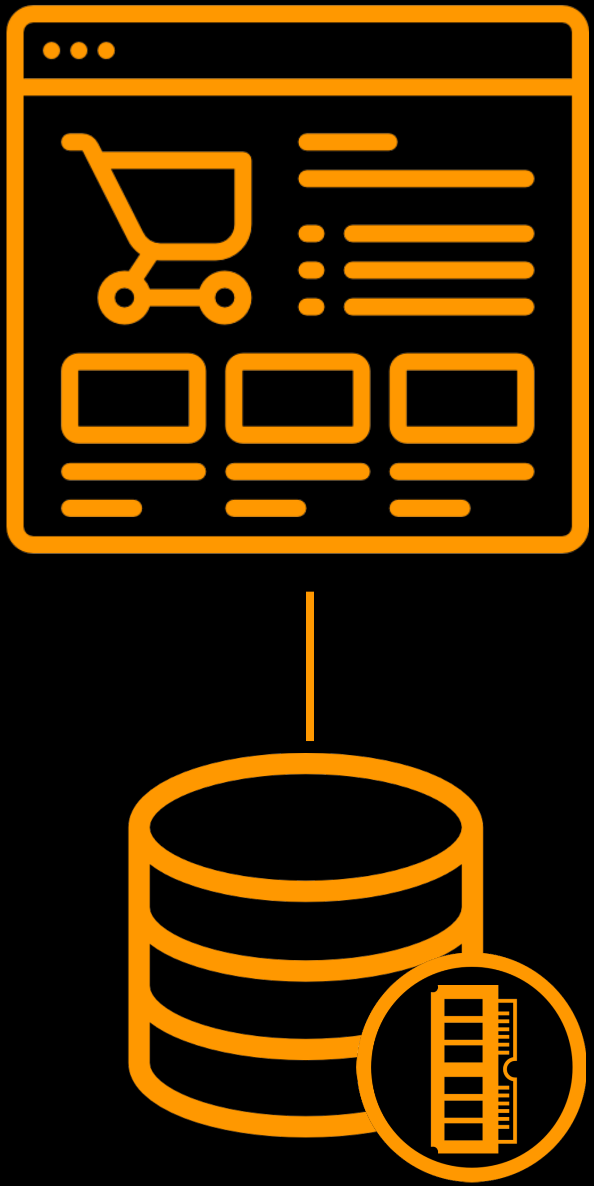
Reducing the Storage Overhead of Main-Memory OLTP Databases with Hybrid Indexes

SIGMOD'16

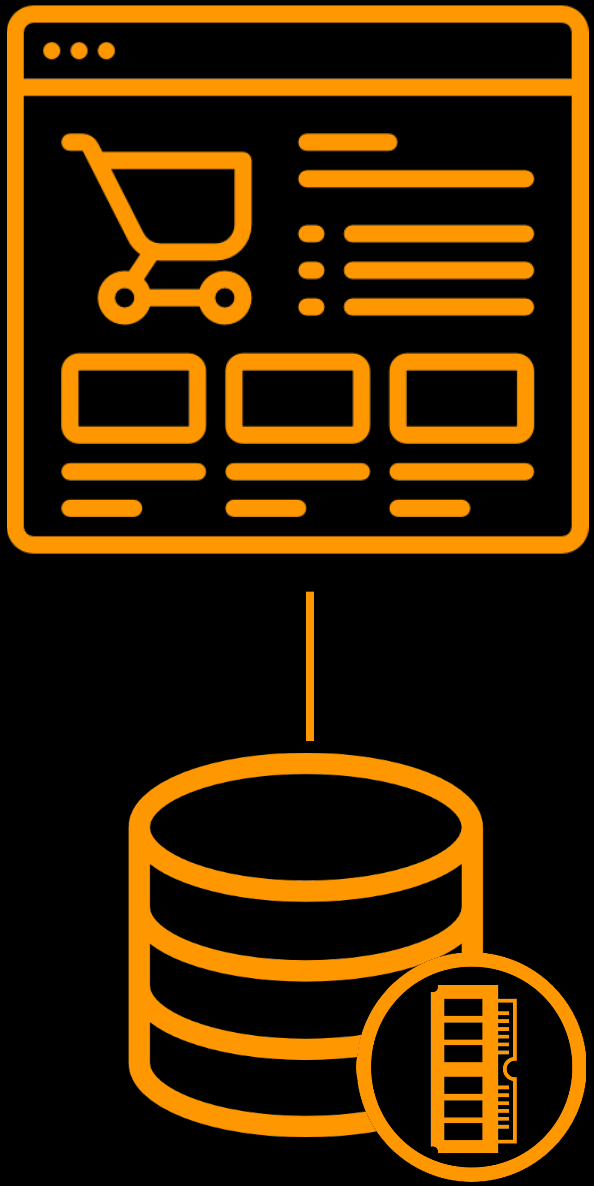
Huanchen Zhang

Co-authors:
David G. Andersen
Andrew Pavlo
Michael Kaminsky
Lin Ma
Rui Shen

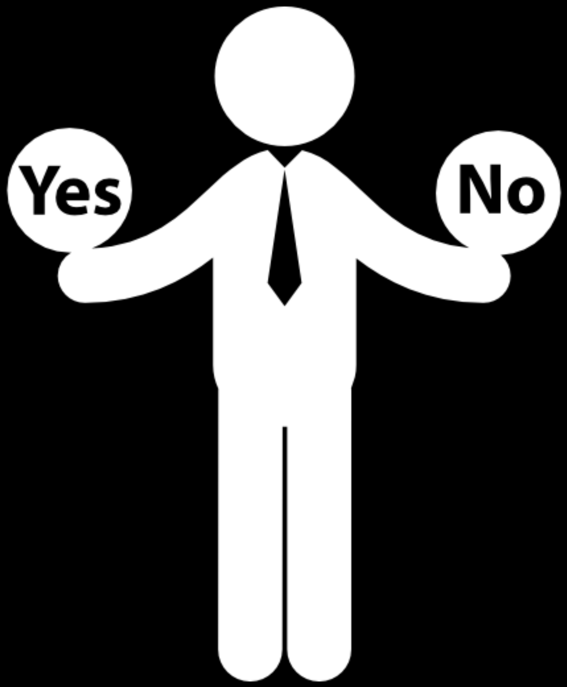
⚠ You are running out of memory



⚠ You are running out of memory

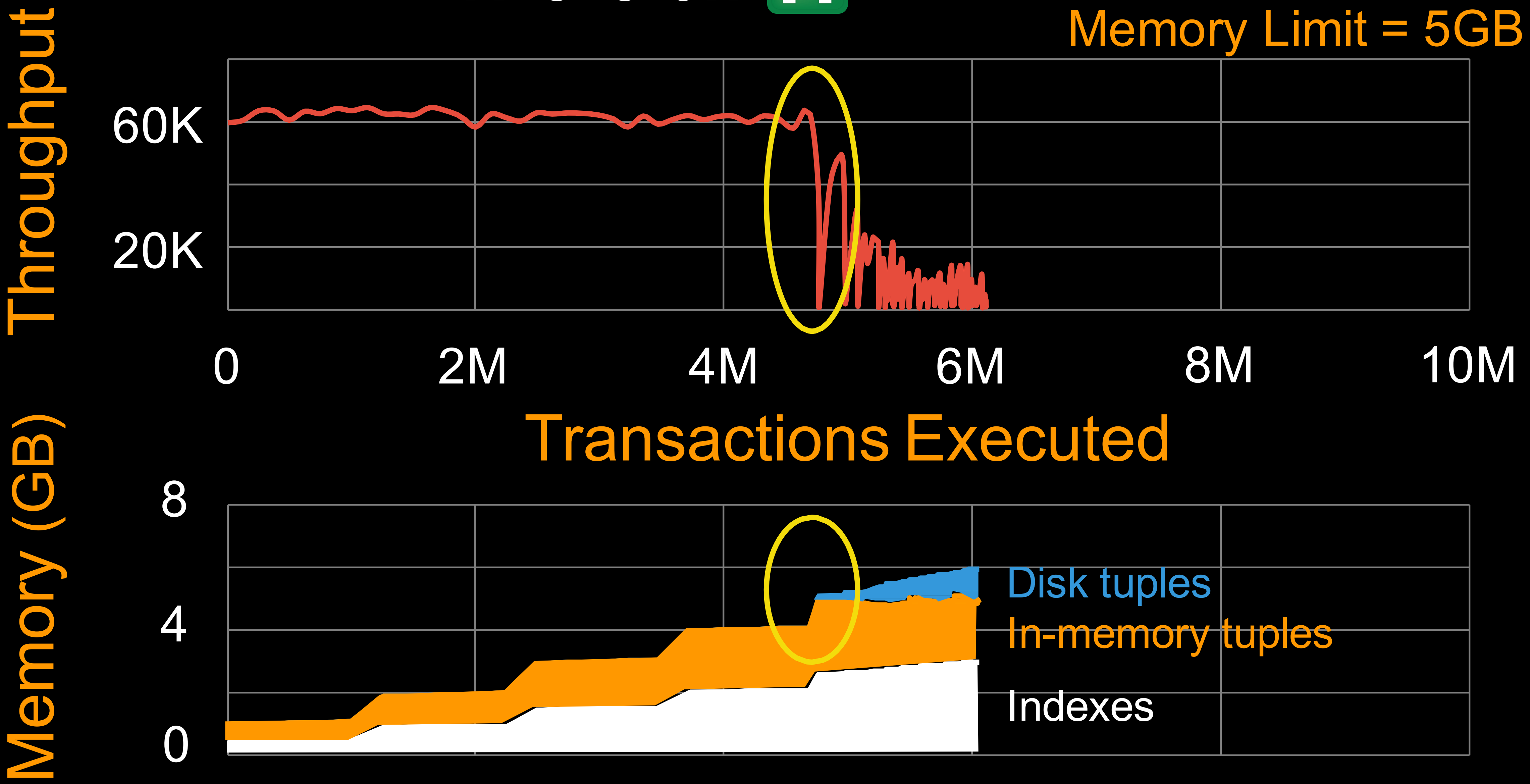


Buy more ?



