BG3: A Cost Effective and I/O Efficient Graph Database in ByteDance

Chen Cheng

ByteDance字节跳动

About Me

RD from ByteGraph team in ByteDance
Ph.D. from the National University of Singapore

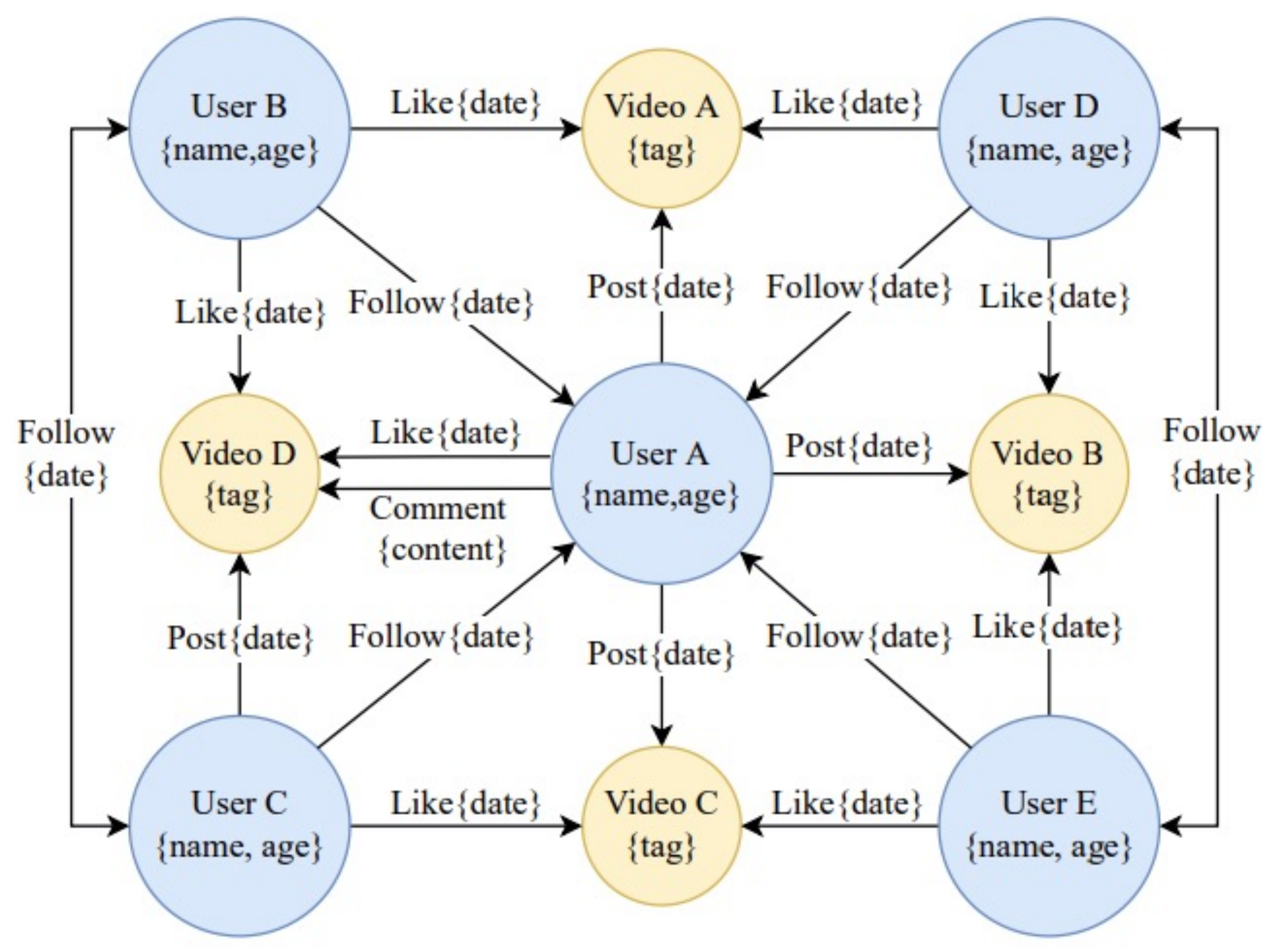
Research：
- Published ~40 papers in top-tier conferences and journals, with over 1200 citations
- Interests include GNN, AI infrastructure, Graph Computing, Graph Databases, and Storage

Awards:
- Best Paper (Runner up) - EuroSys 2024
- Best Industry Paper (Runner up) - VLDB 2023

Services:
- PC member of ICDE 2025 (research track)
- PC member of ICDE 2024 (industrial track)
- PC member of SIGMOD 2024 (industrial track)

Overview

Background



ByteGraph 2.0 (BG 2.0) is a distributed graph database independently developed by ByteDance, used for managing and processing massive amounts of graph data generated by products such as Douyin, and Toutiao every day.

Its deployment scale is huge, including 1000 clusters, with 1 million CPU cores and 100PB of persistent storage.

The businesses involved include social network relationship data, friend recommendations, video recommendations, search, encyclopedia, e-commerce, knowledge graphs, internal microservices network relationships, and more.

