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

Huanchen Zhang

Co-authors:

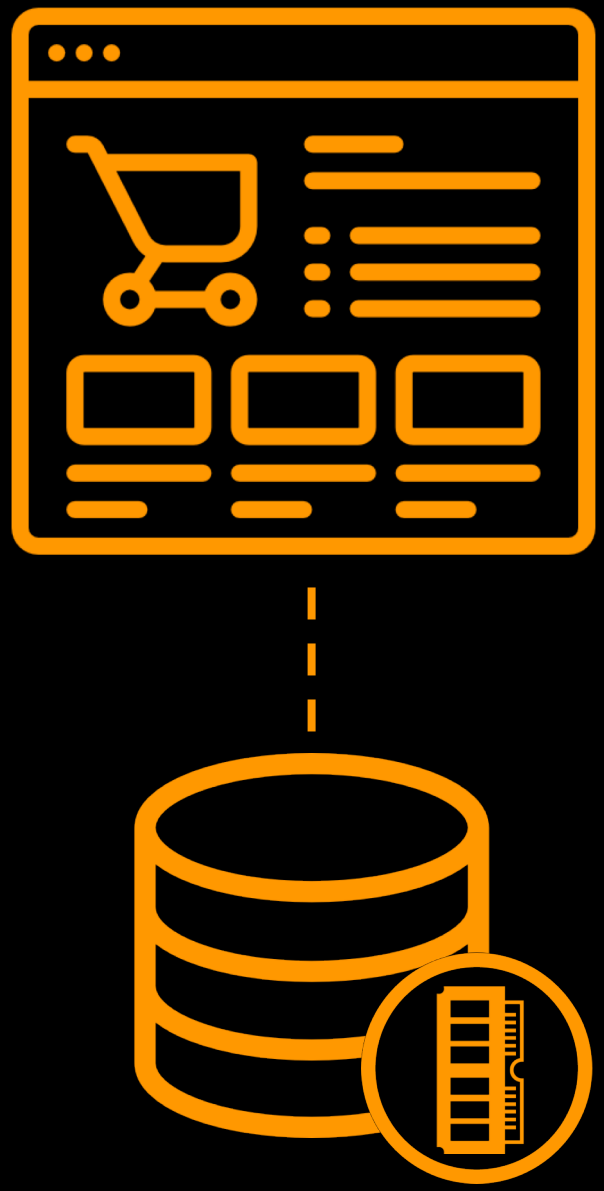
David G. Andersen

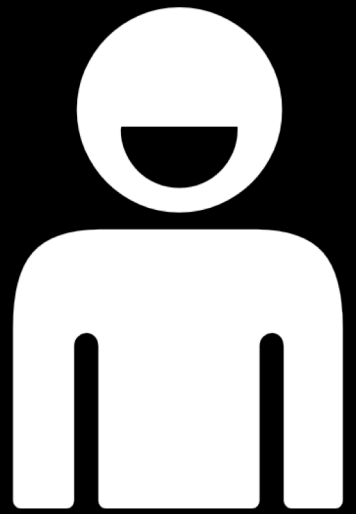
Andrew Pavlo

Michael Kaminsky

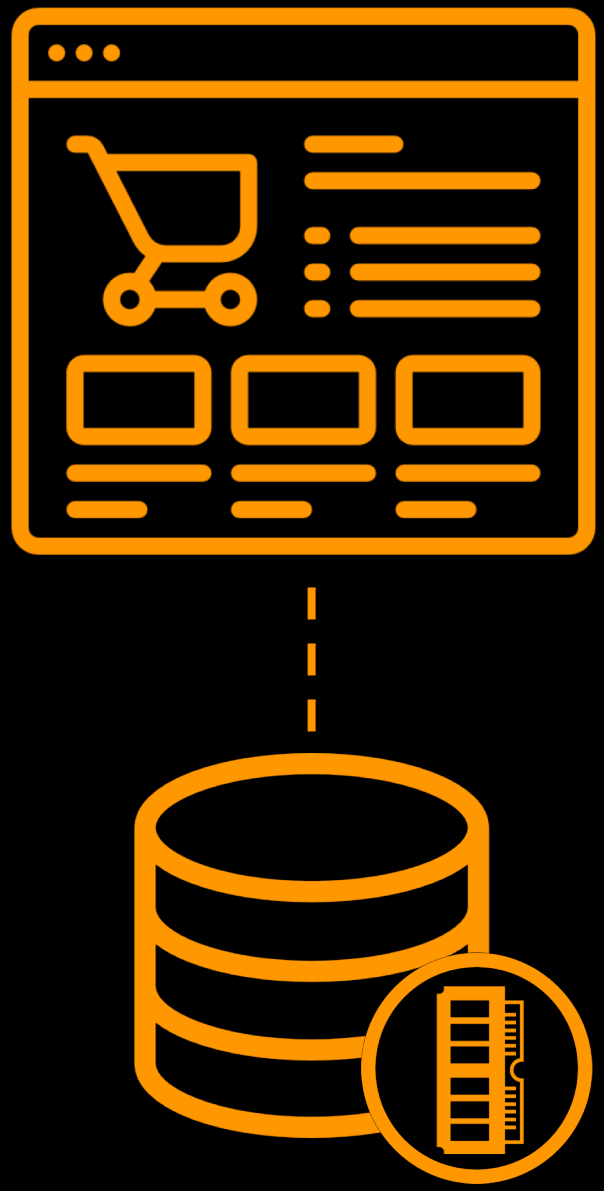
Lin Ma

Rui Shen

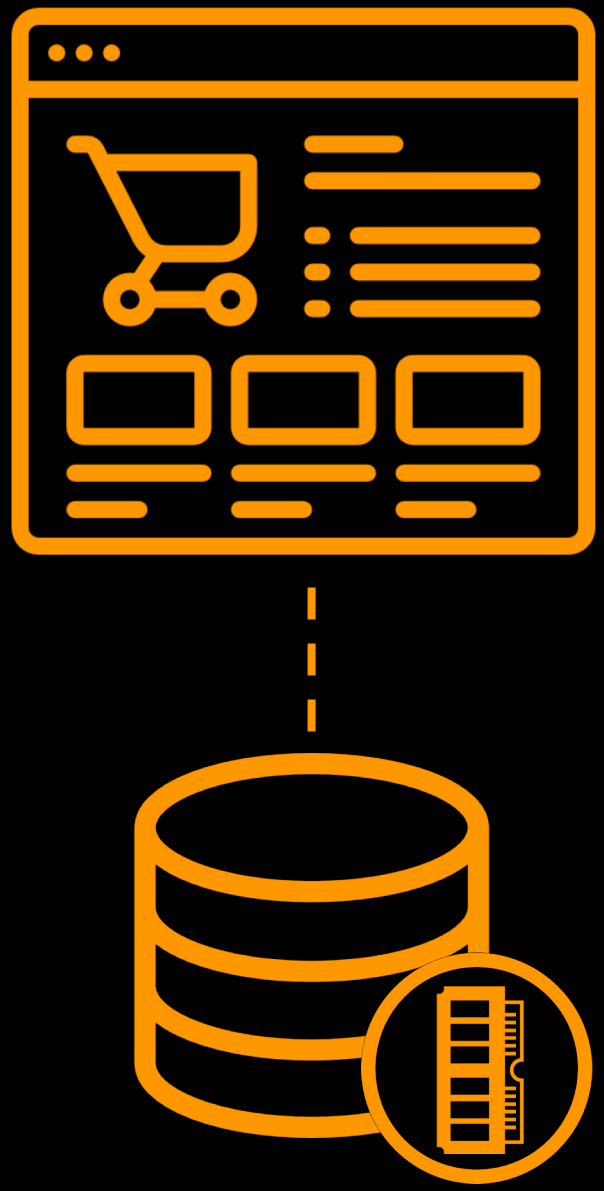




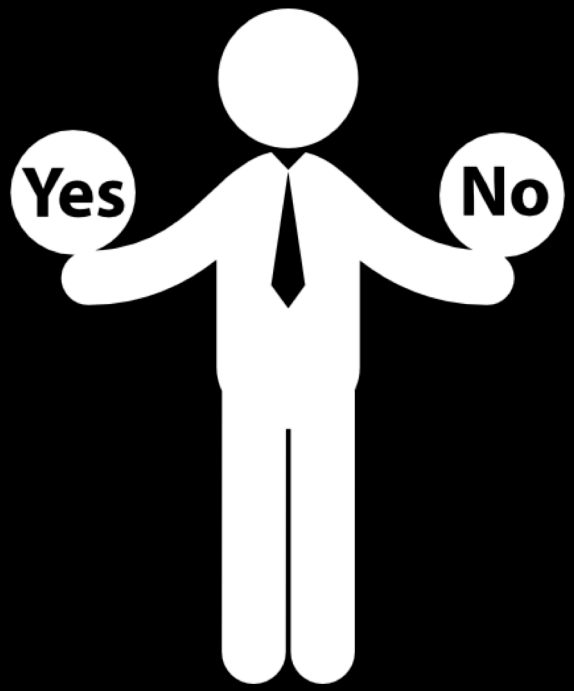
You are running out of memory



You are running out of memory

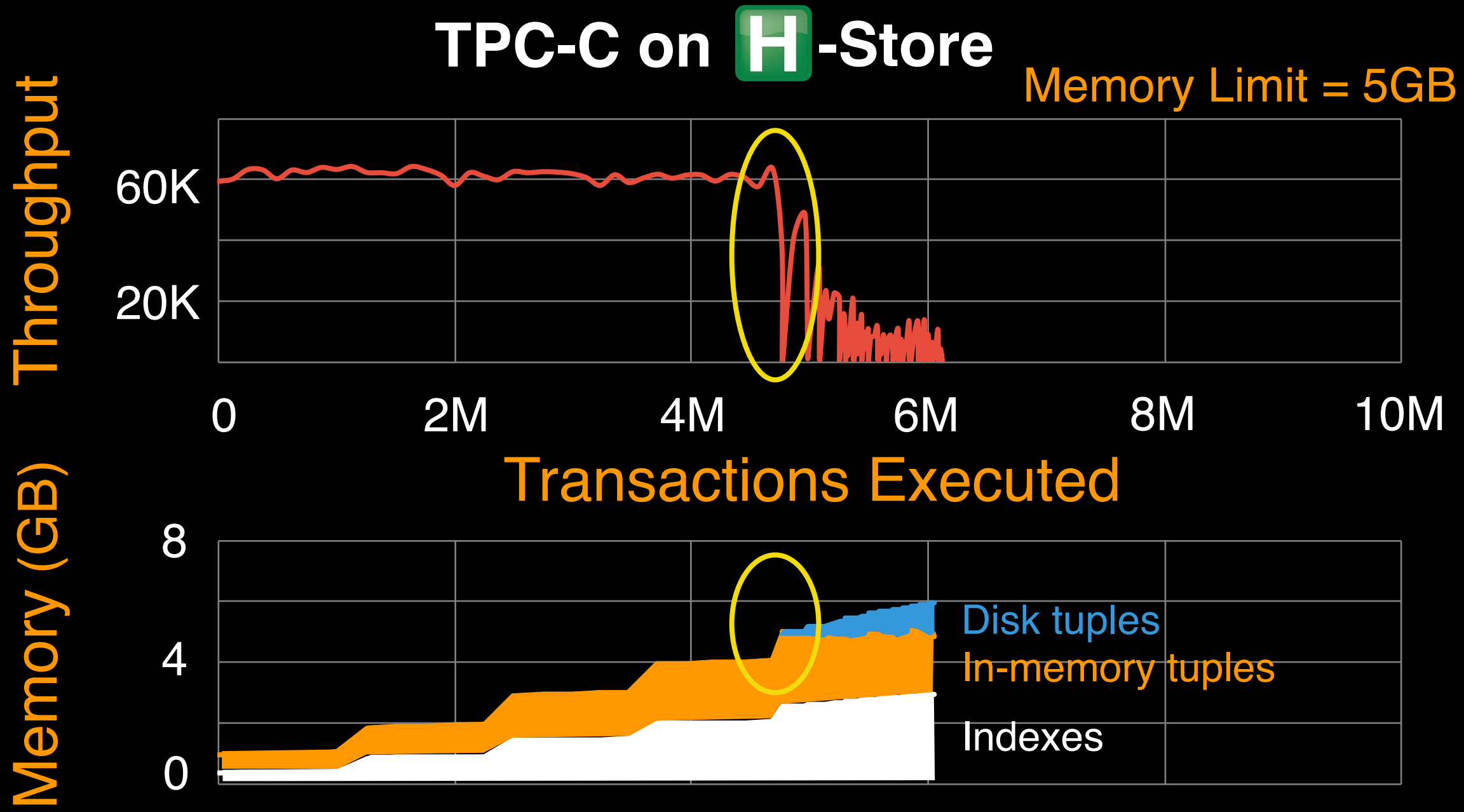


Buy more  ?



You are running out of memory

TPC-C on H-Store




I GOT STUCK
SO I WENT TO SLEEP
zipmeme