⚠ You are running out of memory

TPC-C on H-Store


Memory Limit = 5GB

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)

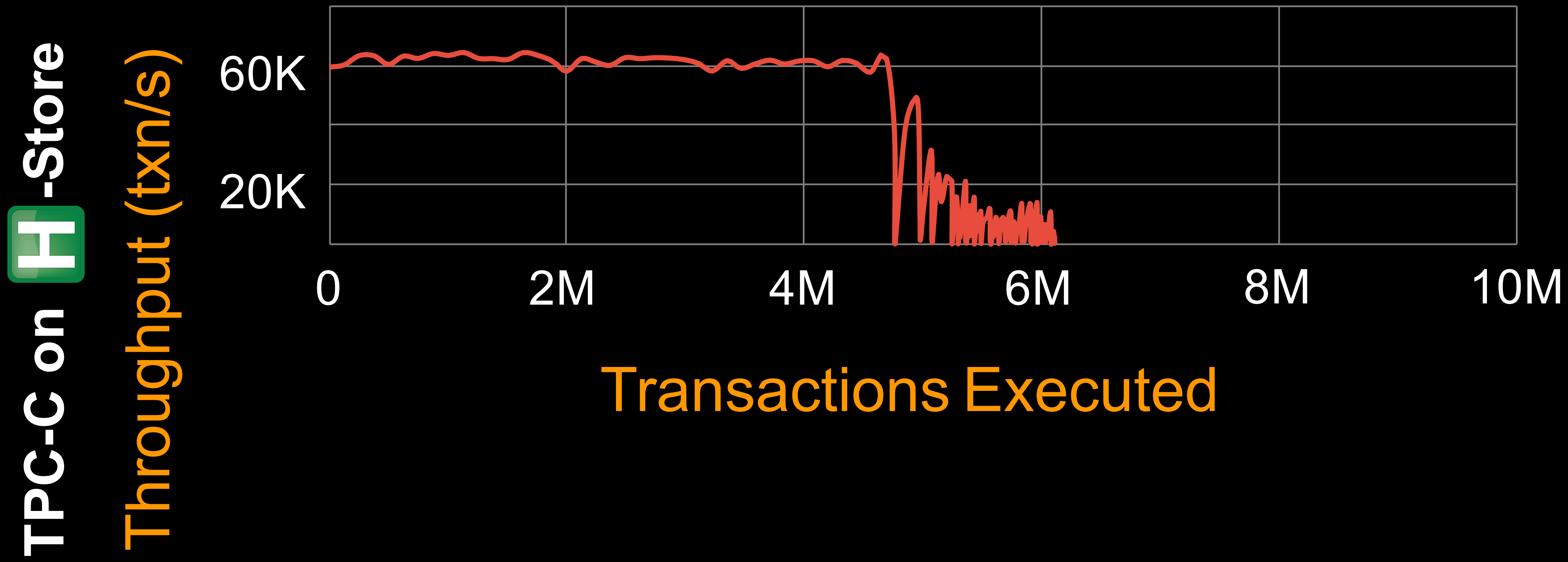
Performance

≈

Space

30 – 70% ↓

Did we solve this problem?



How do hybrid indexes achieve memory savings ?

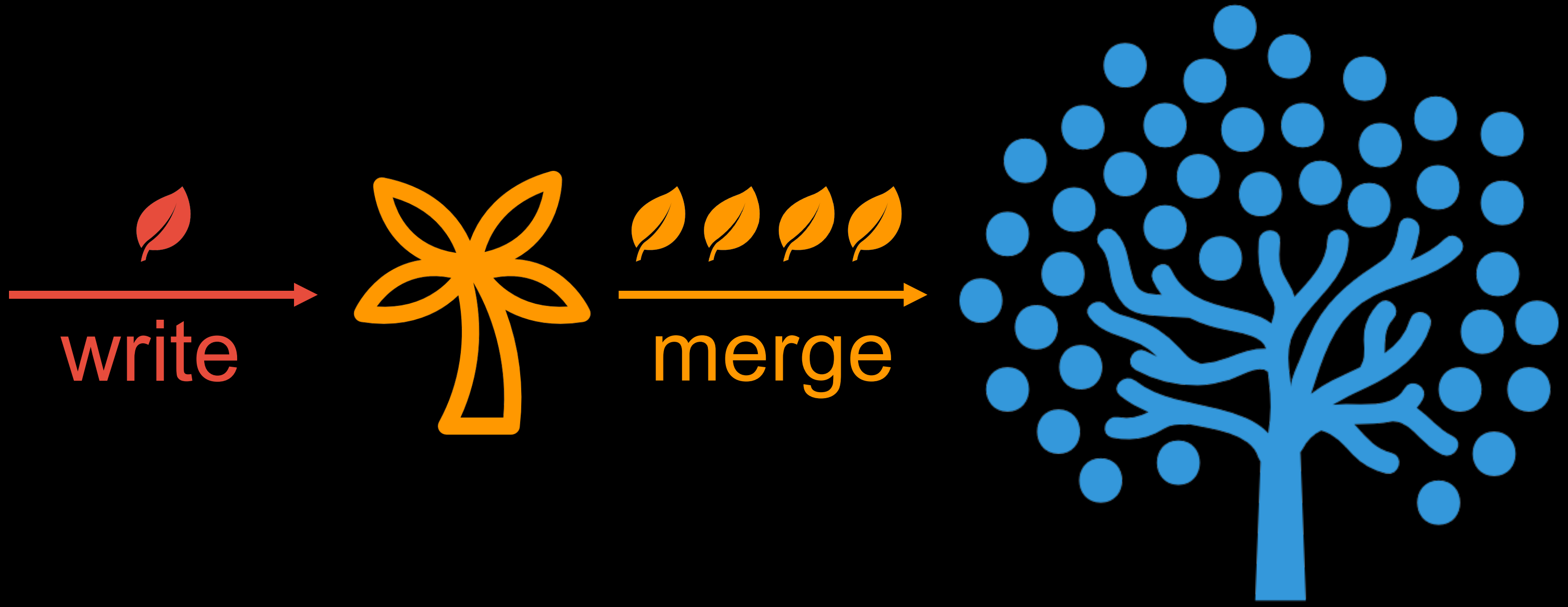
🔑 Static

Hybrid Index: a dual-stage architecture

dynamic stage

static stage

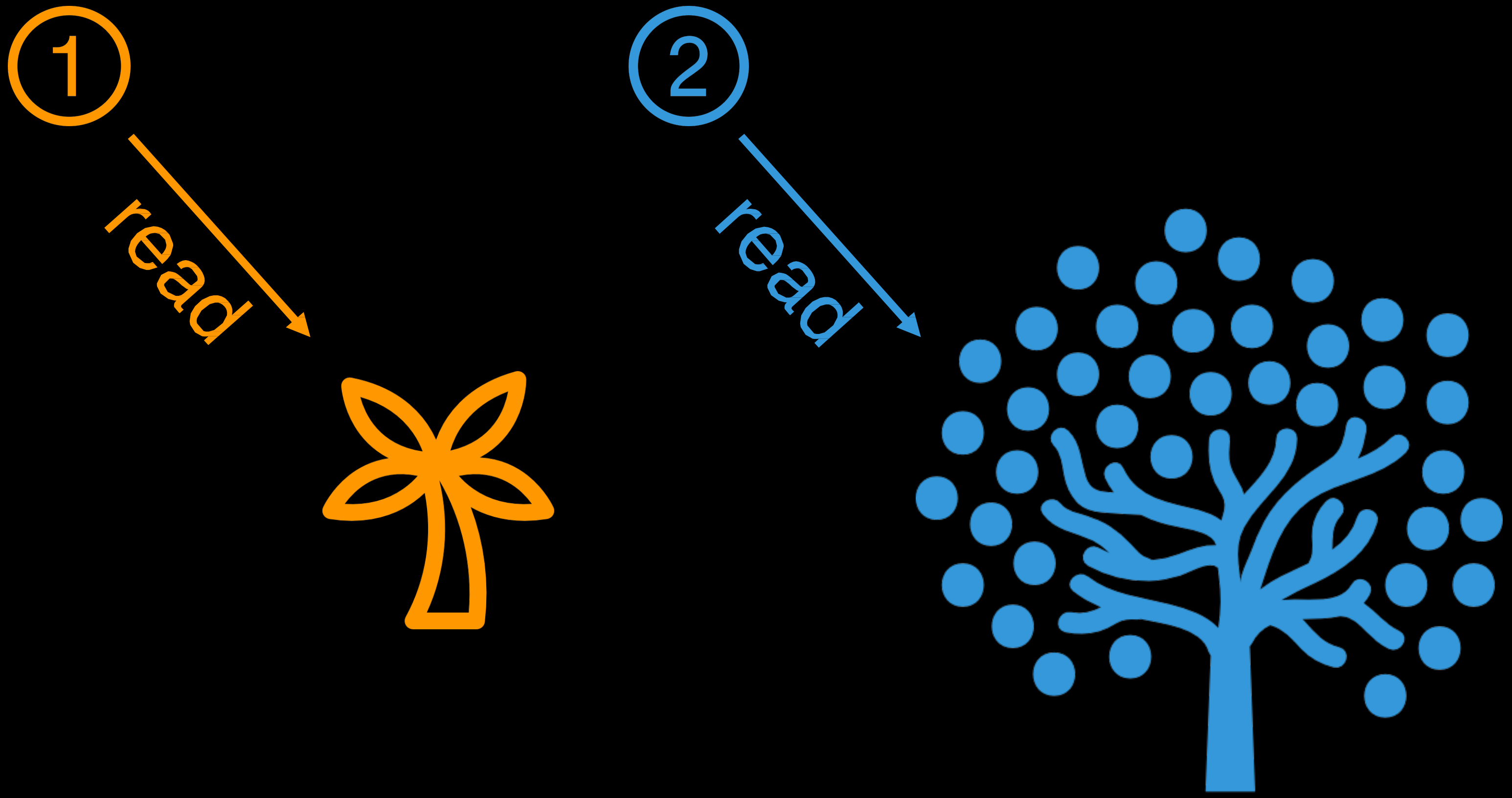
Inserts are batched in the dynamic stage