Overview

Architecture of ByteGraph 2.0

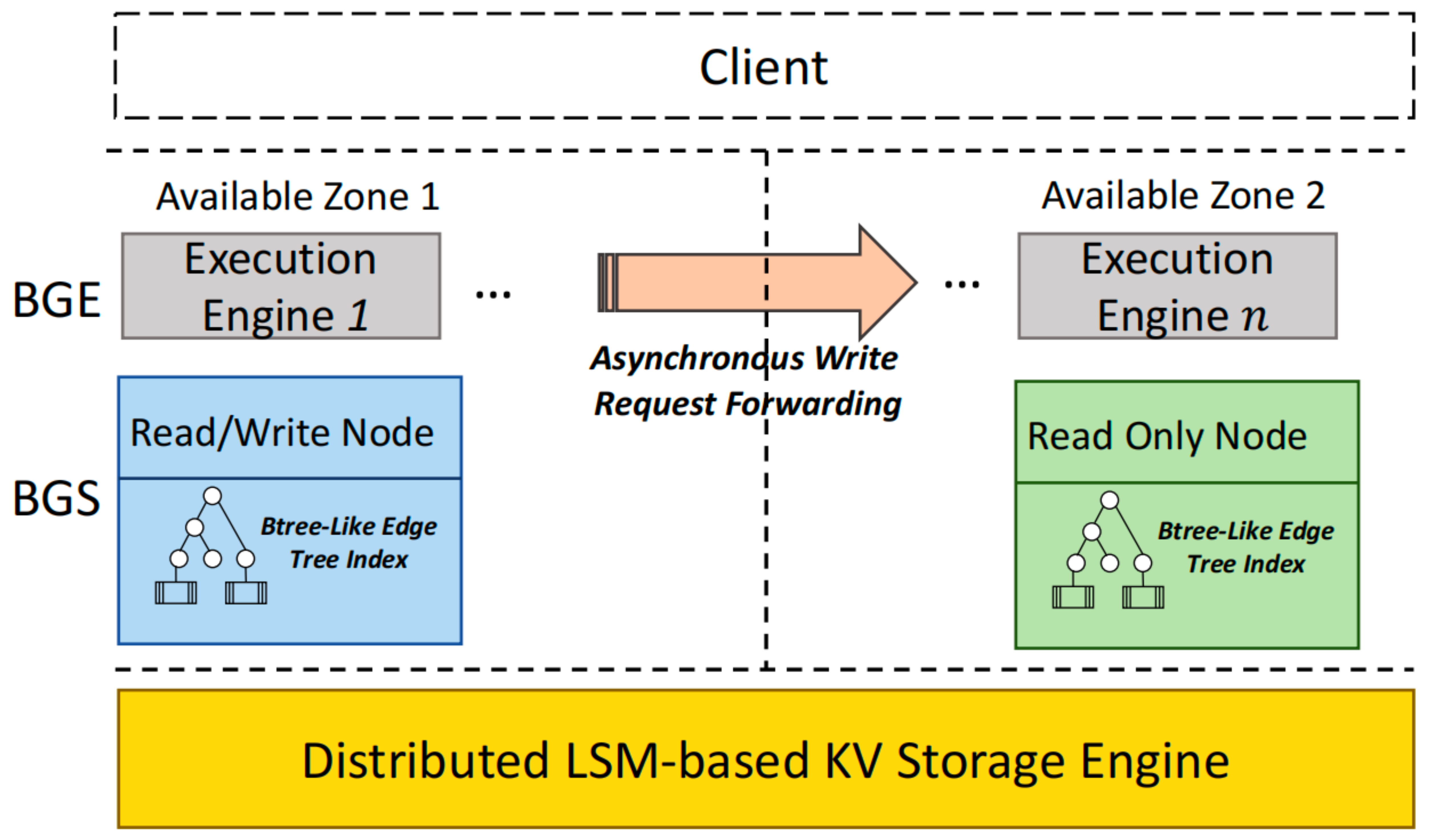

Client
Available Zone 1
Available Zone 2
BGE
Execution Engine 1
...
Asynchrondus Write Request Forwarding
...
Execution Engine n
BGS
Read/Write Node
Btree-Like Edge Tree Index
Read Only Node
Btree-Like Edge Tree Index
Distributed LSM-based KV Storage Engine

Figure 1: The Architecture of ByteGraph

(VLDB2022) ByteGraph: A High-Performance Distributed Graph Database in ByteDance
https://www.vldb.org/pvldb/vol15/p3306-li.pdf

ByteGraph 2.0 Problems -- Issue 1

Inefficient graph access and high operational cost caused by LSM KV.

- Weak read performance of LSM KV.
- Data redundancy due to multiple copies for data safety.
- Write amplification caused by B-tree on LSM architecture.
- Memory redundancy in LSM KV block cache/row cache.

ByteGraph 2.0 Problems -- Issue 2

Workload-unaware space management.

- Write amplification caused by B-tree on LSM architecture.
- Unable to separate hot and cold data or perform space reclamation based on the degree of hotness or coldness.

ByteGraph 2.0 Problems -- Issue 3

Lack of support for scaling real-time graph analytics

- Limitations of the read-write node synchronization mechanism: The legacy system’s read-write node synchronization mechanism uses asynchronous forwarding of write requests, achieving only eventual consistency. This limits the system’s scalability for real-time graph analysis. Because it cannot synchronize read and write nodes within a limited time, the system struggles to meet the demands of graph analysis tasks that require high levels of real-time responsiveness.

Overview

ByteGraph 3.0 Achitechture

We propose ByteGraph 3.0 (BG3), which includes:

Design 1: Space Optimized BwTree Forest.