The better way:

Use memory more efficiently

Indexes are LARGE

Benchmark	% space for index		Hybrid Index
TPC-C	58%	→	34%
Voter	55%	→	41%
Articles	34%	→	18%

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)

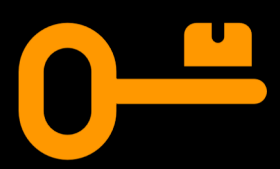
Performance ≈

Space 30 – 70% ↓

Did we solve this problem?

Stay tuned

How do hybrid indexes achieve memory savings ?



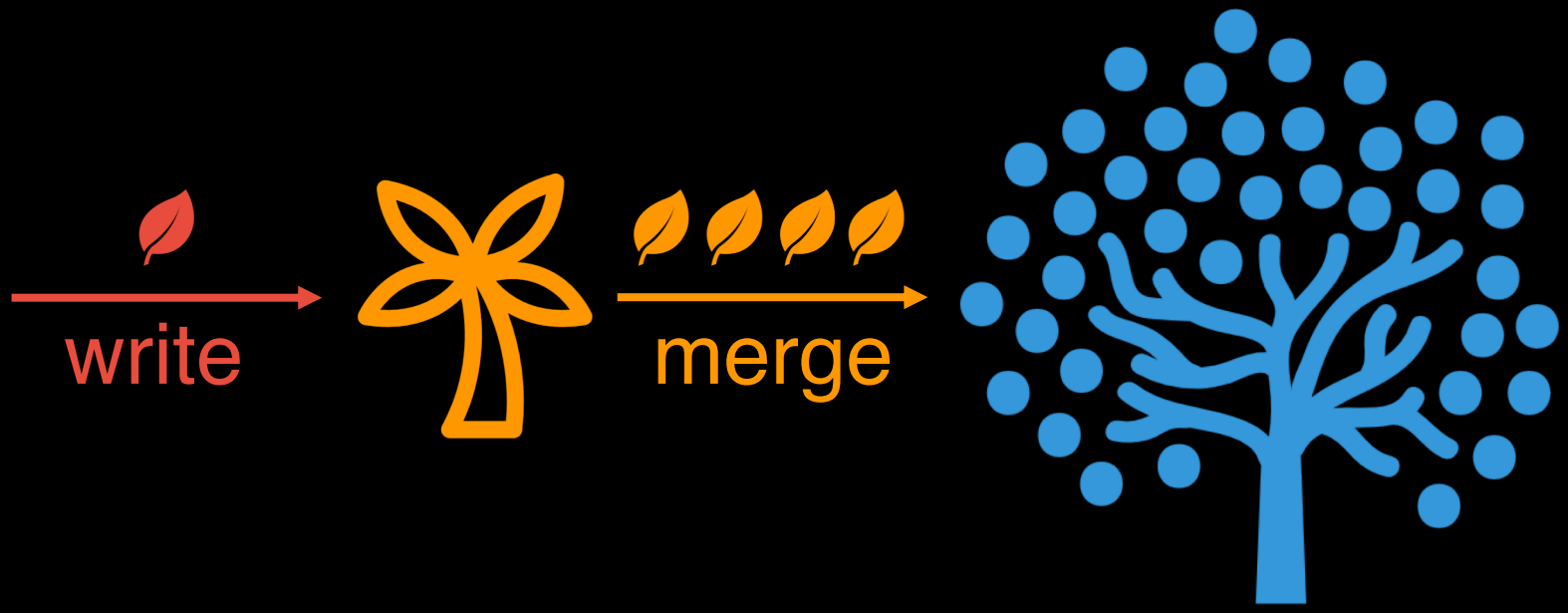
Static

Hybrid Index: a dual-stage architecture

dynamic stage

static stage

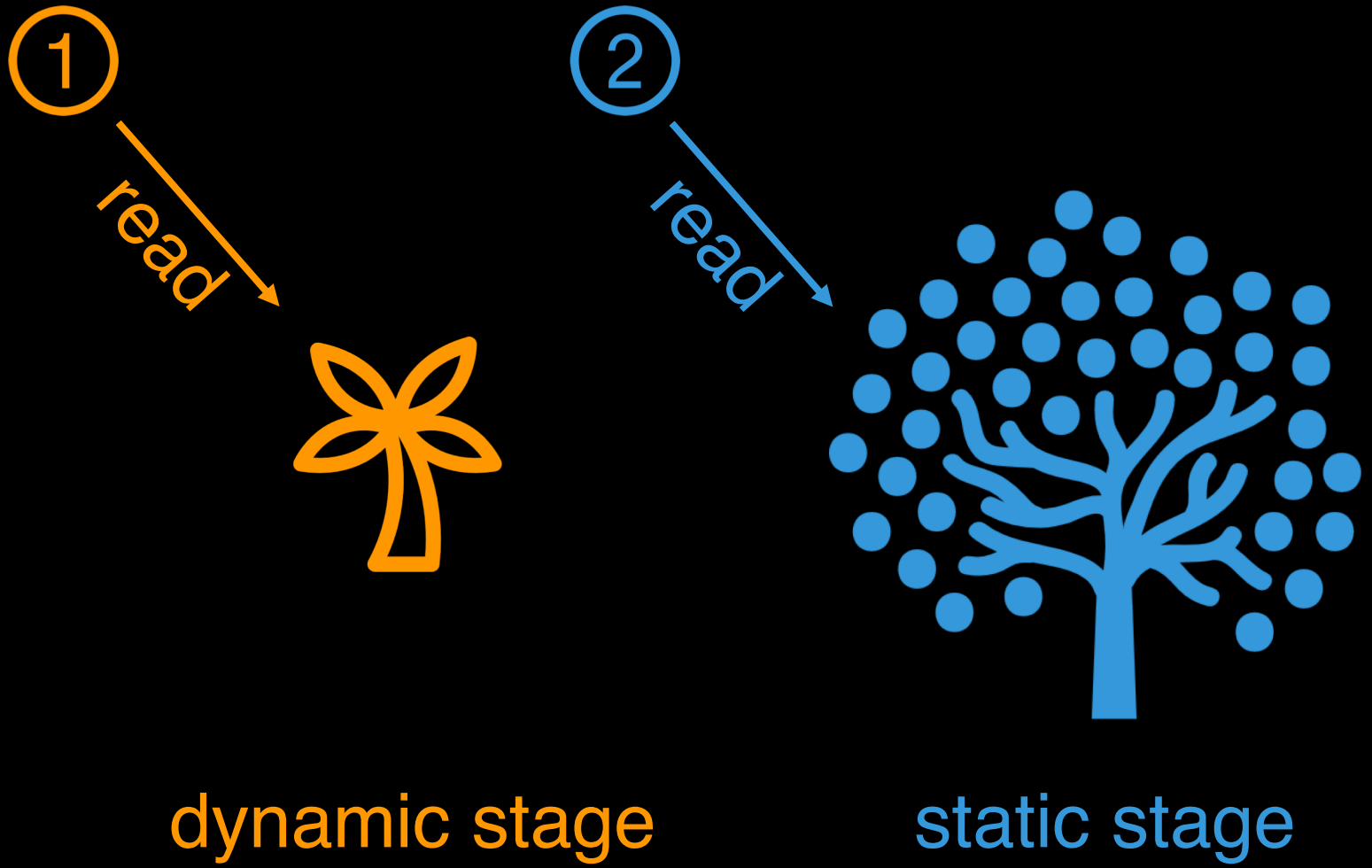
Inserts are batched in the dynamic stage