dynamic stage

static stage

Reads search the stages in order



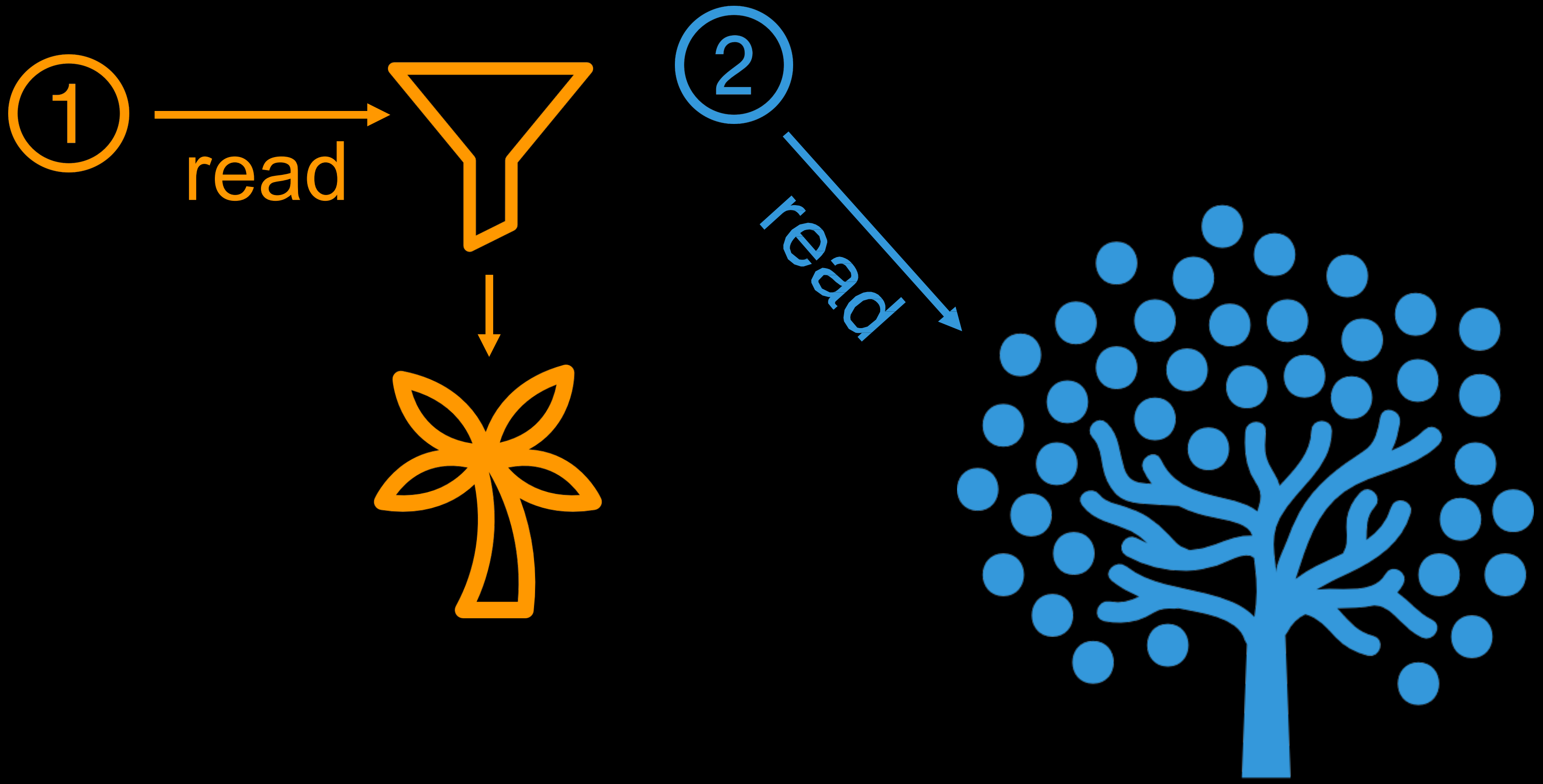
① read

② read

dynamic stage

static stage

A Bloom filter improves read performance

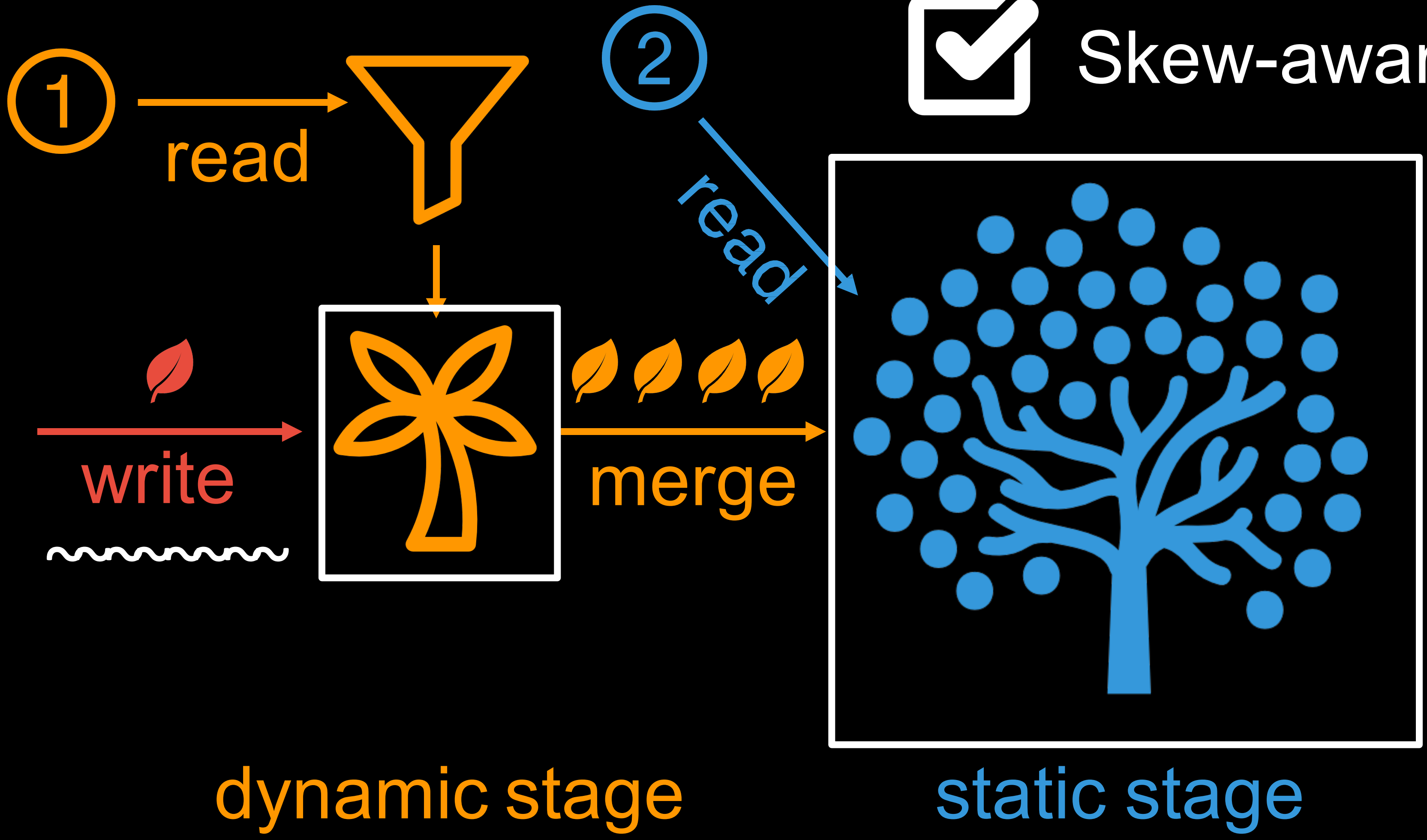


dynamic stage

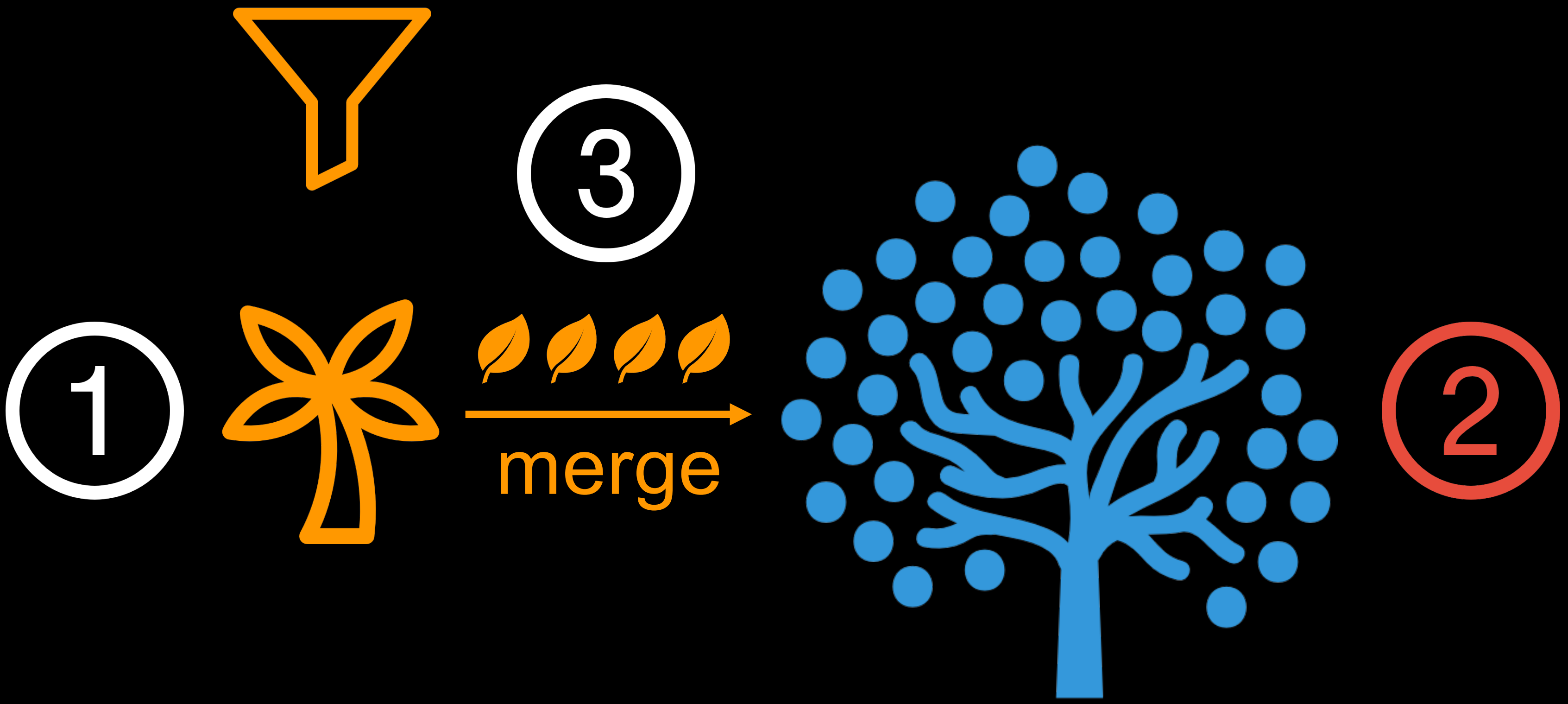
static stage

