Reducing the Storage Overhead of Main-Memory OLTP Databases with Hybrid Indexes

Huanchen Zhang

Co-authors:
David G. Andersen
Andrew Pavlo
Michael Kaminsky
Lin Ma
Rui Shen
You are running out of memory
You are running out of memory
You are running out of memory

Buy more?
TPC-C on H-Store

Transactions Executed

Throughput

Memory (GB)

Transactions Executed

Memory Limit = 5GB

Disk tuples

In-memory tuples

Indexes

Transactions Executed

Memory (GB)

Transactions Executed

Memory (GB)

Transactions Executed

Memory (GB)
I GOT STUCK

SO I WENT TO SLEEP
The better way:
Use memory more efficiently
Indexes are **LARGE**

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>% space for index</th>
<th>Hybrid Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC-C</td>
<td>58%</td>
<td>34%</td>
</tr>
<tr>
<td>Voter</td>
<td>55%</td>
<td>41%</td>
</tr>
<tr>
<td>Articles</td>
<td>34%</td>
<td>18%</td>
</tr>
</tbody>
</table>
Our Contributions

1. The hybrid index architecture
2. The Dual-Stage Transformation
3. Applied to 4 index structures
   - B+tree
   - Masstree
   - Skip List
   - Adaptive Radix Tree (ART)

Performance ≈ Space

30 – 70%
Did we solve this problem?

Stay tuned

Transactions Executed
How do hybrid indexes achieve memory savings?

Static
Hybrid Index: a dual-stage architecture

dynamic stage

static stage
Inserts are batched in the dynamic stage

dynamic stage → static stage
Reads search the stages in order

1. Dynamic stage
2. Static stage
A Bloom filter improves read performance

1. Dynamic stage
2. Static stage

read
Dynamic stage

Static stage

1. read
2. read
read
write
merge

Memory-efficient

Skew-aware
The Dual-Stage Transformation

1. dynamic stage
2. static stage
3. merge
The Dynamic-to-Static Rules

- Compaction
- Reduction
- Compression
Compaction: minimize # of memory blocks
Compaction: minimize # of memory blocks
Reduction: minimize structural overhead
Reduction: minimize structural overhead
The Dual-Stage Transformation

Merge Questions:
1. Partial?
2. When?
3. Blocking?

dynamic stage  static stage
Did we solve this problem?

Transactions Executed
Yes, we improved the DBMS’s capacity!

Transactions Executed

TPC-C on H-Store

Throughput (txn/s)

B+tree

Hybrid
TPC-C on H-Store

Memory (GB)  Throughput (txn/s)

Transactions Executed

- Hybrid
- B+tree
- Disk tuples
- In-memory tuples
- Indexes

20K  60K  20K  60K

Hybrid  B+tree

2M  4M  6M  8M  10M
Transactions Executed

Throughput (txn/s)

Memory (GB)

TPC-C on H-Store

B+tree

Hybrid

Disk tuples

In-memory tuples

Indexes

Transactions Executed
Transactions Executed

Throughput (txn/s)

Memory (GB)

TPC-C on H-Store

B+tree

Hybrid

Disk tuples

In-memory tuples

Indexes

Transactions Executed
Take Away:

Memory saved by indexes → Larger working set in memory → Higher throughput

Transactions Executed

Memory (GB)

0 2M 4M 6M 8M 10M

B+tree

There are two graphs, one for B+tree and one for Hybrid, showing the memory usage and transactions executed. The graphs indicate that the memory saved by indexes leads to a larger working set in memory, resulting in a higher throughput.
This is just the **BEGINNING**
Conclusions

1. The hybrid index architecture
2. The Dual-Stage Transformation
3. Applied to 4 index structures
   - B+tree
   - Masstree
   - Skip List
   - Adaptive Radix Tree (ART)
Toll-Free Hotline:

1-844-888-CMUDB