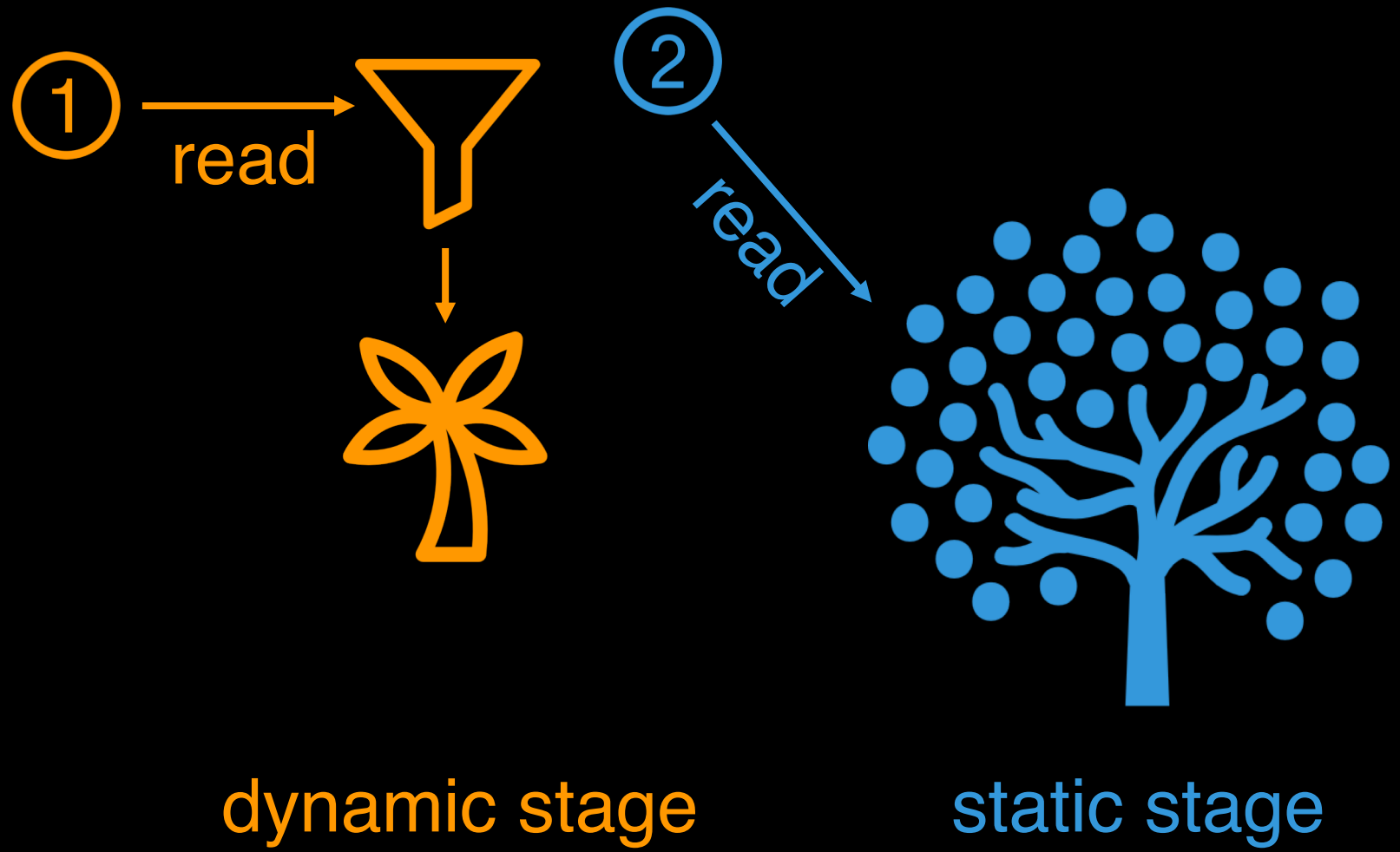
dynamic stage

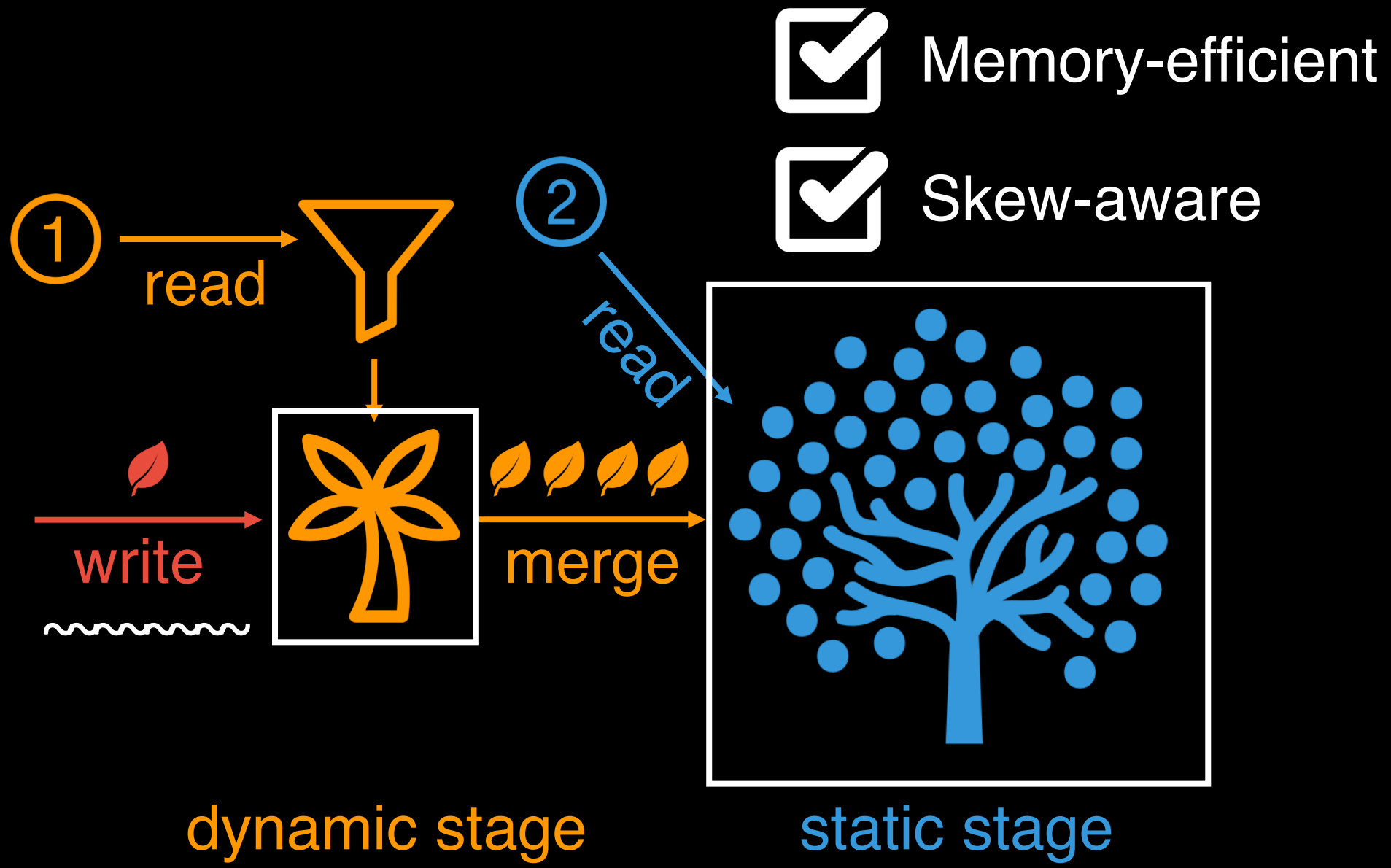
static stage

Reads search the stages in order

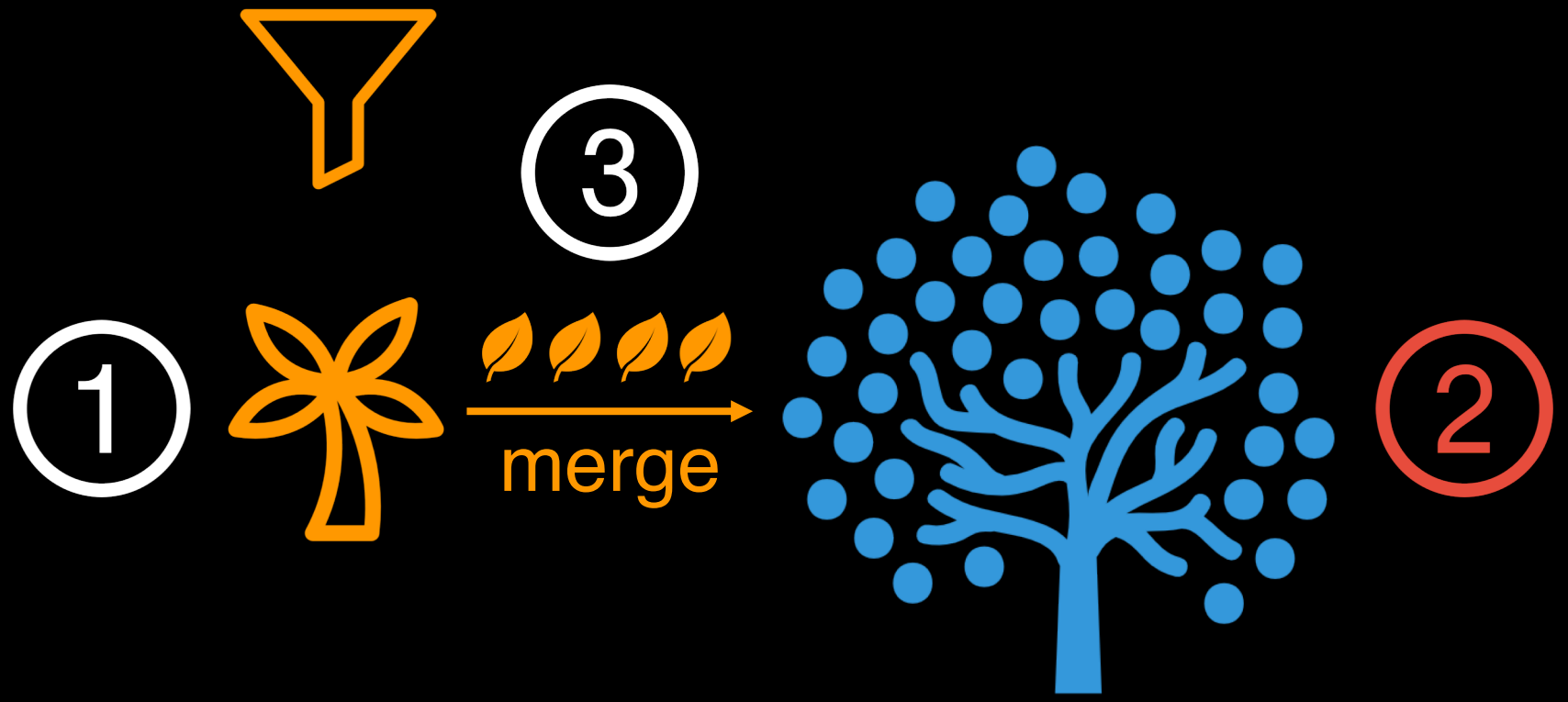


A Bloom filter improves read performance




