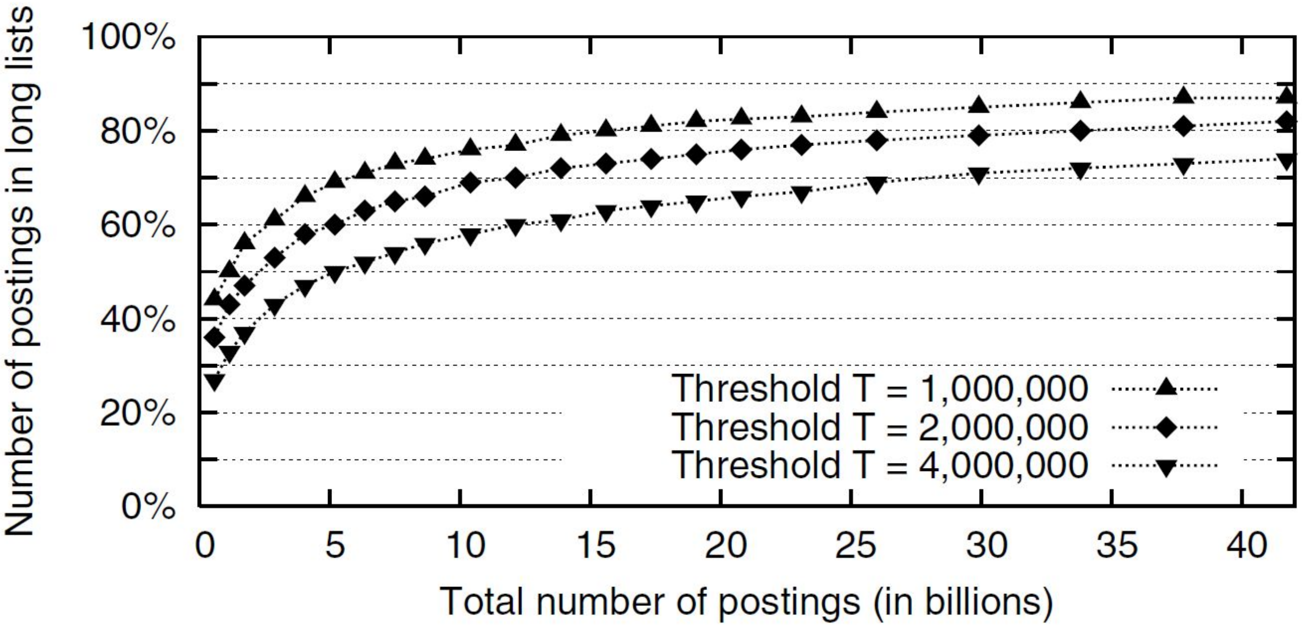


Figure 2. Total number of postings in the index vs. number of postings in long lists.
Büttcher, Stefan, Charles LA Clarke, and Brad Lushman
Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2006.

First Hybrid Strategy

- Combines Merging and In-Place approaches
- Quantity *T* designates *short* or *long* term list length
  - Short lists managed with merge strategy
  - Long lists managed with in-place using filesystem, stored contiguously
- Avoids costly on-disk relocations of long posting lists
- Combined hybrid versions (“C” indicates contiguous lists)
  - Hybrid Immediate Merge ➔ HIM_C
  - Hybrid Sqrt Merge ➔ HSM_C
  - Hybrid Logarithmic Merge ➔ HLM_C

First Hybrid Strategy Flaws

- File system in-place update hard to analyze
  - File system operation differs system to system
  - Cannot be sure if posting lists are actually contiguous

- Static per-term *T* value inflexible for ever growing index
  - Highly sensitive to the amount of main memory available

New Hybrid Strategy

- Terms lists split into short and long, non-contiguous portions
  - Query time suffers, but disk reads can be amortized over longer read op
  - Relative query time increases only 5% over old hybrid