☑ Memory-efficient

☑ Skew-aware



The Dual-Stage Transformation



merge

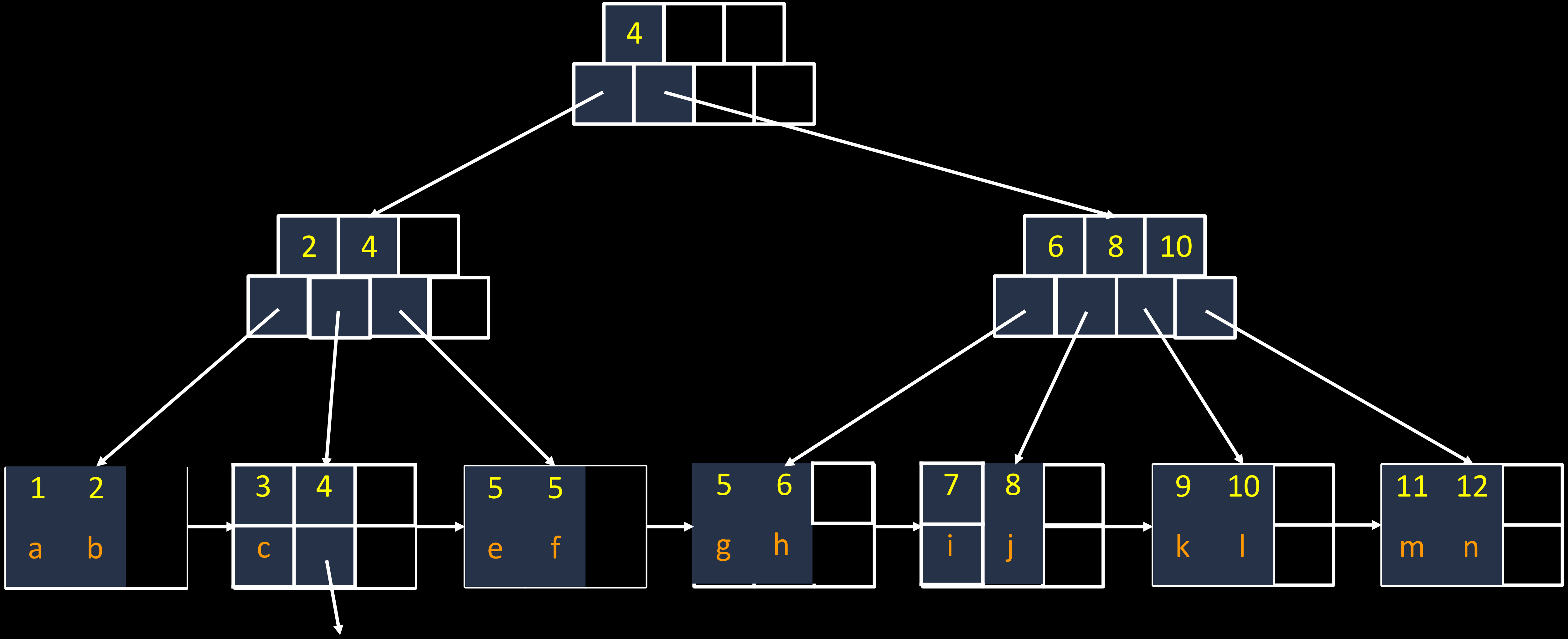
dynamic stage

static stage

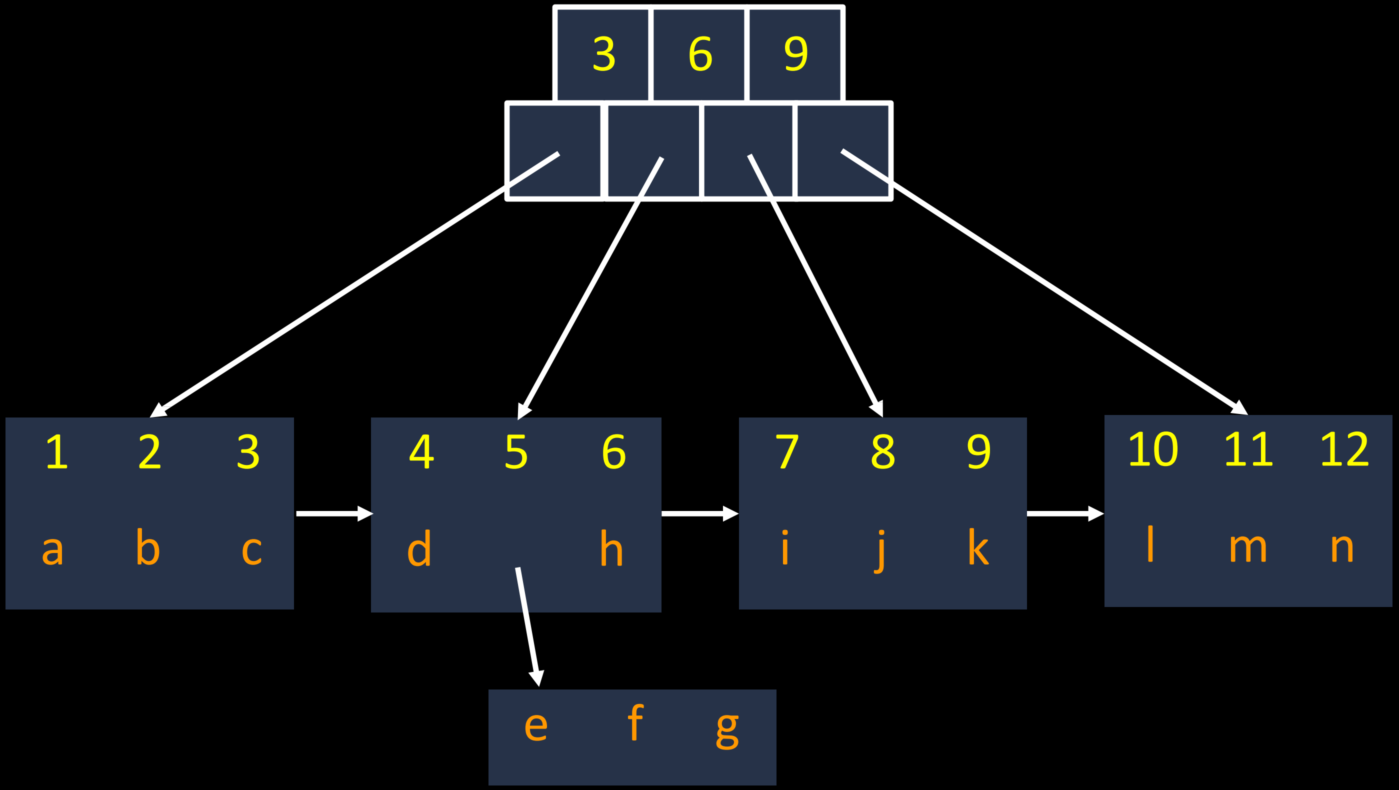
The Dynamic-to-Static Rules

- Compaction
- Reduction
- Compression

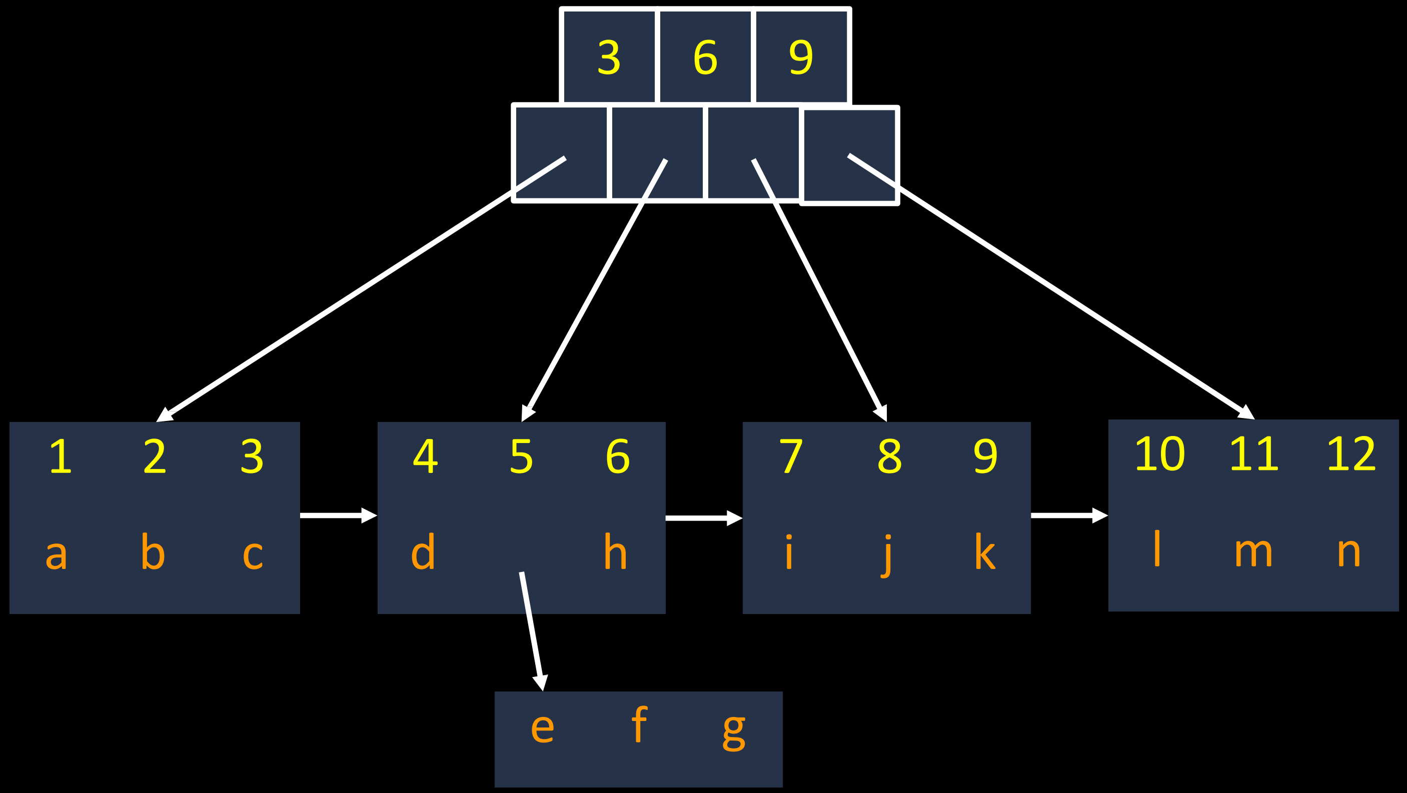