Design 2: Read Optimized BwTree

Design 3: Workload-Aware Space Reclamation

Design 4: I/O Efficient Synchronization Mechanism

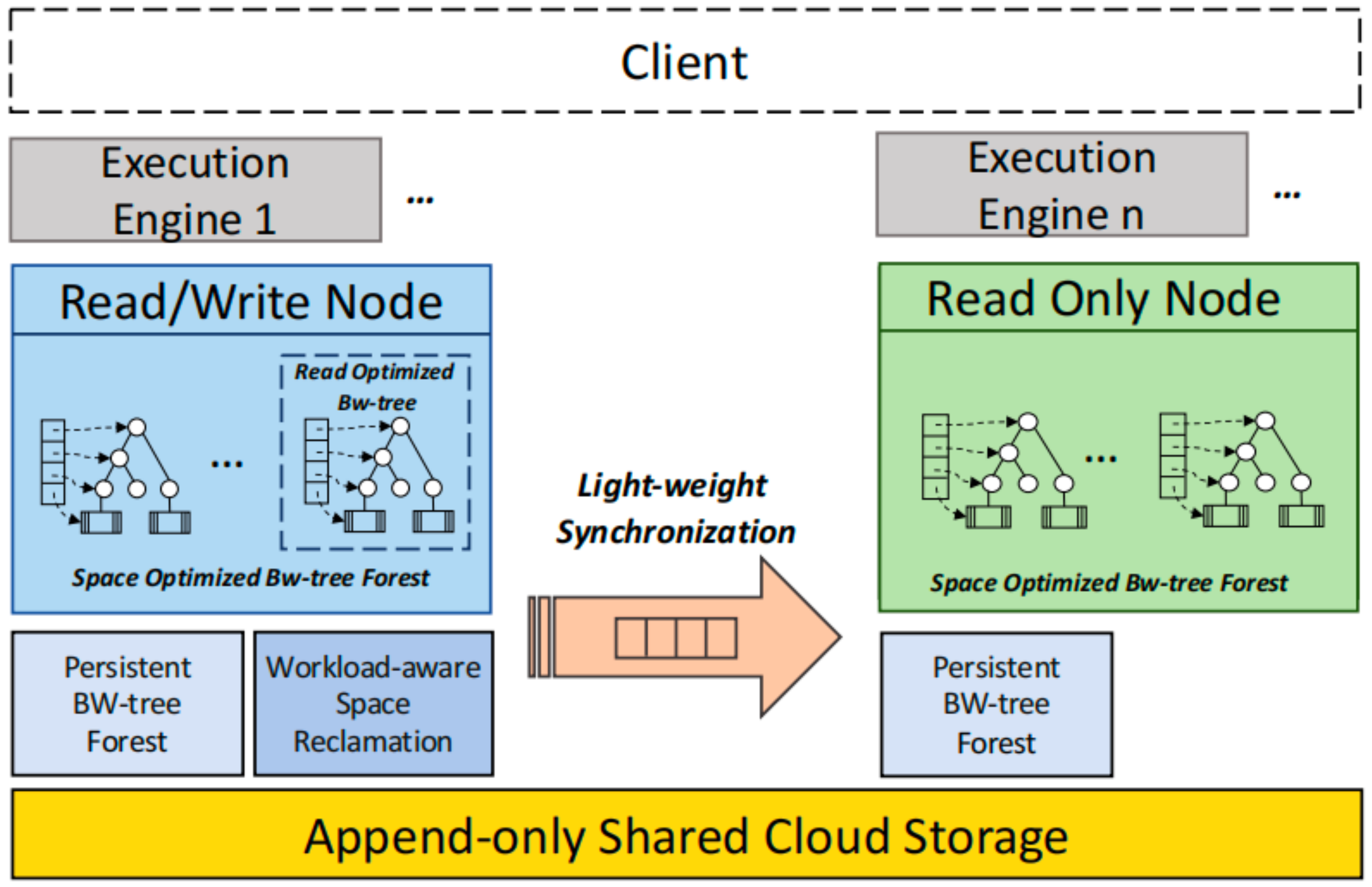

Client
Execution Engine 1
...
Execution Engine n
...
Read/Write Node
Read Optimized Bw-tree
...
Space Optimized Bw-tree Forest
Light-weight Synchronization
Read Only Node
...
Space Optimized Bw-tree Forest
Persistent BW-tree Forest
Workload-aware Space Reclamation
Persistent BW-tree Forest
Append-only Shared Cloud Storage

Figure 2: The architecture of BG3

(SIGMOD 2024) BG3: A Cost Effective and I/O Efficient Graph Database in Bytedance

Design 1: Space Optimized Bwtree Forest

If All Edges in One Bw-Tree

- **Pros:** Memory efficiency
- **Cons:** High conflict rate, low scalability

If Each Vertex Adjacency List in a Separate Bw-Tree (forming a forest of Bw-Trees):

- **Pros:** Low conflict rate, high scalability
- **Cons:** Not memory efficient

Solution: Space-Optimized Bw-Tree Forest

- Insert into a common tree when a vertex's out-degree is small.
- Split into a separate tree when the vertex's out-degree is large.

