Let's Squeeze Memory out of Index Structures

Huanchen Zhang (CMU), Andrew Pavlo (CMU), David G. Andersen (CMU), Michael Kaminsky (Intel Labs)

Motivation & Goal

Project Goal: Reduce memory footprints in main memory OLTP database systems.

Why? Memory hit rate determines the performance of a main memory OLTP database system. Reducing memory footprints gives the system more space to cache frequently accessed data → # disk seeks is reduced.

Problem: A significant fraction of memory space is dedicated to index structures.

Solution: Treat "hot" and "cold" data differently when creating index entries.

Hybrid Masstree Index

Context: Anti-Caching

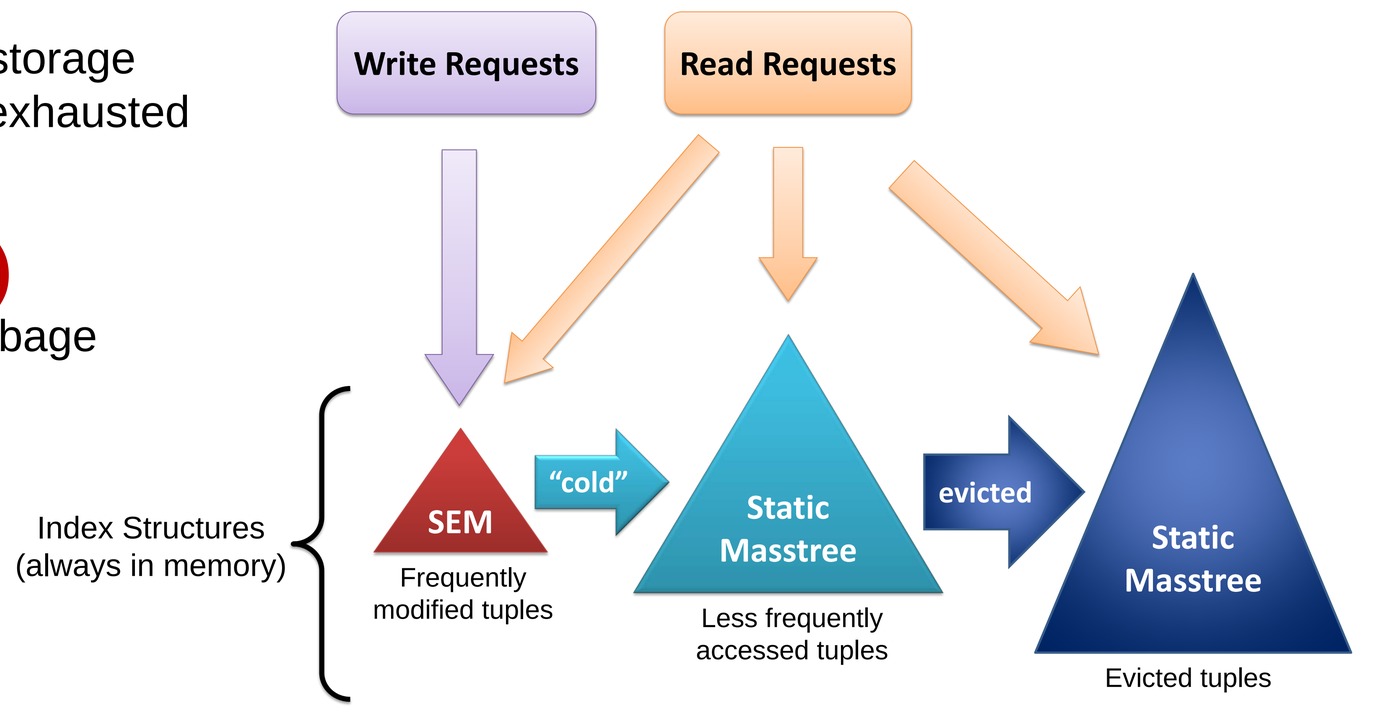
- Similar to "paging" in virtual memory
- Main memory (rather than disk) is the primary storage
- "Cold" data is evicted to disk when memory is exhausted
  [Anti-Caching, VLDB 2013]

Space-Efficient Masstree (SEM)

Masstree [Eurosys 2012] with more effective garbage collection and more efficient memory allocation