Compaction: minimize # of memory blocks



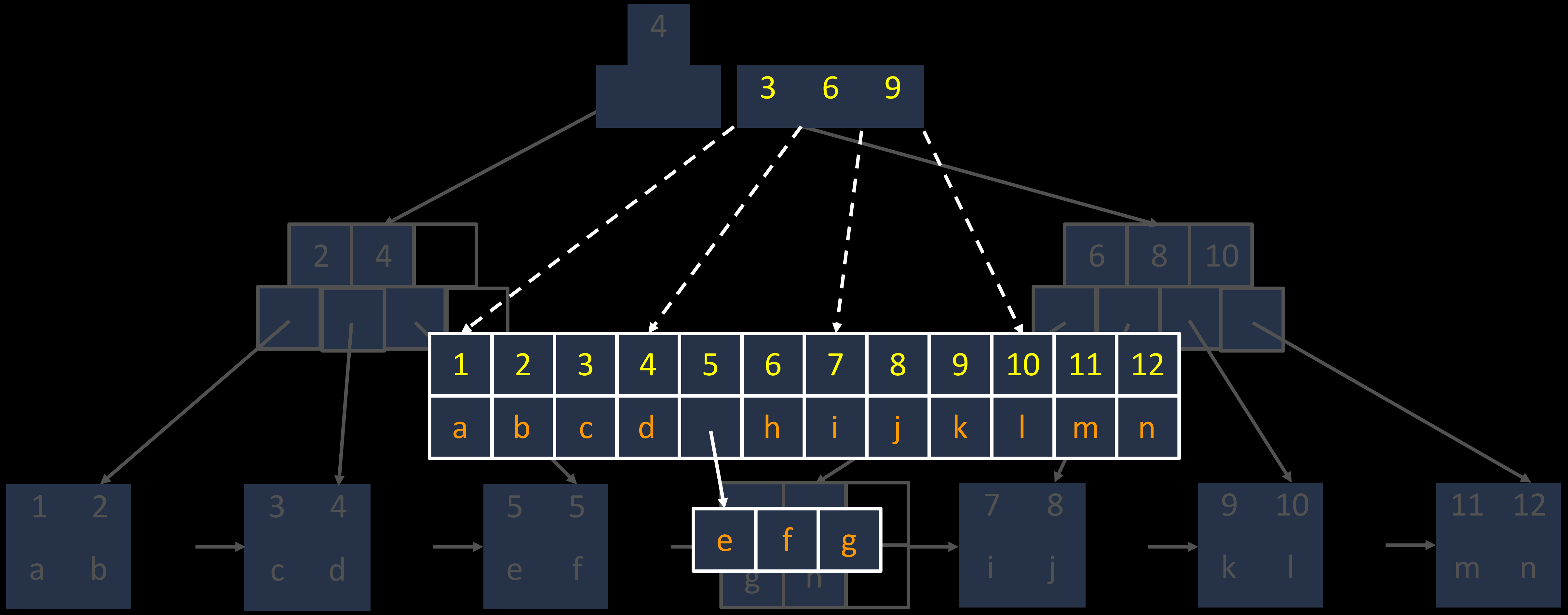
Compaction: minimize # of memory blocks



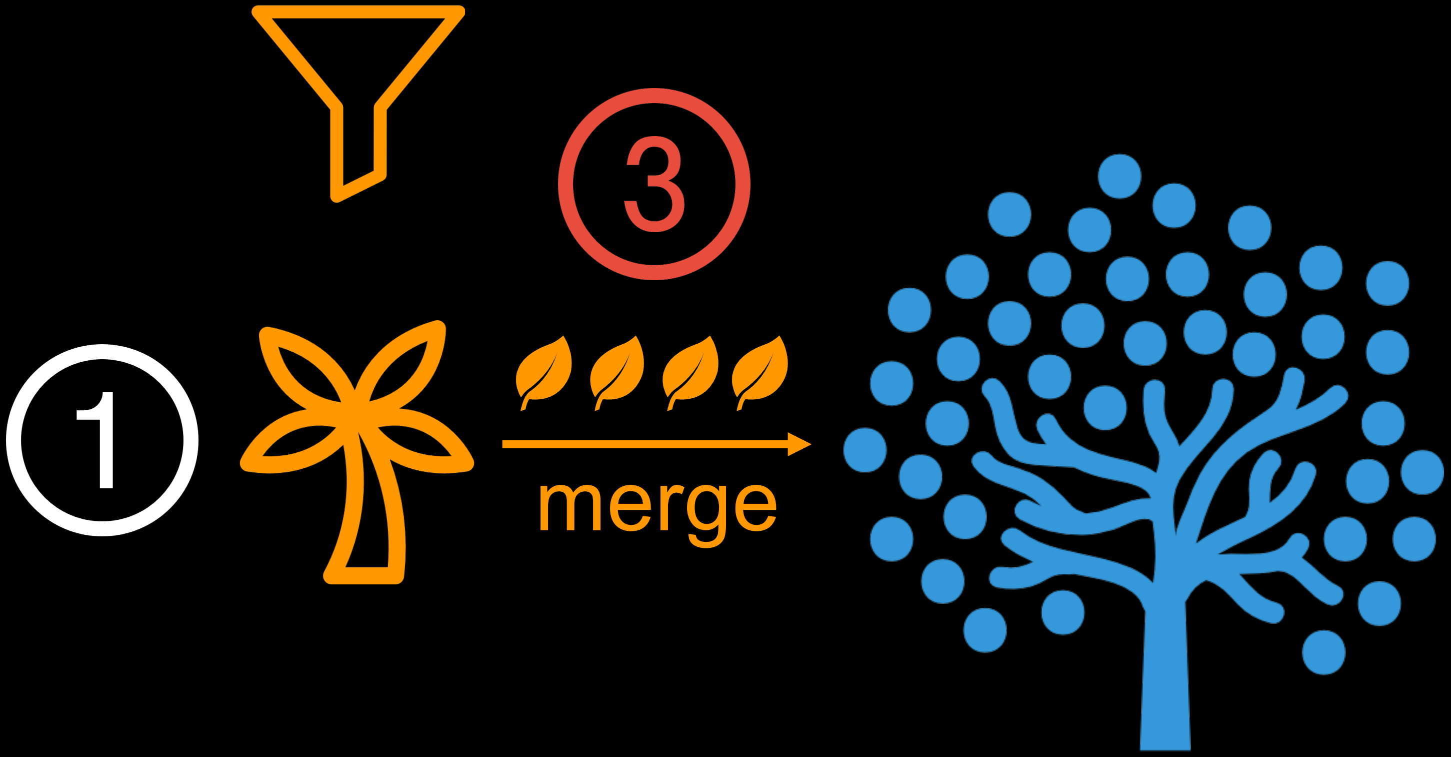
Reduction: minimize structural overhead



Reduction: minimize structural overhead



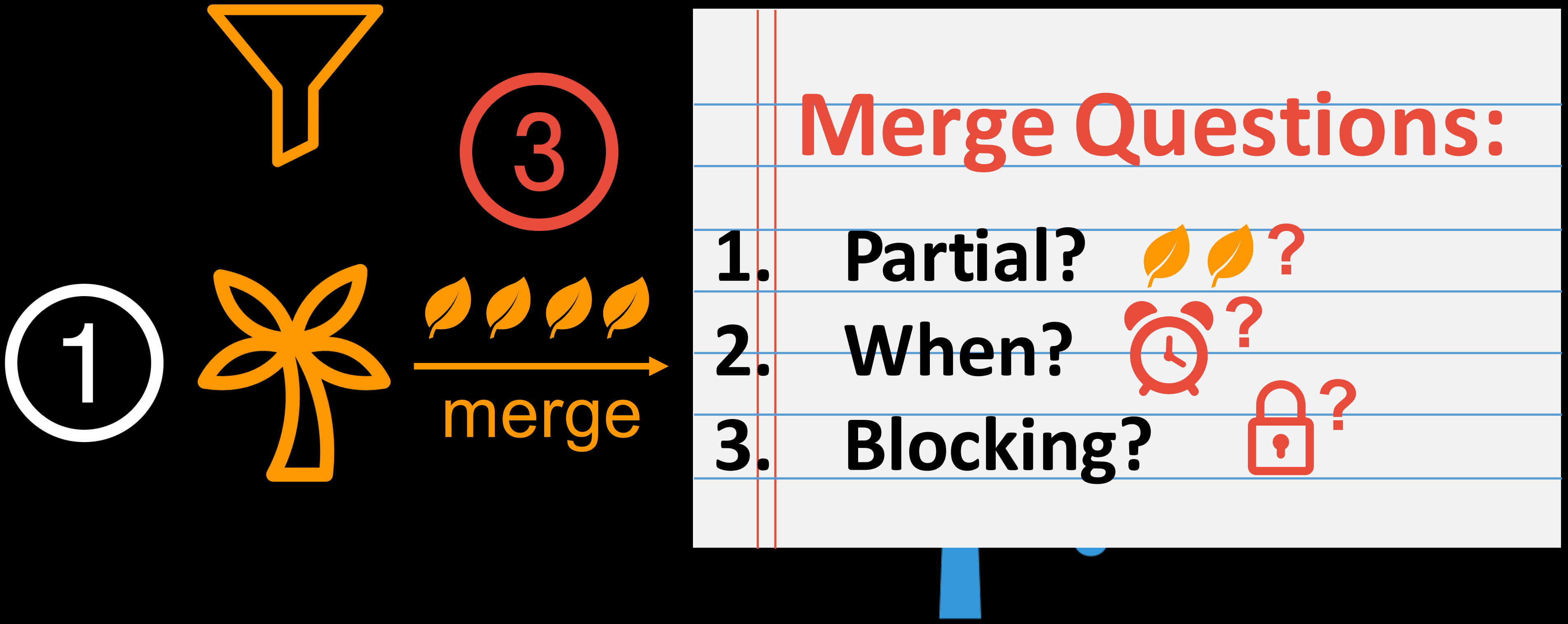
The Dual-Stage Transformation