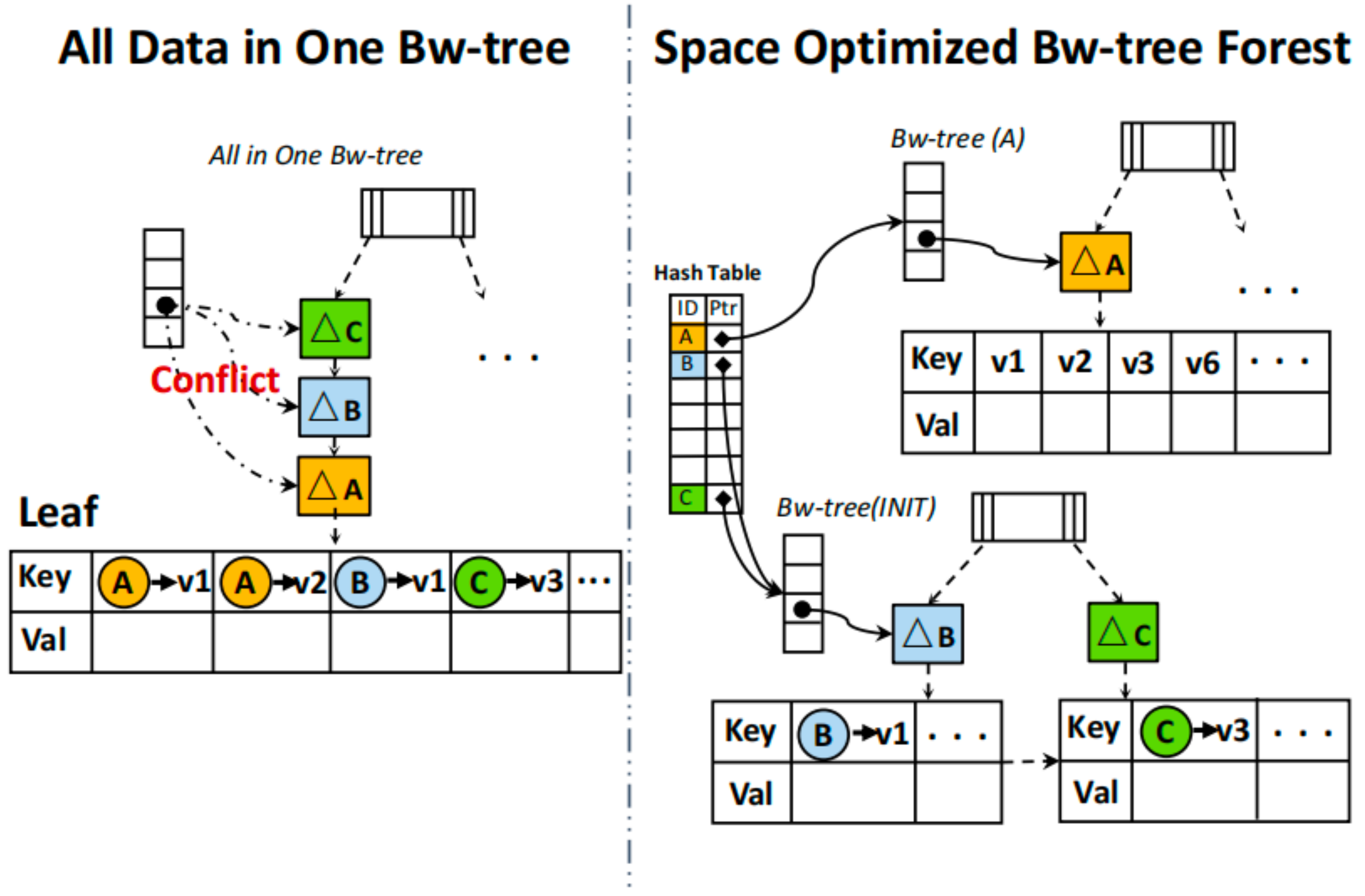


Figure 3: Space Optimized Bw-tree Forest

Design 2: Read Optimized BwTree

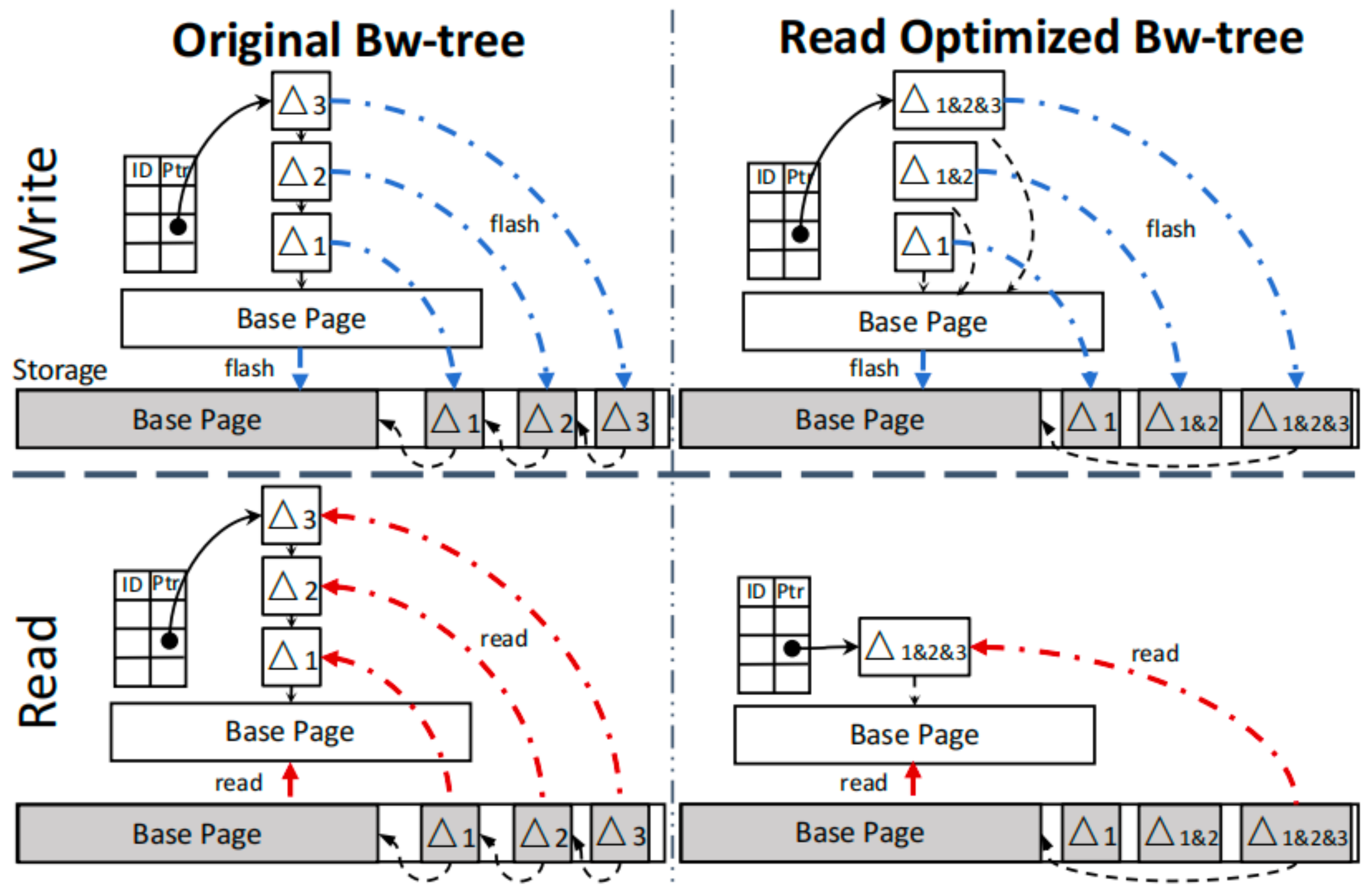


Figure 4: Read & Write Process Comparison

- Compared to the native Bw-tree's delta-chain implementation, the read-optimized B-tree retains only a single delta for each page. This approach sacrifices some write amplification to reduce read IOPS, making it more suitable for read-heavy and write-light workloads.