Memory-efficient
Skew-aware
1
read
2
read
write
merge
dynamic stage
static stage

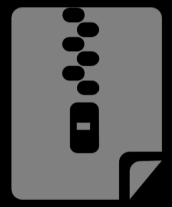
The Dual-Stage Transformation



The Dynamic-to-Static Rules

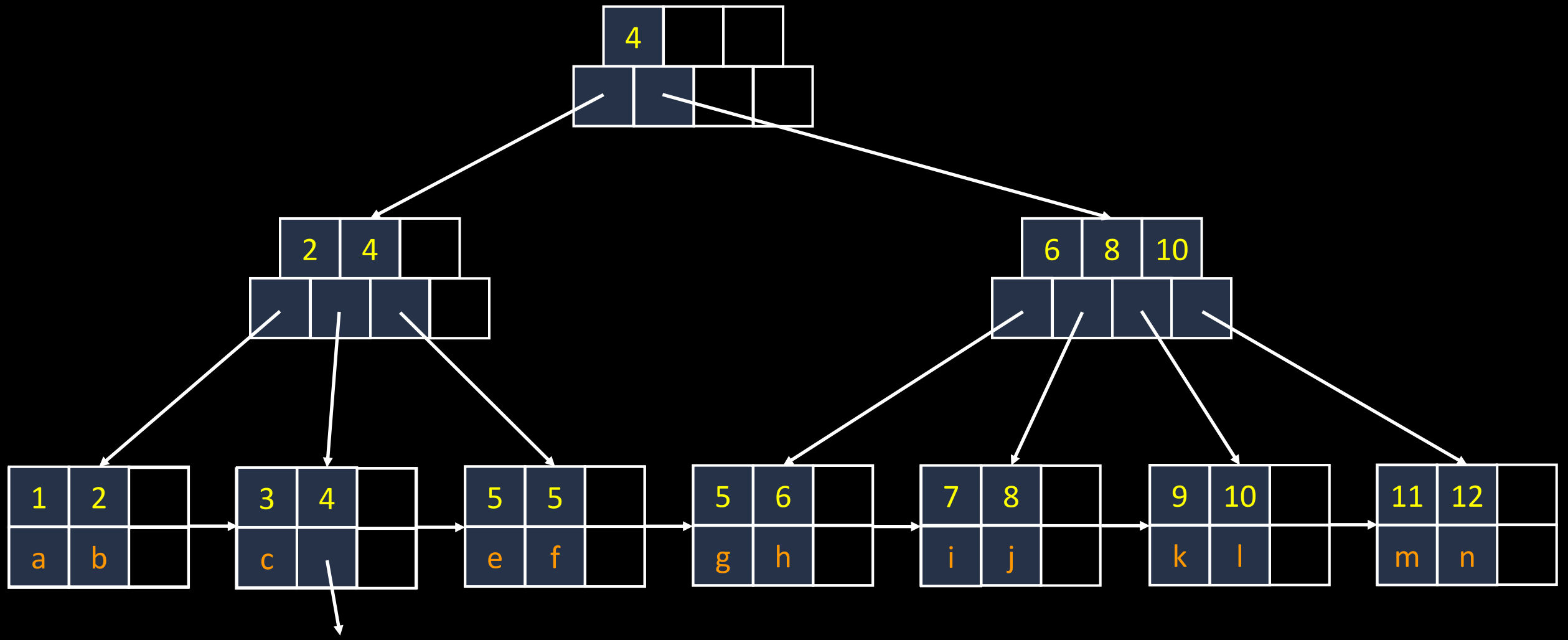
Compaction

Reduction

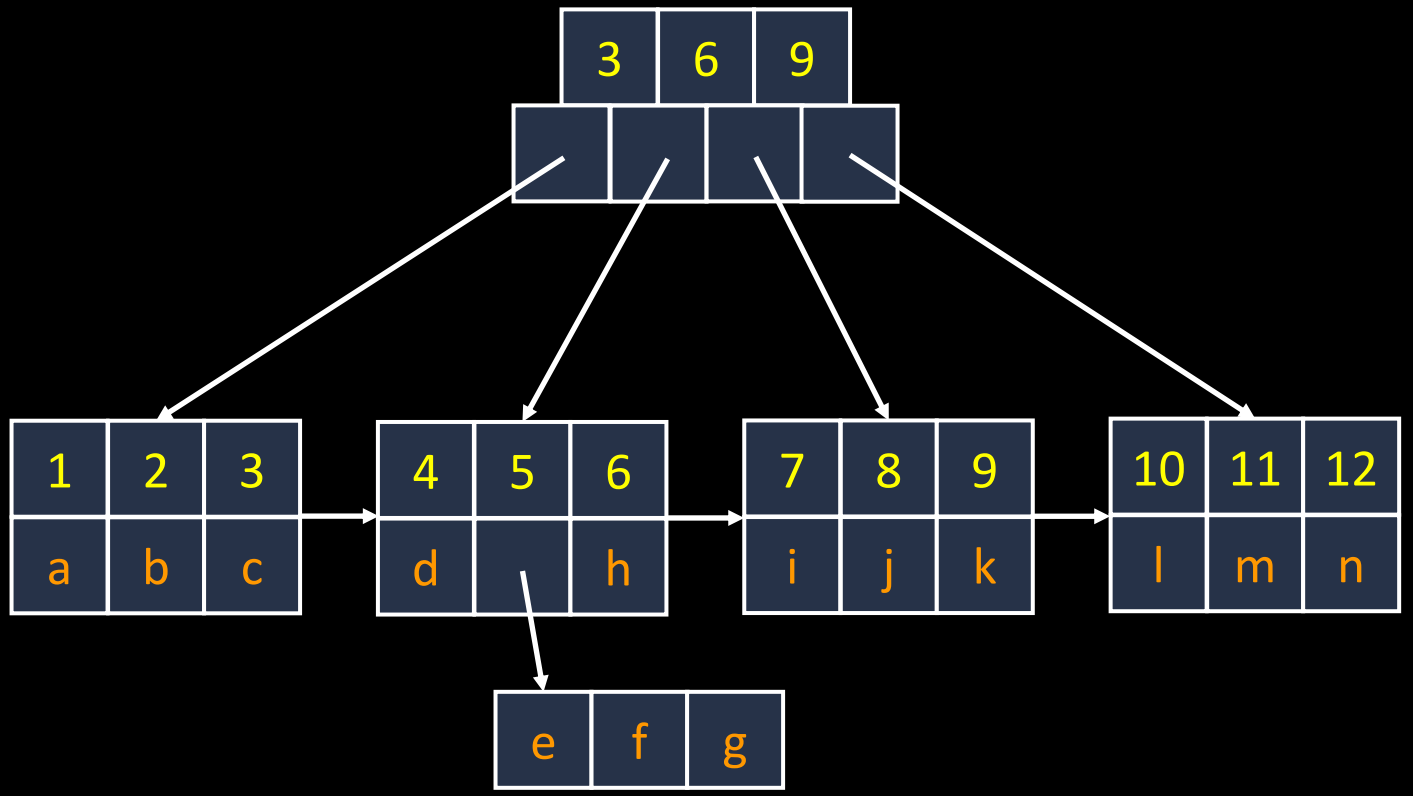


Compression

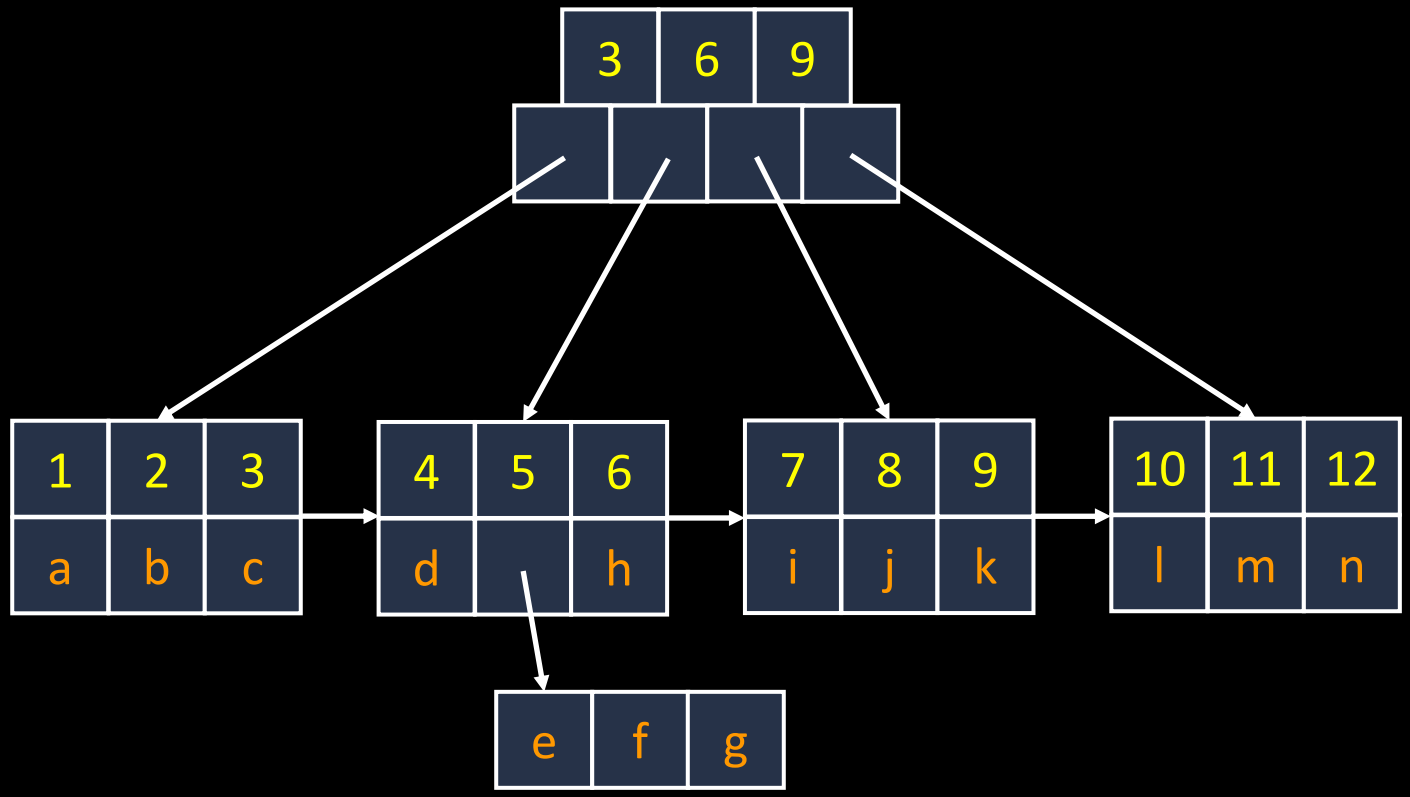
Compaction: minimize # of memory blocks



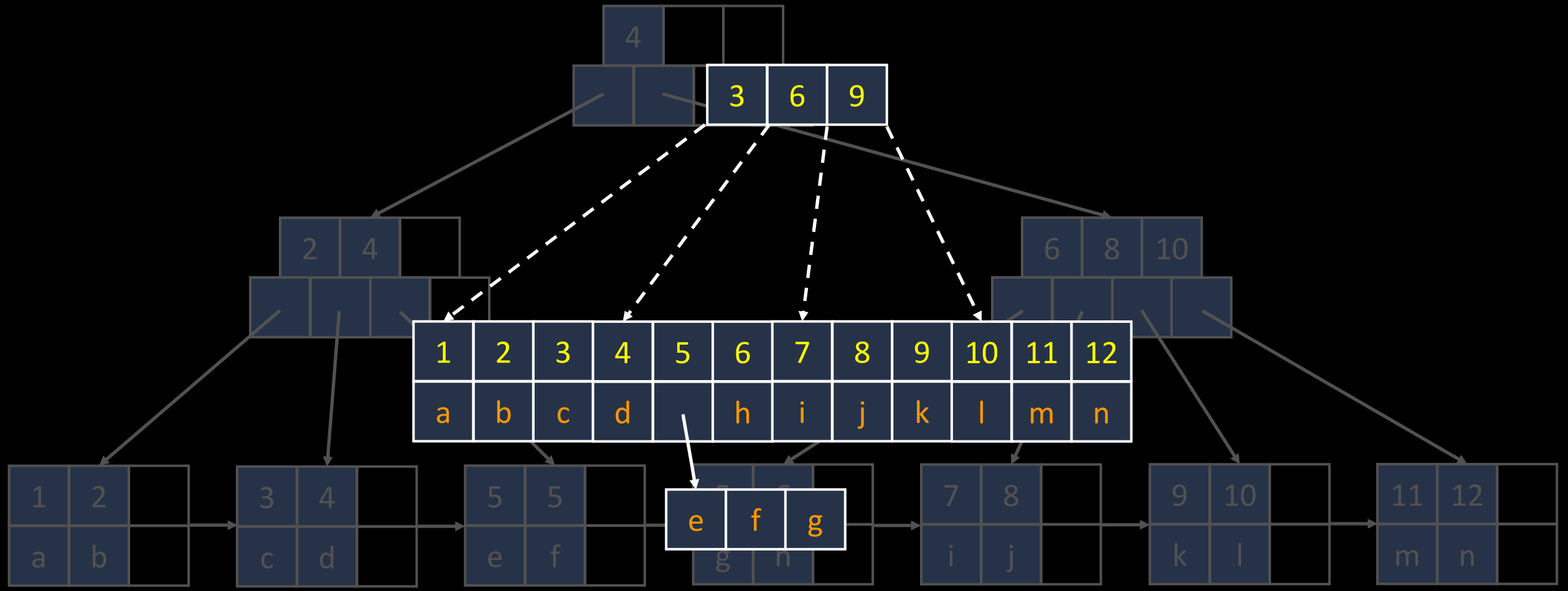