Static Masstree

A compact, read-only version of Masstree
(Please refer to poster "**Pruning Masstree**" for detail)



Mechanism

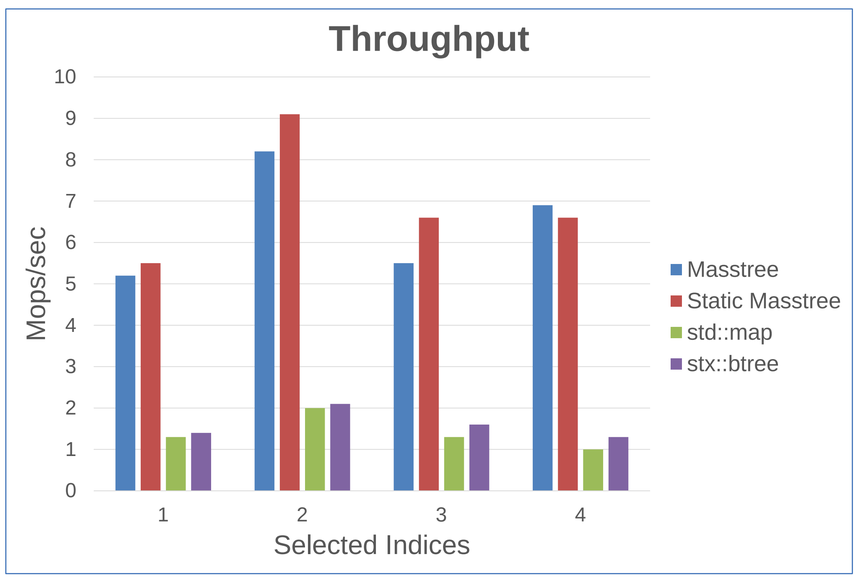
- Index entries for "hot" tuples stay in Space-Efficient Masstree while those for "cold" tuples are periodically merged to Static Masstree
- Use hints provided by the Anti-Caching Logic to decide when and what tuples to migrate

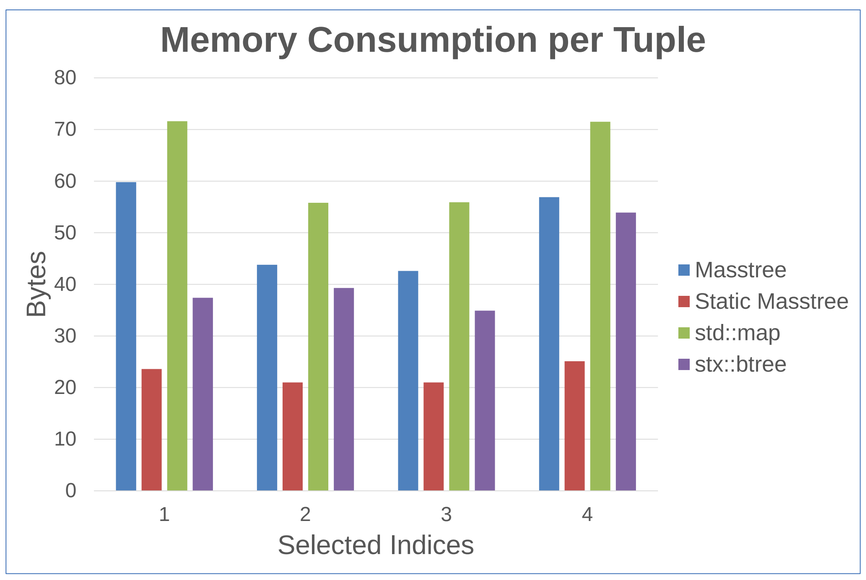
Preliminary Results: Less Memory AND Faster

Workload: **TPC-C**

	key size (B)	# entries	workload
Index 1	11	47,105	100% put
Index 2	7	3,000	1.7% put 98.3% get
Index 3	6	100,000	5.6% put 94.4% get
Index 4	11	471,051	100% put

Value size = 8B for all indices
CPU: Intel Core i7-4770, 3.4GHz
L2 cache size: 8MB





Intel Science & Technology Center for Cloud Computing
Carnegie Mellon University
Georgia Tech
intel
PRINCETON UNIVERSITY
UC Berkeley
UNIVERSITY of WASHINGTON