Design 3: Workload-Aware Space Reclamation

Three strategies:

- **Update Gradient:** Represents the rate of invalid page growth for each extent.
- **Fragmentation Rate:** Represents the rate of invalid pages for each extent.
- **Time-to-Live:** Indicates the expiration time of the data.

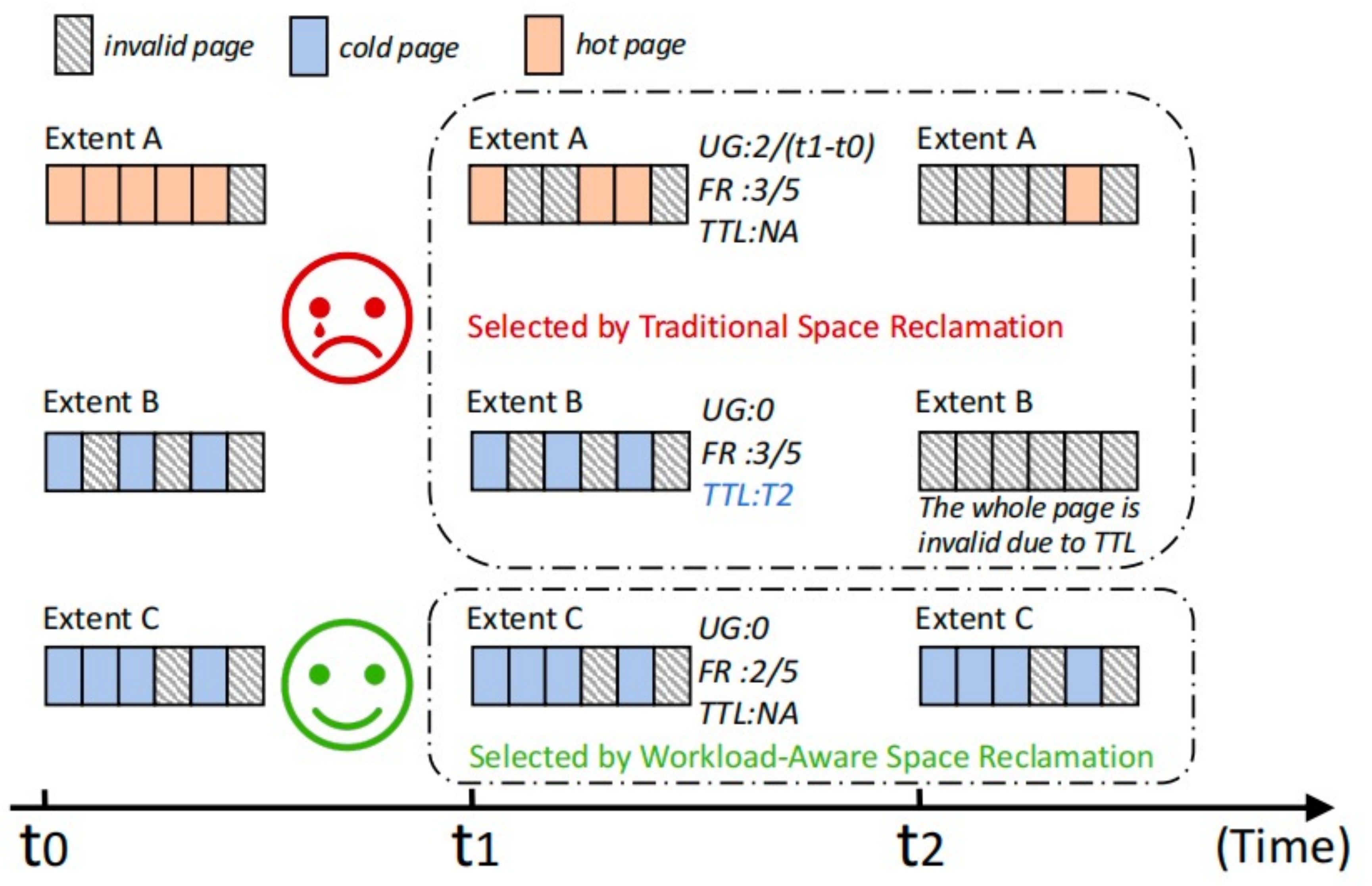


Figure 5: Spatial Changes of Different Extents (UG stands for Update Gradient; FR stands for Fragmentation Rate; TTL stands for Time to Live)