Compaction: minimize # of memory blocks



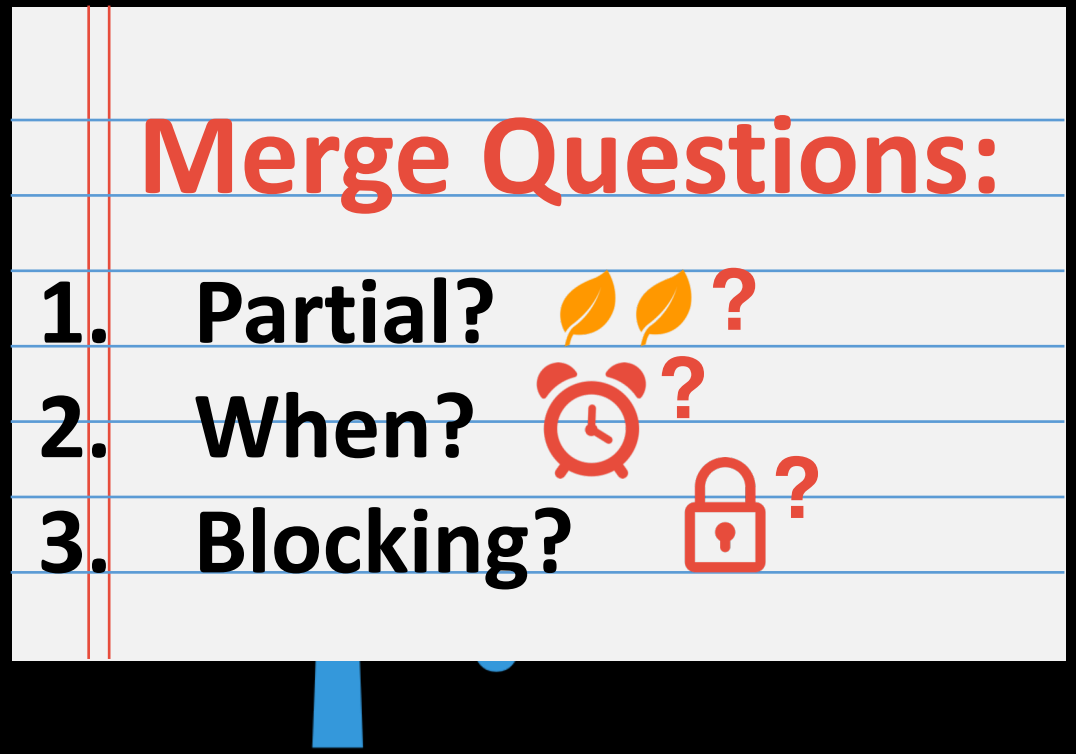
Reduction: minimize structural overhead



Reduction: minimize structural overhead



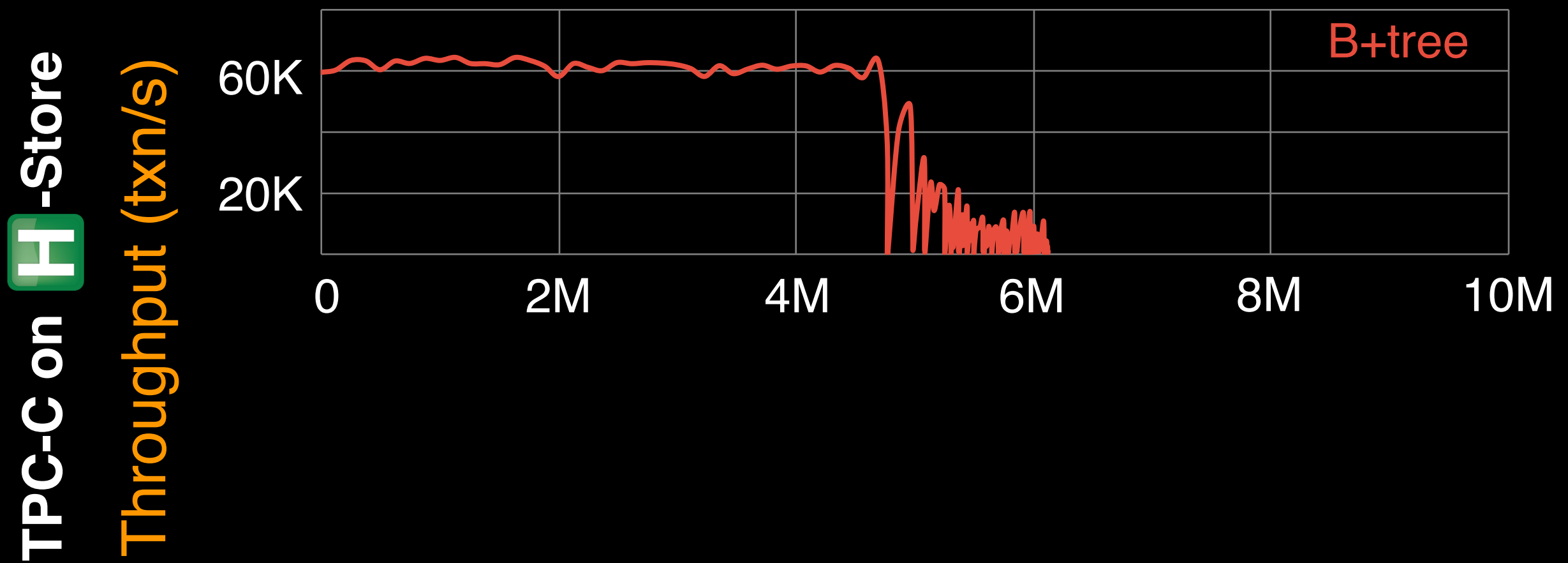