- New Hybrid ("NC" indicates non-contiguous lists)
  - Hybrid Immediate Merge → HIM_{NC}
  - Hybrid Sqrt Merge → HSM_{NC}
  - Hybrid Logarithmic Merge → HLM_{NC}

New Hybrid Strategy Illustrated

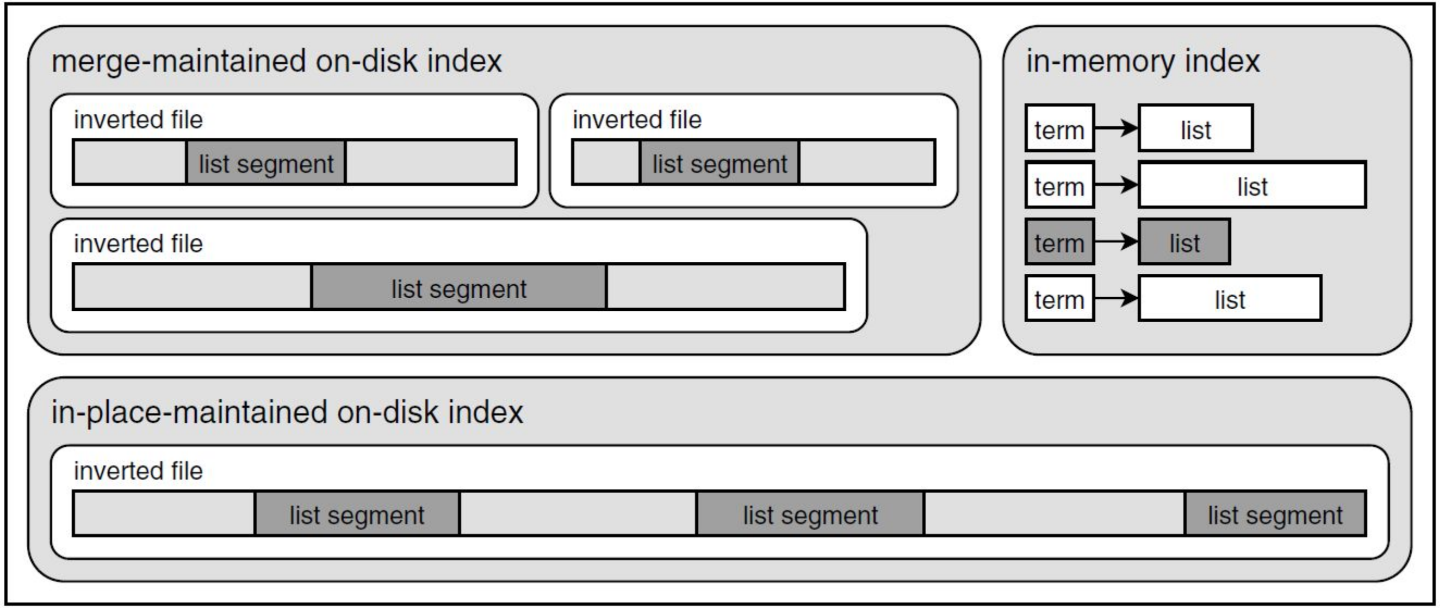


Figure 3. Index layout for a hybrid maintenance strategy with non-contiguous posting lists.
Büttcher, Stefan, Charles LA Clarke, and Brad Lushman
Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2006.