Design 4: I/O Efficient Synchronization Mechanism -- Issue

Mechanism: Write-Ahead Log Synchronization

- Read-write nodes write data update operations to the write-ahead log and store it in shared storage.
- Read-only nodes read these logs from the shared storage and replay them in memory to achieve data synchronization with the read-write nodes.

Issue:

- B-tree split will cause some read inconsistency, as shown in the figure.

13 end

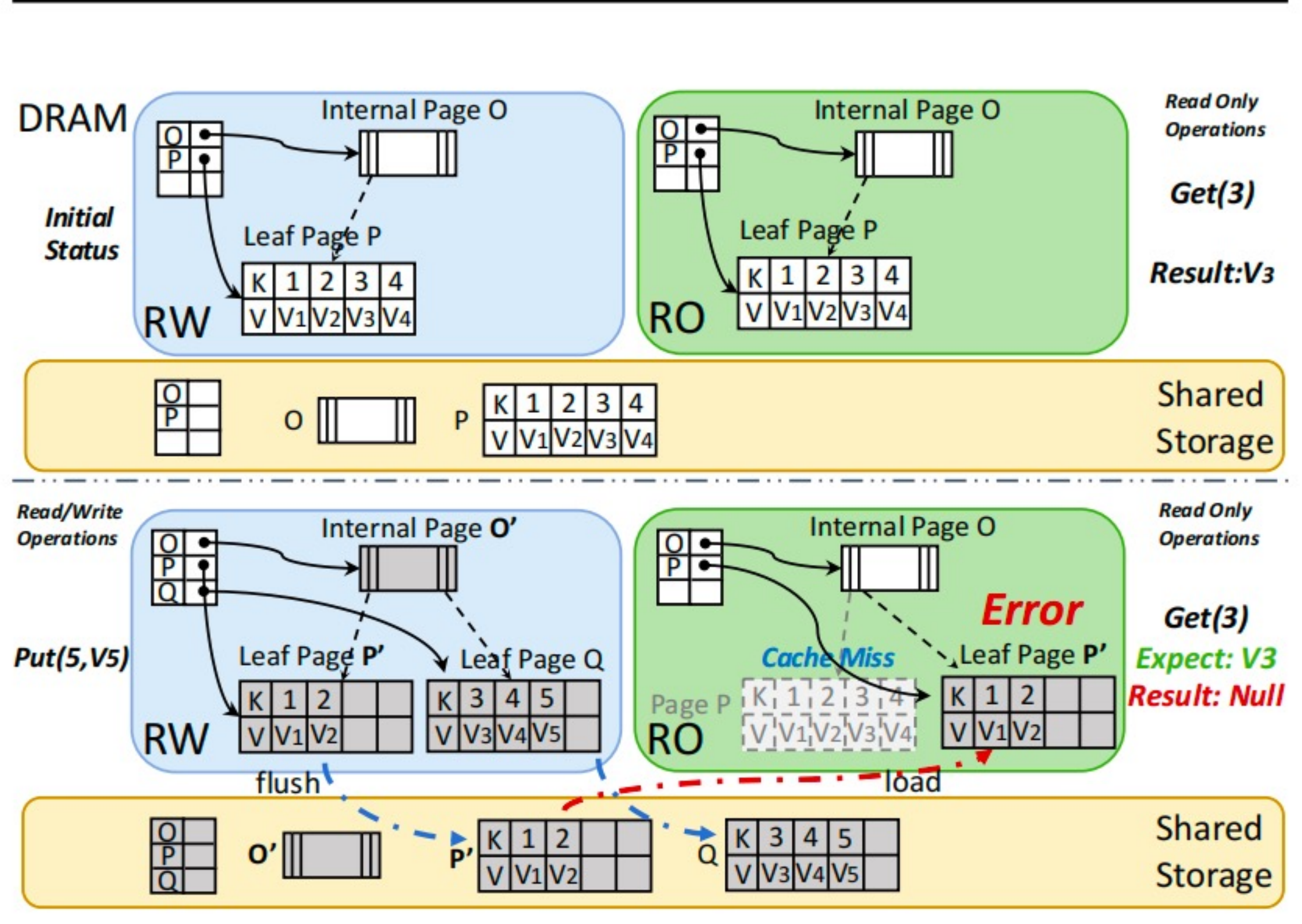


Figure 6: Data Inconsistency Issue

Design 4: I/O Efficient Synchronization Mechanism -- Solution

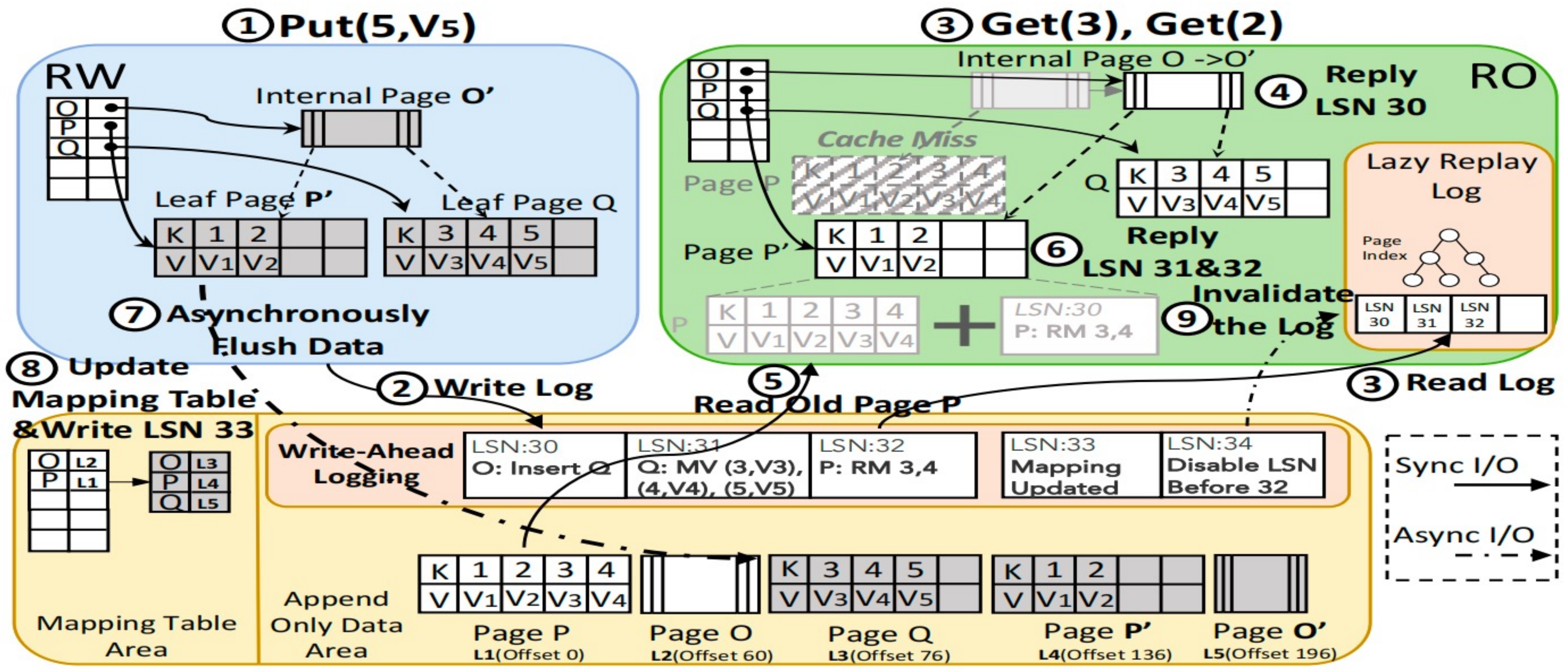


Figure 7: Workflow of I/O Efficient Synchronization

Unify WAL Stream:

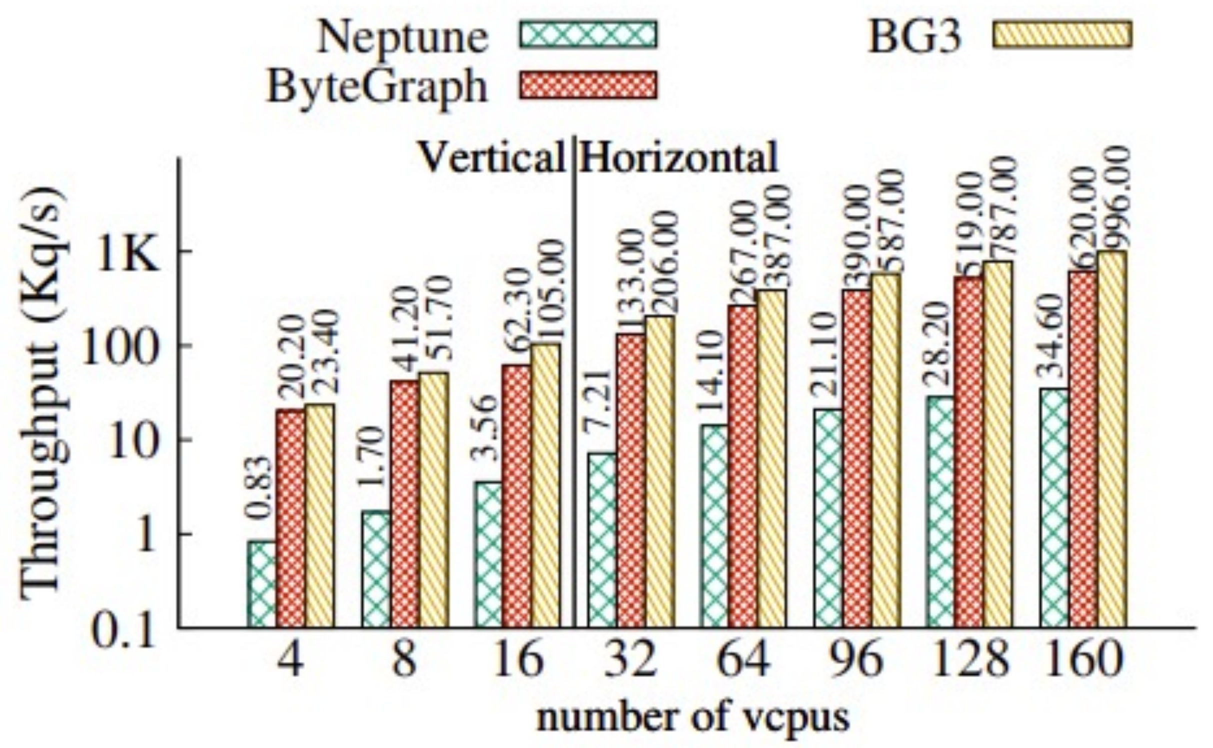
- BG3 ensures consistency by maintaining multiple versions of the data and synchronizing the read-write node's flush/update mapping operations in the log stream.