dynamic stage

static stage

The Dual-Stage Transformation



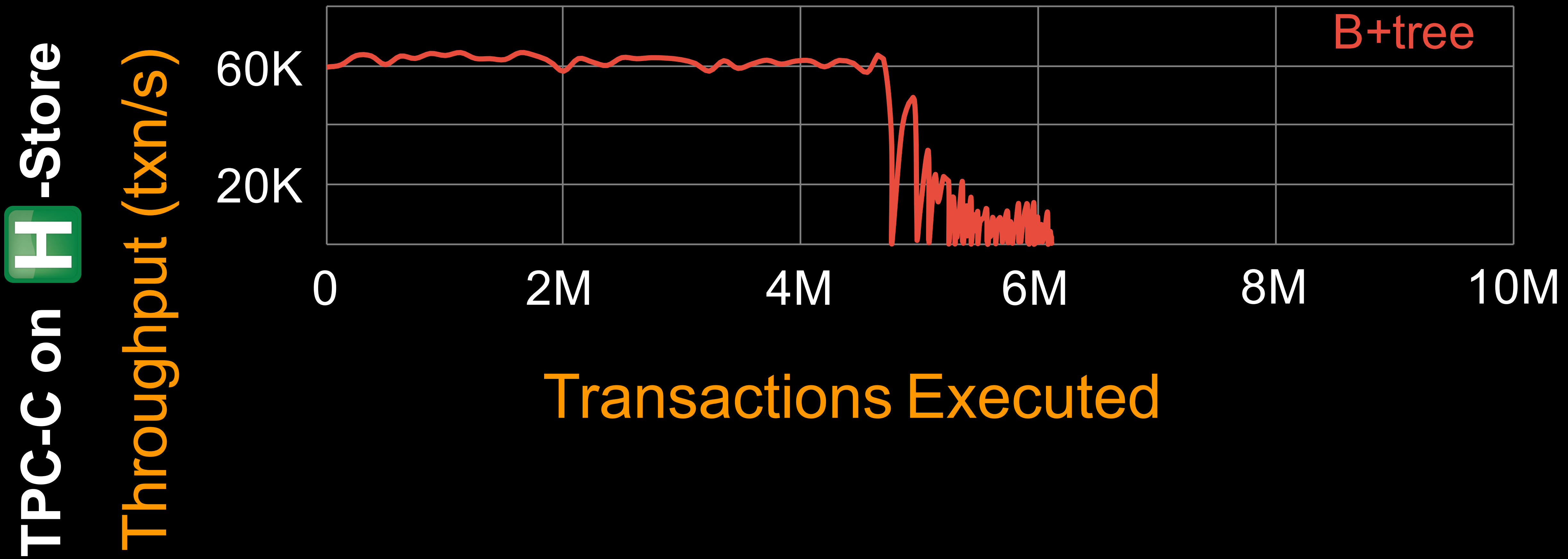
Merge Questions:

1. Partial?
2. When?
3. Blocking?

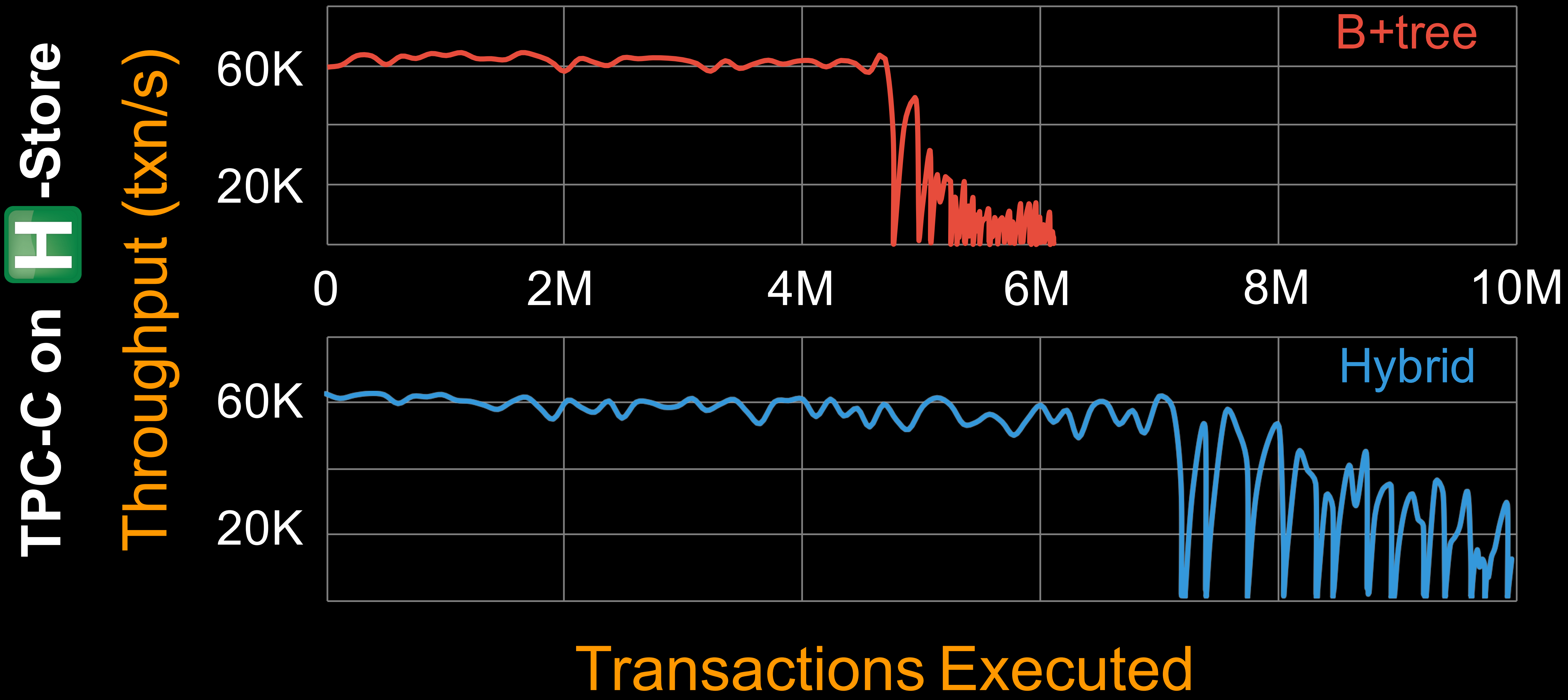
dynamic stage

static stage

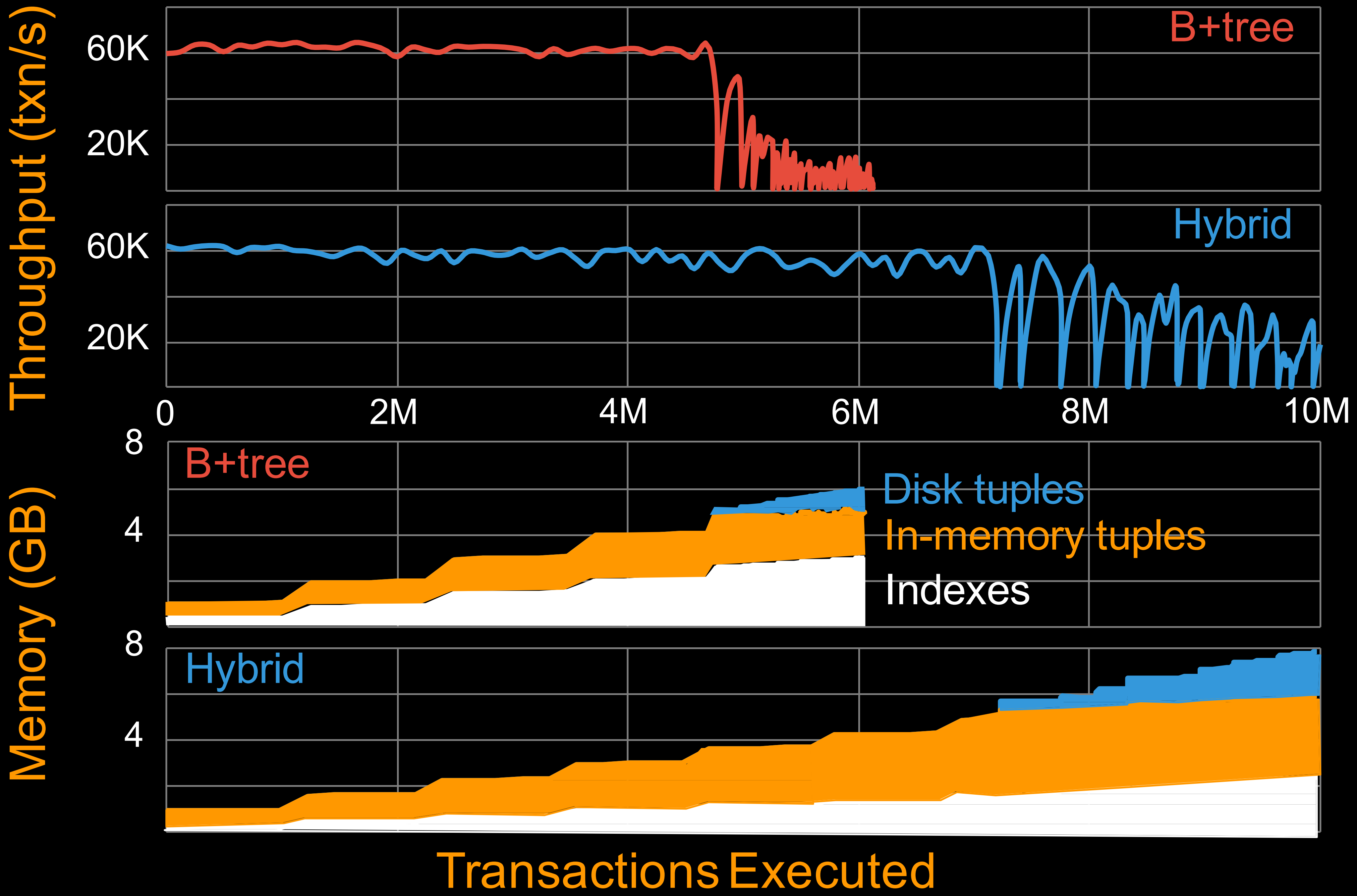
Did we solve this problem?