Indexing Time and Query Processing Performance

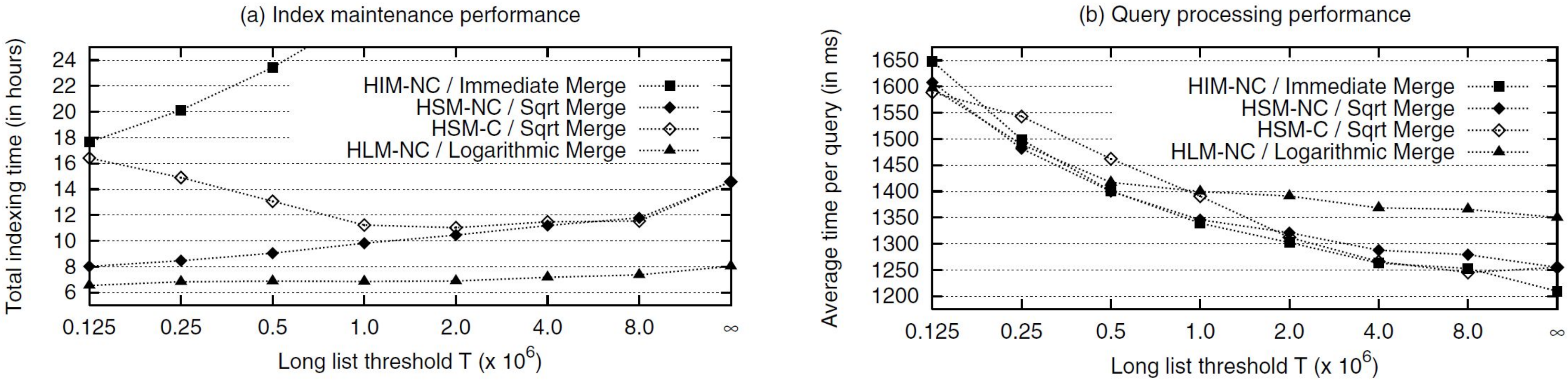


Figure 5. Index maintenance and query processing performance for different strategies with various parameter settings.
Büttcher, Stefan, Charles LA Clarke, and Brad Lushman
Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2006.

Memory Size versus Maintenance Speed

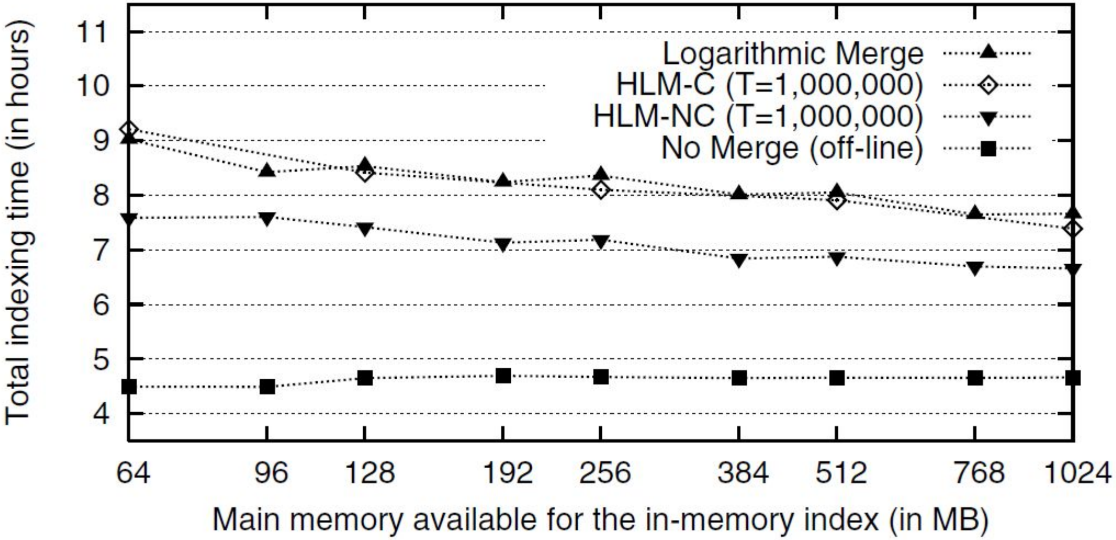


Figure 6. Impact of memory size on index maintenance performance.
Büttcher, Stefan, Charles LA Clarke, and Brad Lushman
Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2006.

Conclusions

- New Hybrid approach outperforms previous index maintenance strategies including previous hybrid approach

- HLM_{NC} require only $\Theta(N)$ disk operations for text collection of size N

- Main shortfall is increased query times due to multiple partition fragments, future work could improve this metric with over-allocating disk space