Overview

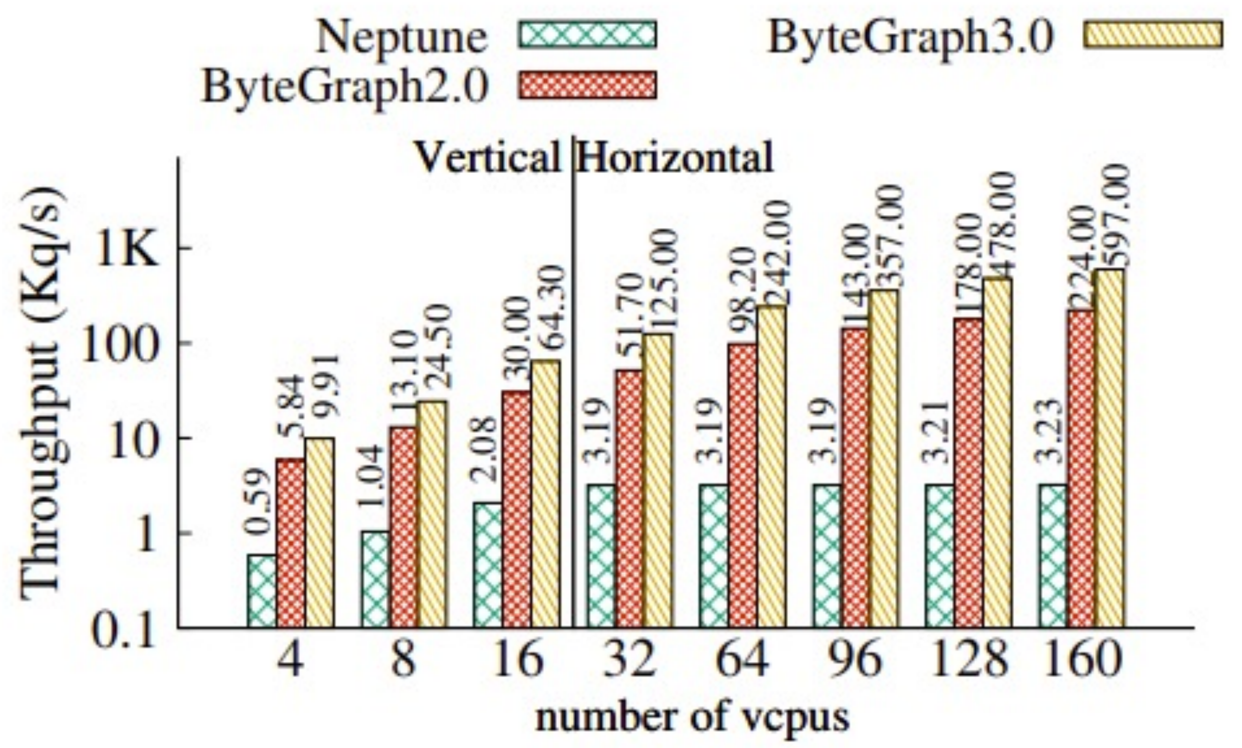
Evaluation

Table 1: Workload description.

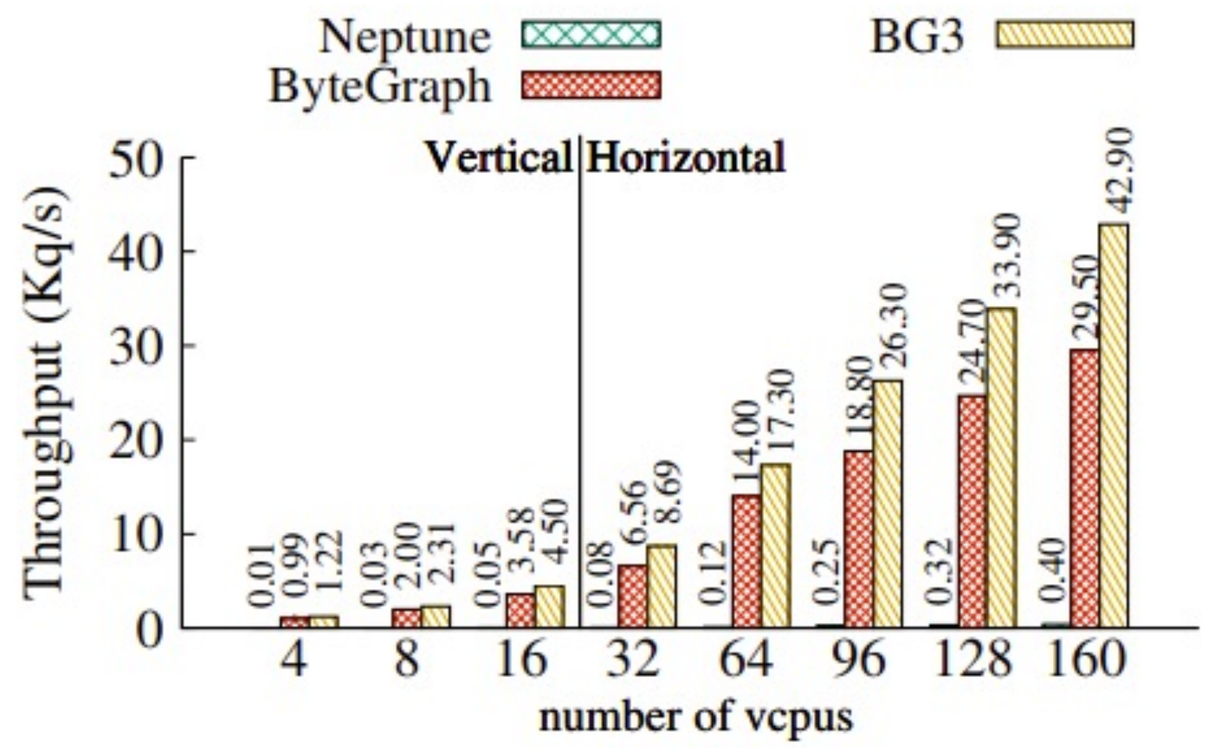
Workload	Read/Write	Description	|V|	|E|	Hops
Douyin Follow	99%/1%	Managing Douyin *Follow* records, single edge insertion, one-hop neighbor query	3M	0.5B	1
Financial Risk Control	50%/50%	Pattern matching[32], single edge insertion, full graph reading, 10 hops and 100 edges	5B	100B	5 to 10
Douyin Recommendation	read-only	multi-hop neighbor query, 70% 1-hop, 20% 2-hop, and 10% 3-hop	3M	0.5B	1 to 3



(a) Douyin follow



(b) Financial risk control



(c) Douyin recommendation

Figure 8: Overall Performance (Vertical: a single machine; Horizontal: 2 to 10 machines, each with 16 cores)