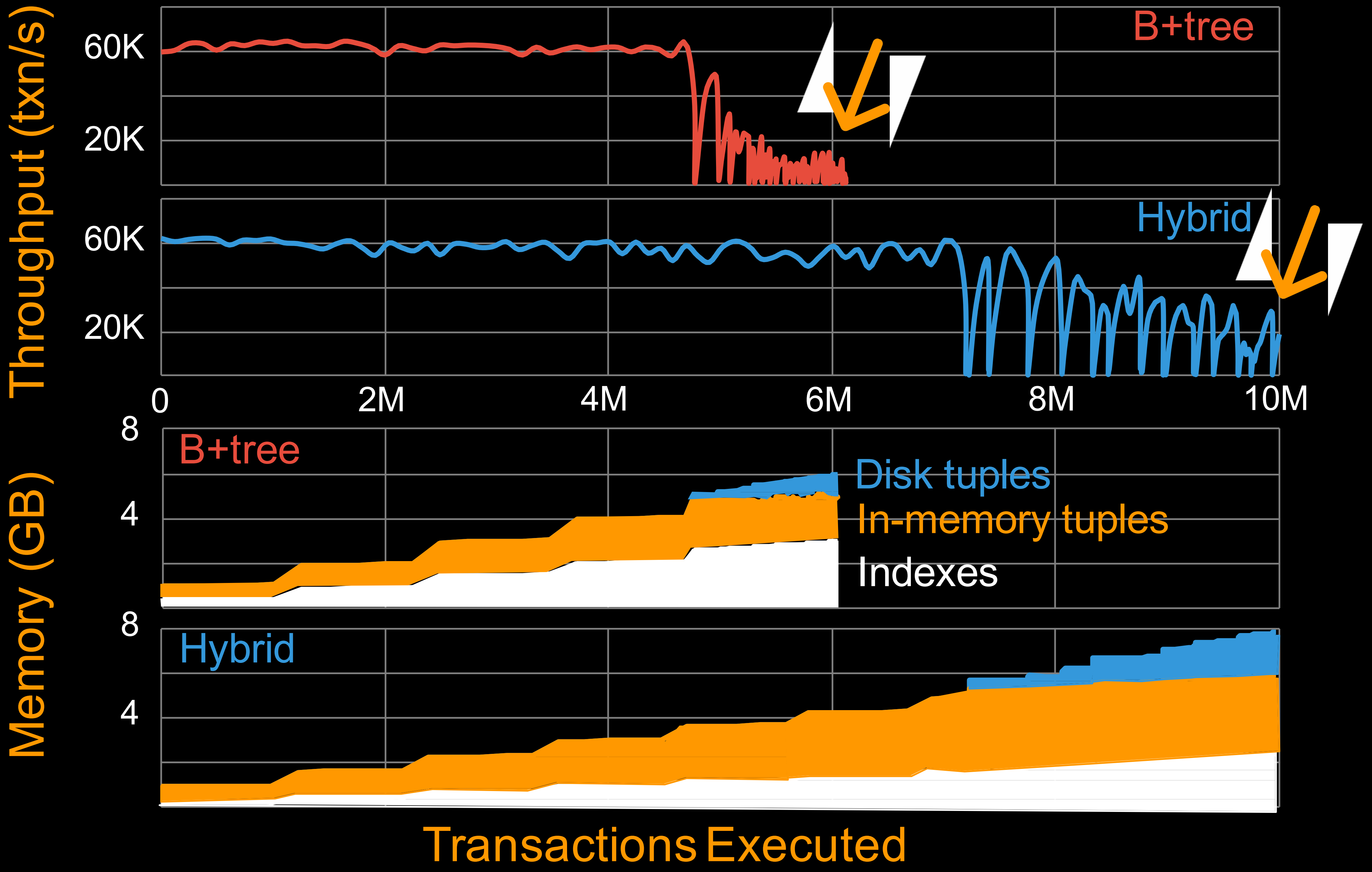
Yes, we improved the DBMS's capacity!