The Dual-Stage Transformation



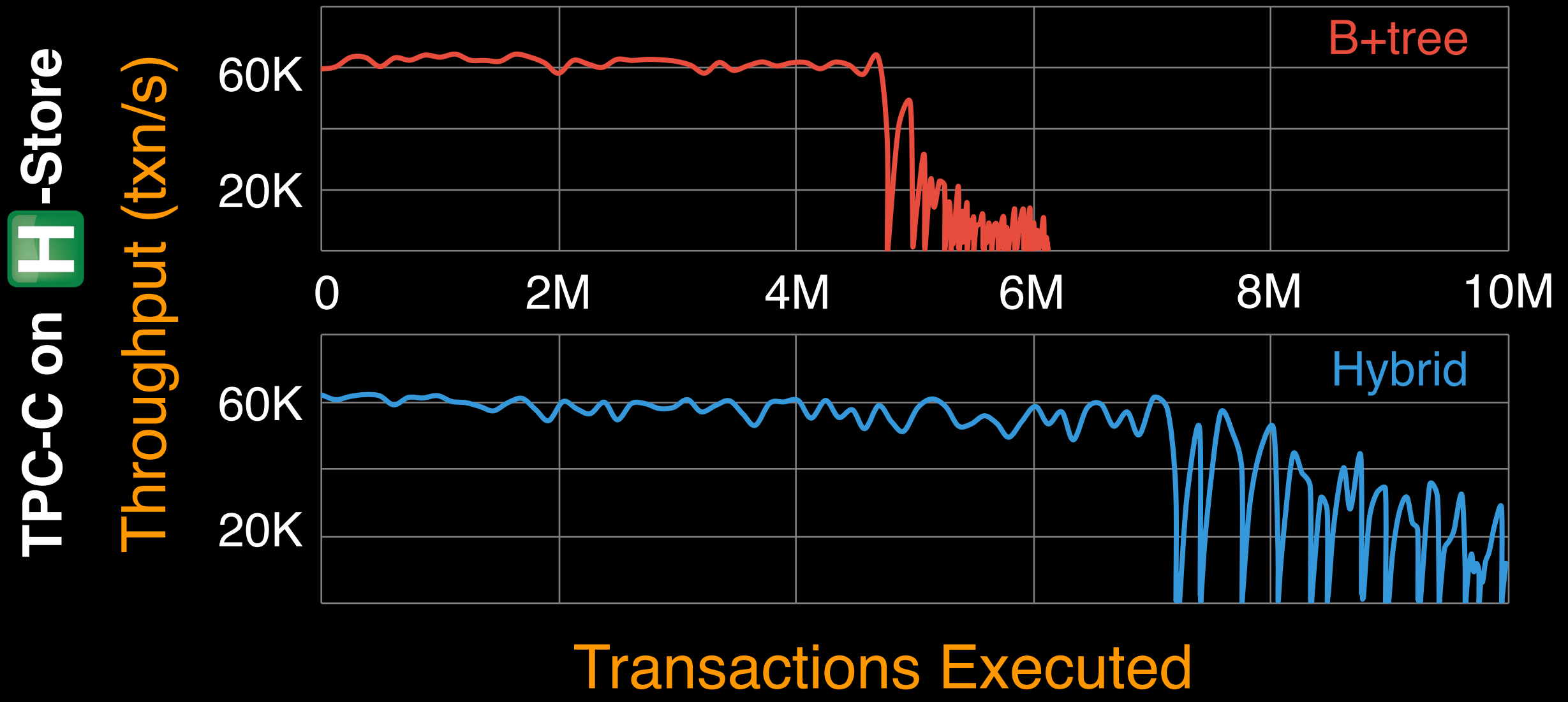
dynamic stage

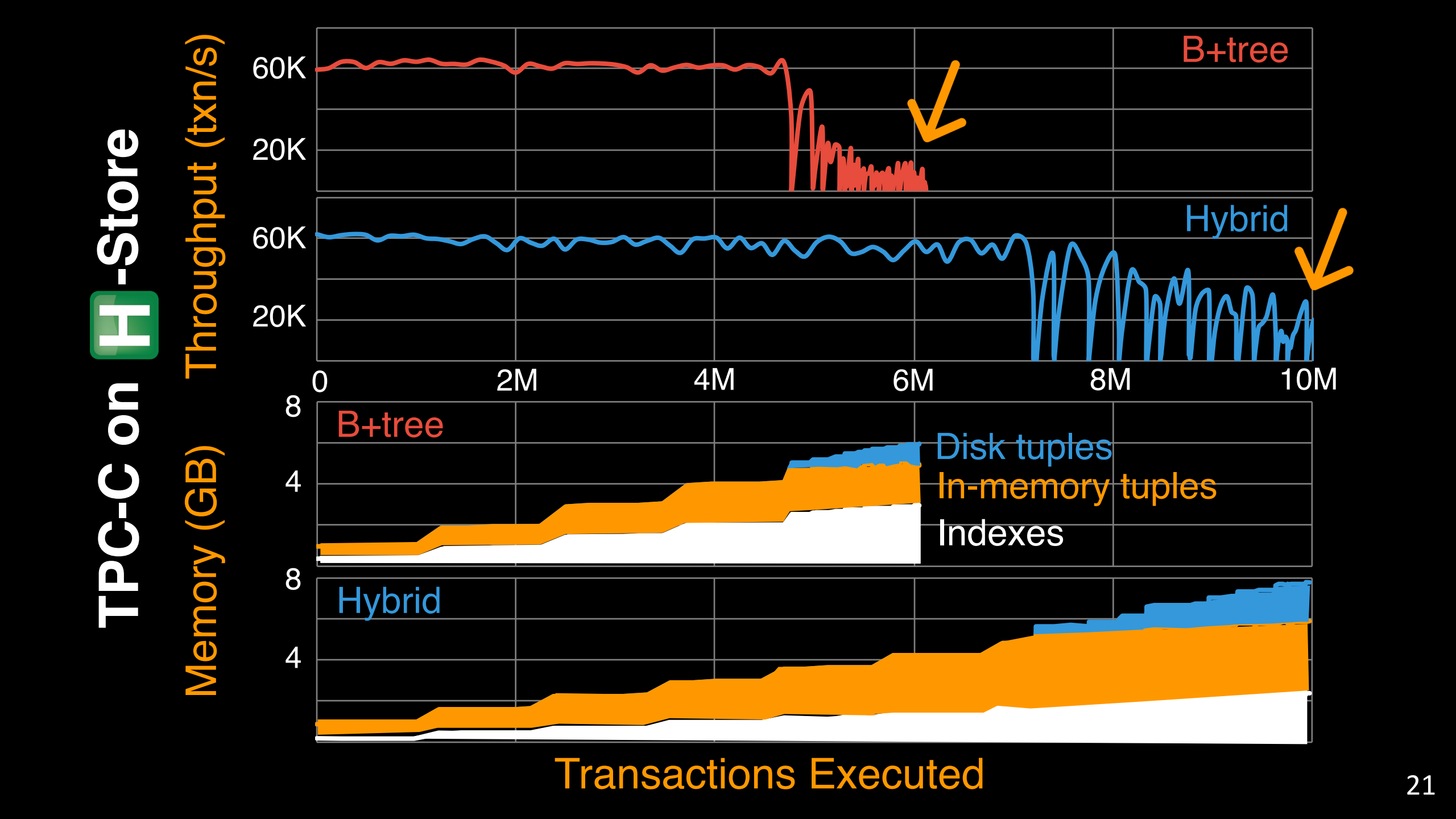
static stage

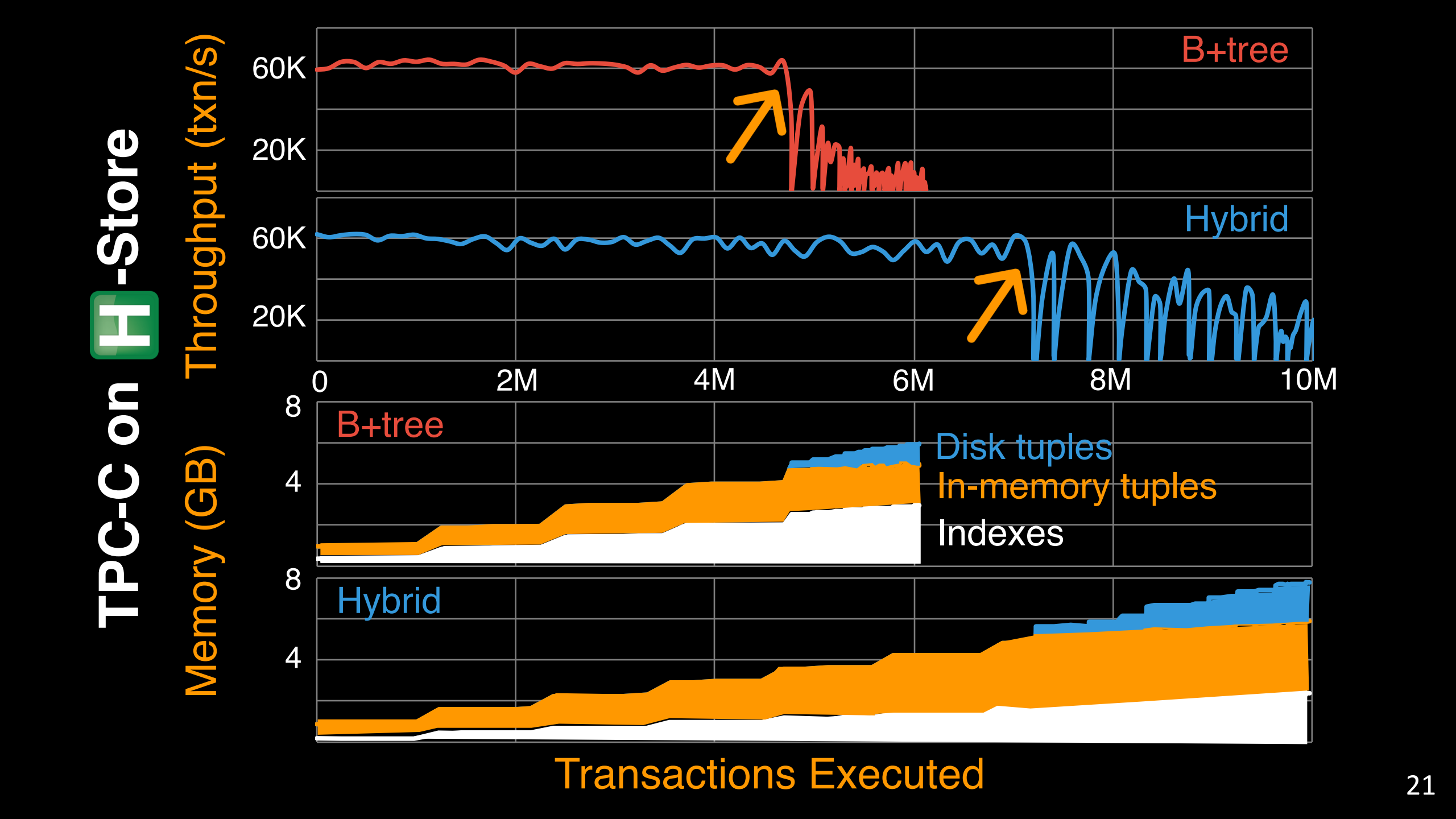
Did we solve this problem?