Evaluation

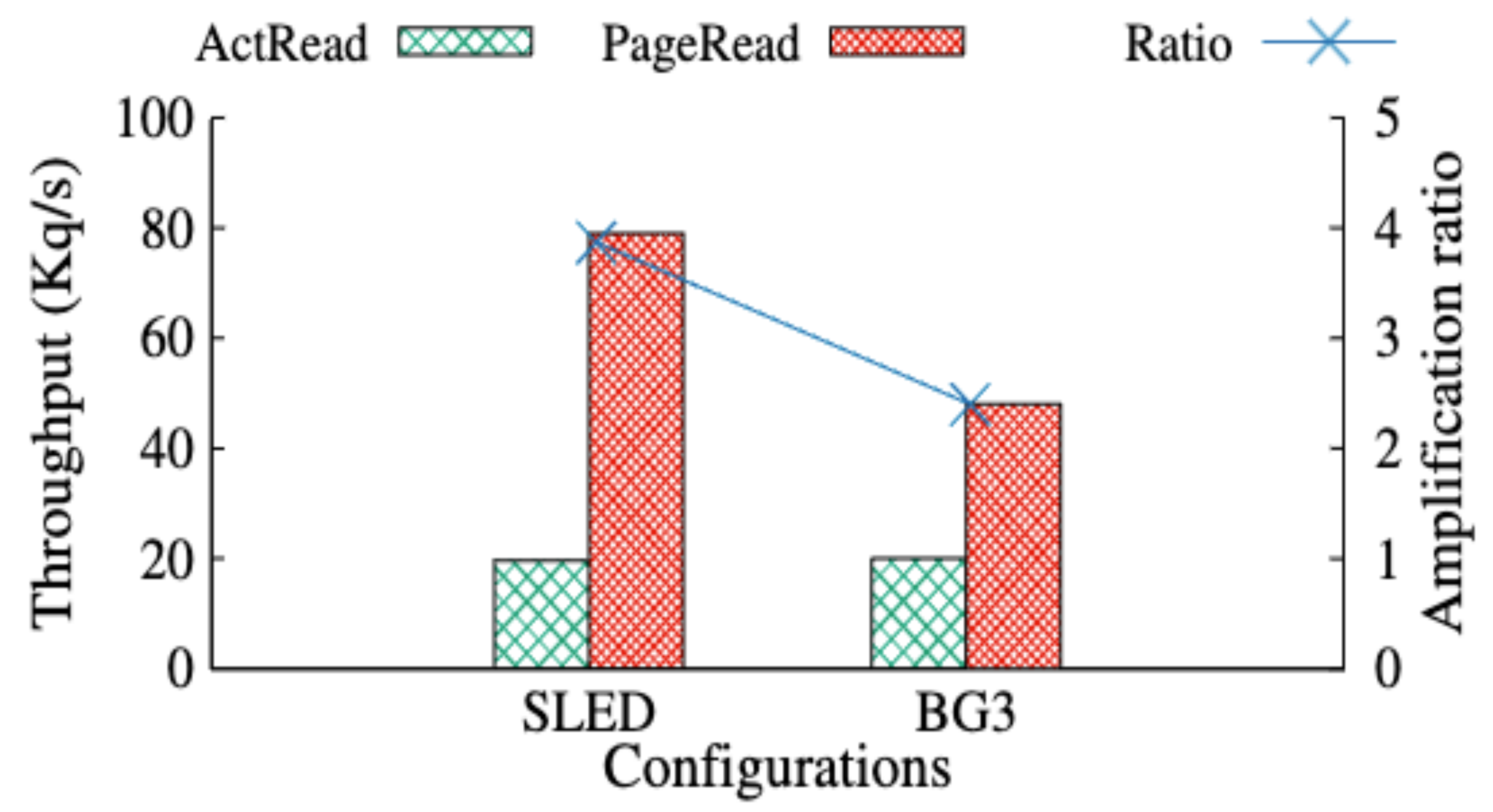


Figure 9: Read Amplification Comparision Between the Traditional Bw-tree and the Read Optimized Bw-tree.

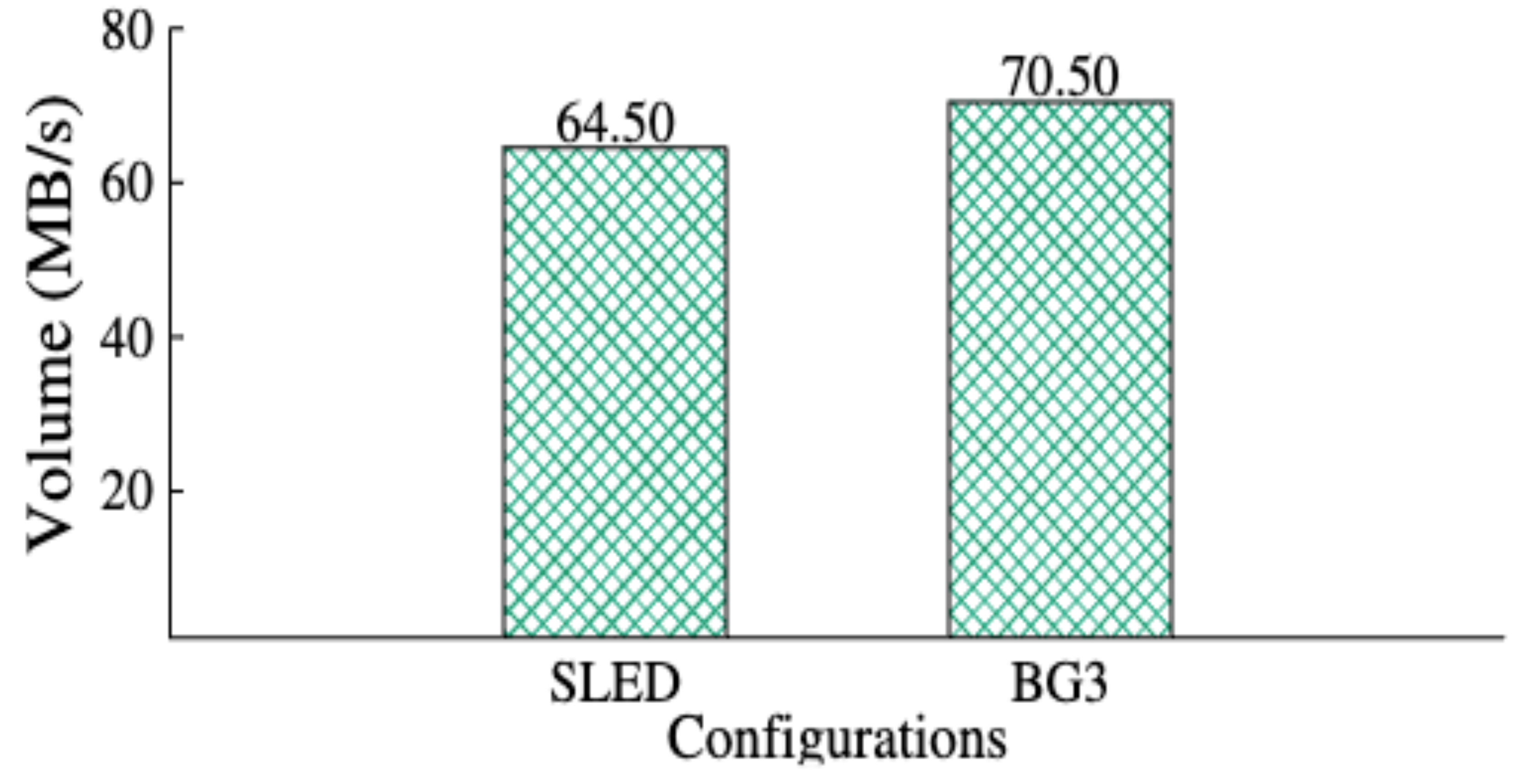


Figure 10: Write Bandwidth Comparision Between the Traditional Bw-tree and the Read Optimized Bw-tree.

Overview

Conclusion

We present BG3, a high-performance distributed graph database for the efficient management and processing of large-scale graphs at ByteDance.

- BG3 provides a new storage engine based on cost-effective shared storage and BW-tree indexes, a workload-aware space reclamation mechanism, and a lightweight yet efficient leader-follower synchronization mechanism.
- We experimentally show that BG3 achieves competitive performance compared to Amazon Neptune and ByteGraph 2.0.

Join US

We are hiring, looking for talents with expertise in

- Graph Database,
- Graph Computing or Graph Neural Network.

Please feel free to drop me an email at chencheng.sg@bytedance.com.

Thanks

Q&A

ByteDance 字节跳动