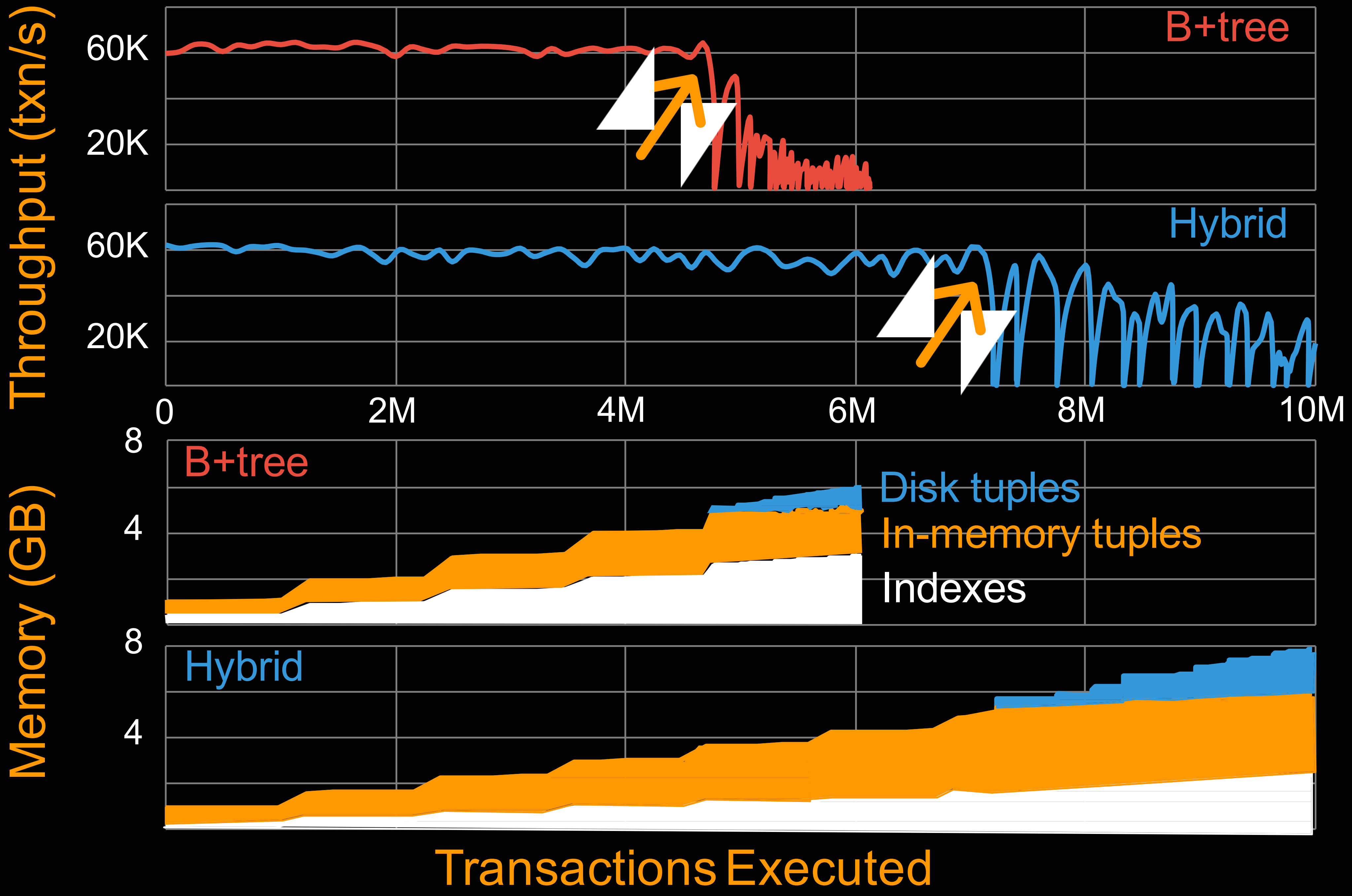
TPC-C on H-Store



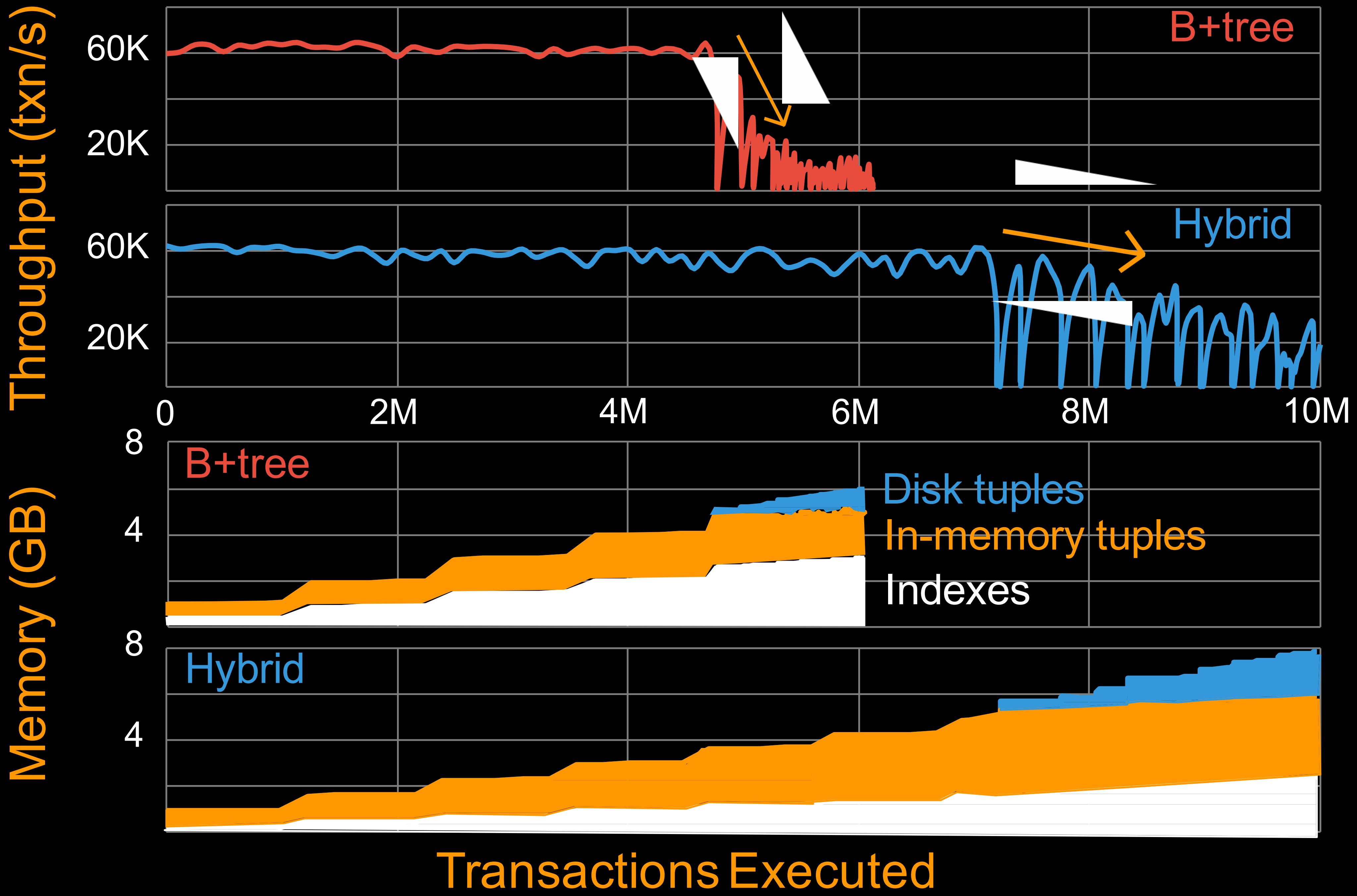
TPC-C on H-Store



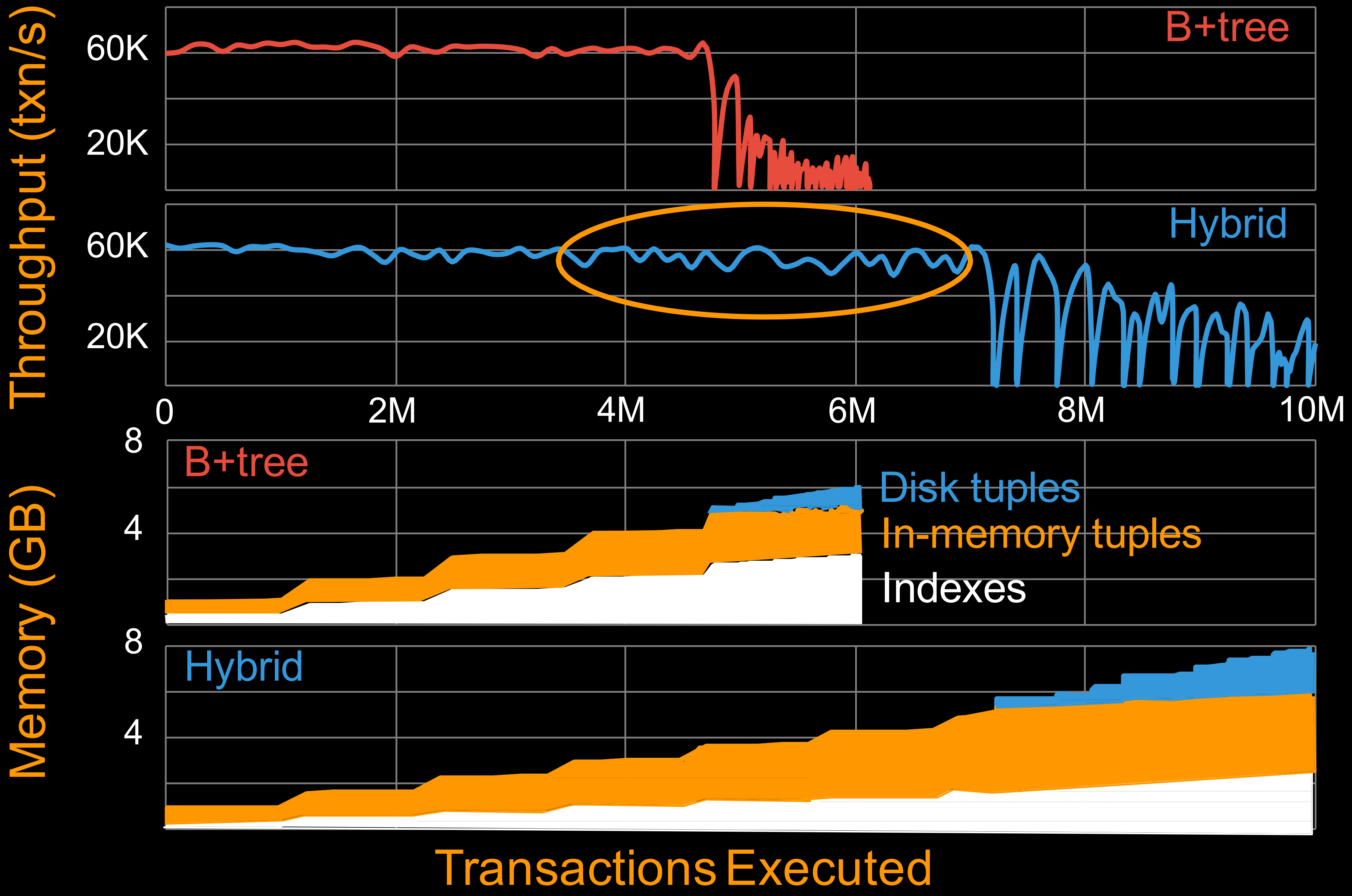
TPC-C on H-Store



TPC-C on H-Store



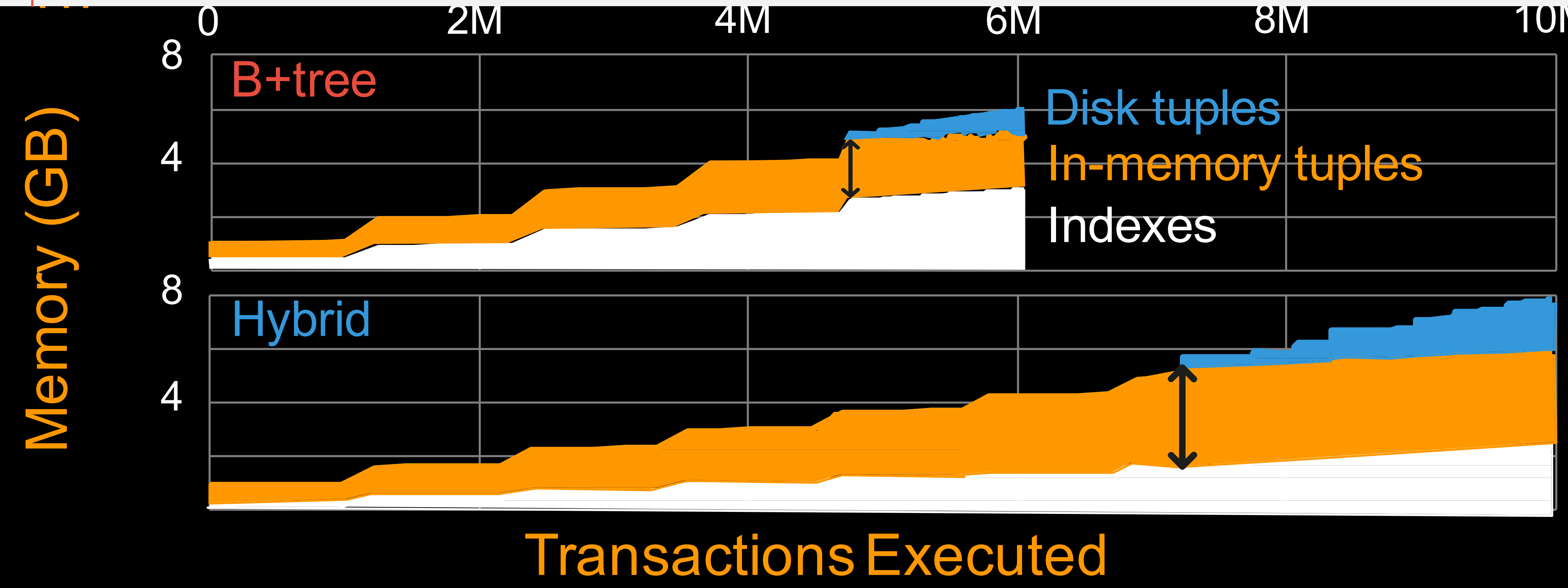
TPC-C on H-Store



Take Away:

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

TPC-C on



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)

Hybrid Index Inspirations:

1. A Tradeoff Research among Data Tuples, Indexes and Evicted Tables in Memory Consumption
2. Dual-stage Architecture of Indexes in Hybrid Memory
3. A Memory-efficient Hash Table Index with Range Query Optimization (DHT might be possible)
4. Non-blocking Merging for Hybrid Indexes with COW

Anti-caching Inspirations:

1. Revisit Anti-caching Mechanism for OLTP Workloads in Hybrid Memory (Evict cold tuples to NVM)
2. Non-blocking Eviction in NVM-Optimized Anti-caching

Non-volatile Memory Inspirations:

1. DiRedis: A NVM-Optimized KV-Store Based on Redis
2. Rethinking Program Scheme in Persistent Memory Era

Thanks !

Note website:
http://kaixinhuang.com/Research/hybrid-index-db/

The Art of
Research Presentation