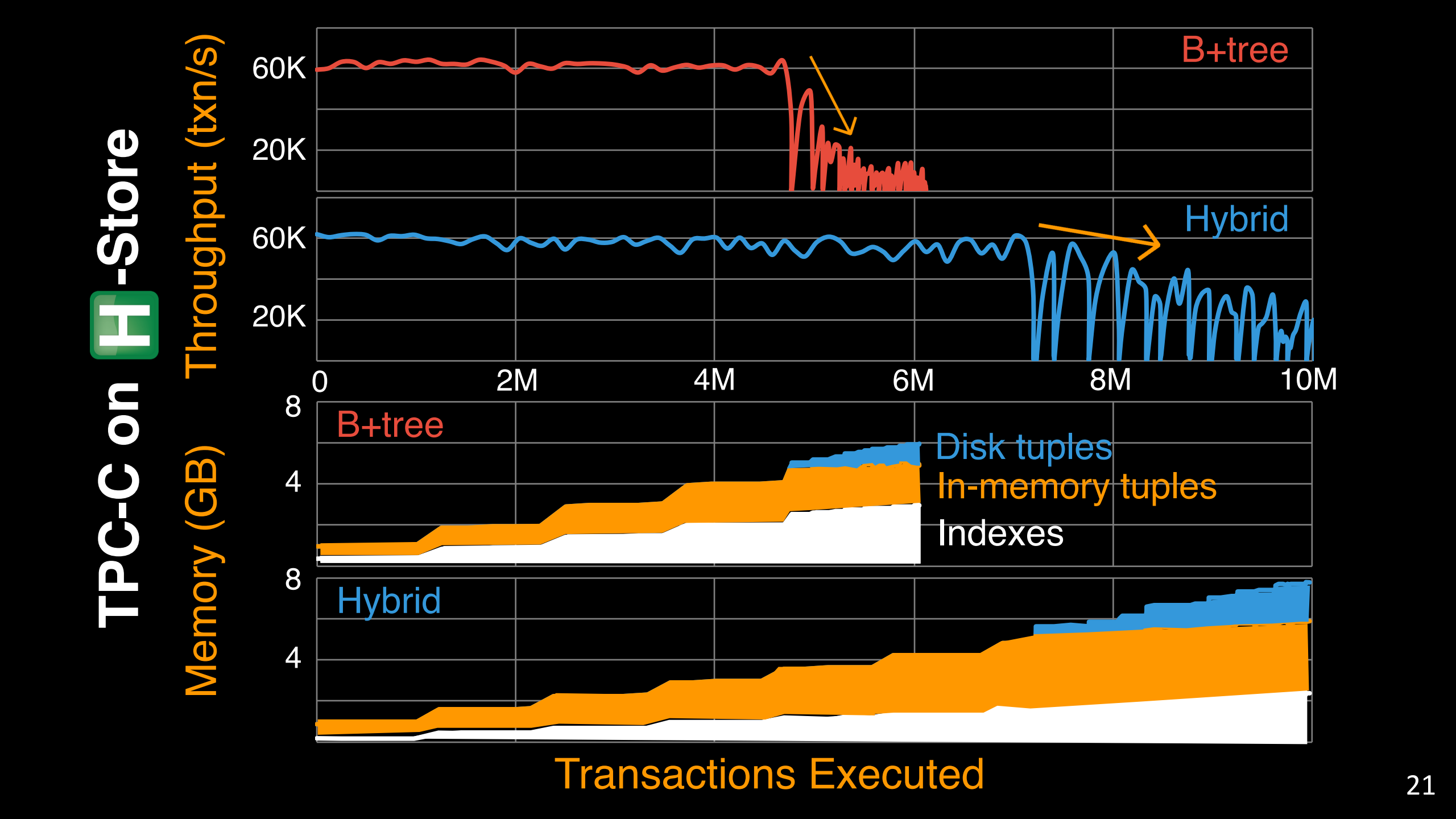


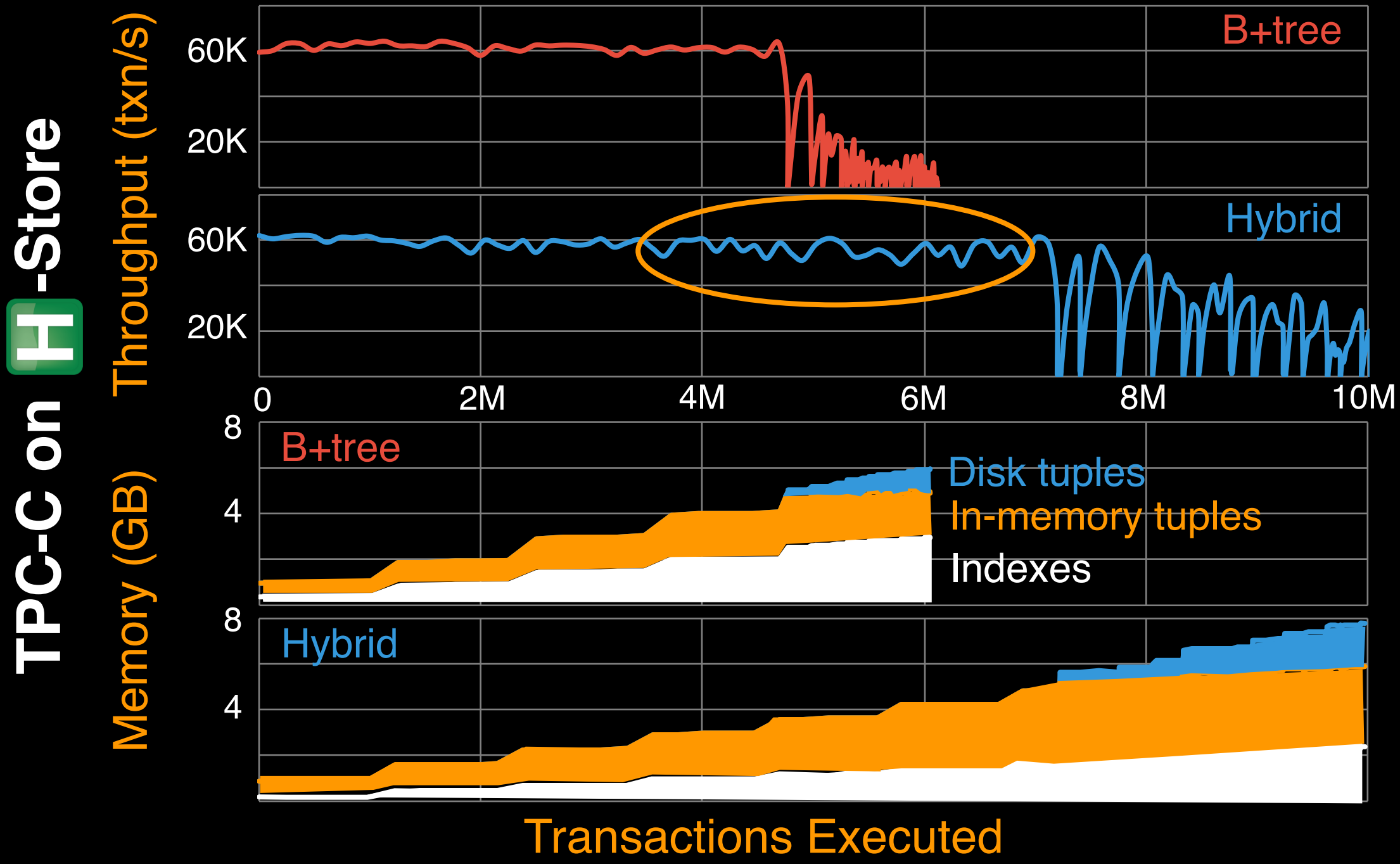
Yes, we improved the DBMS's capacity!




TPC-C on H-Store
Throughput (txn/s)
60K
20K
B+tree
60K
20K
Hybrid
0
2M
4M
6M
8M
10M
Memory (GB)
8
4
B+tree
Disk tuples
In-memory tuples
Indexes
8
4
Hybrid
Transactions Executed

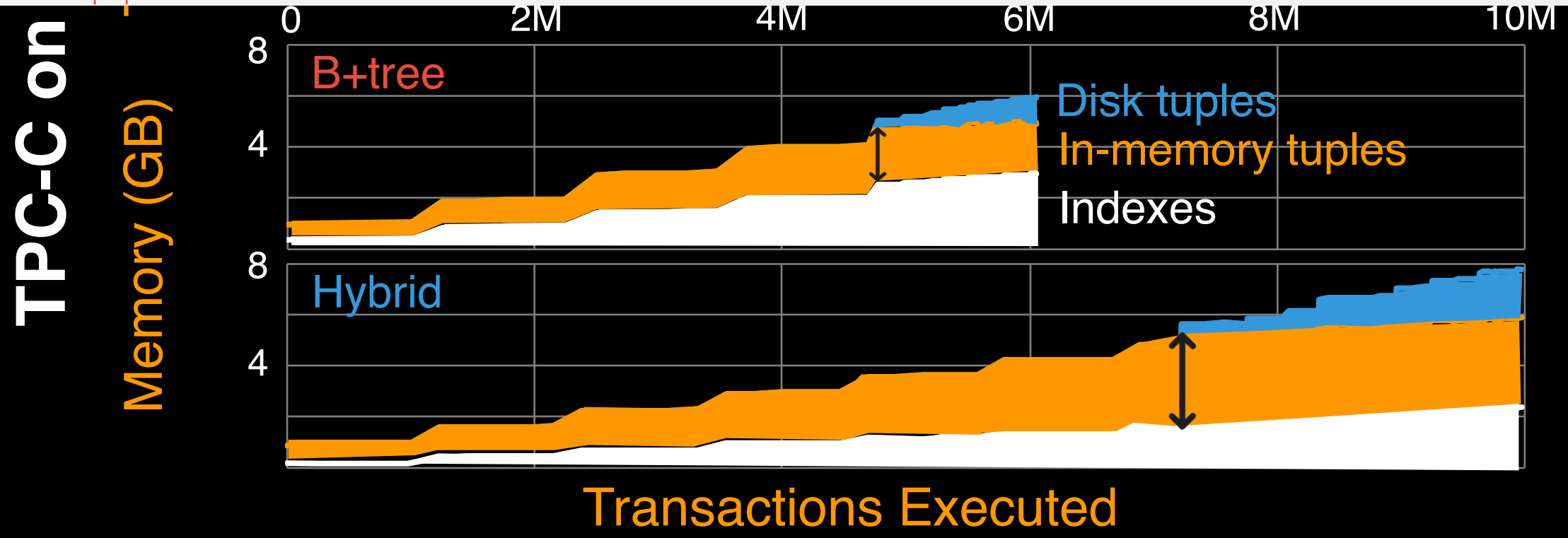

TPC-C on H-Store
Throughput (txn/s)
60K
20K
B+tree
60K
20K
Hybrid
0
2M
4M
6M
8M
10M
Memory (GB)
8
4
B+tree
Disk tuples
In-memory tuples
Indexes
8
4
Hybrid
Transactions Executed


TPC-C on H-Store
Throughput (txn/s)
60K
20K
B+tree
60K
20K
Hybrid
0
2M
4M
6M
8M
10M
Memory (GB)
8
4
B+tree
Disk tuples
In-memory tuples
Indexes
8
4
Hybrid
Transactions Executed


TPC-C on H-Store
Throughput (txn/s)
60K
20K
B+tree
60K
20K
Hybrid
0
2M
4M
6M
8M
10M
Memory (GB)
8
4
B+tree
Disk tuples
In-memory tuples
Indexes
8
4
Hybrid
Transactions Executed

Take Away:

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



This is just the BEGINNING

Conclusions

① The hybrid index architecture **GENERAL**

② The Dual-Stage Transformation **PRACTICAL**

③ Applied to 4 index structures **USEFUL**

- B+tree
- Masstree
- Skip List
- Adaptive Radix Tree (ART)

Toll-Free Hotline:

1-844-88-CMUDB