REXIO: Indexing for Low Write Amplification by Reducing Extra I/Os in Key-Value Store under Mixed Read/Write Workloads

Speaker: Zizhao Wang

Authors: Zizhao Wang, Qiang Qu, Nan Han, Zhelang Deng, Yizhuo Ma, Xiaowen Huang, Jintao Meng*

Date: 5, December, 2024

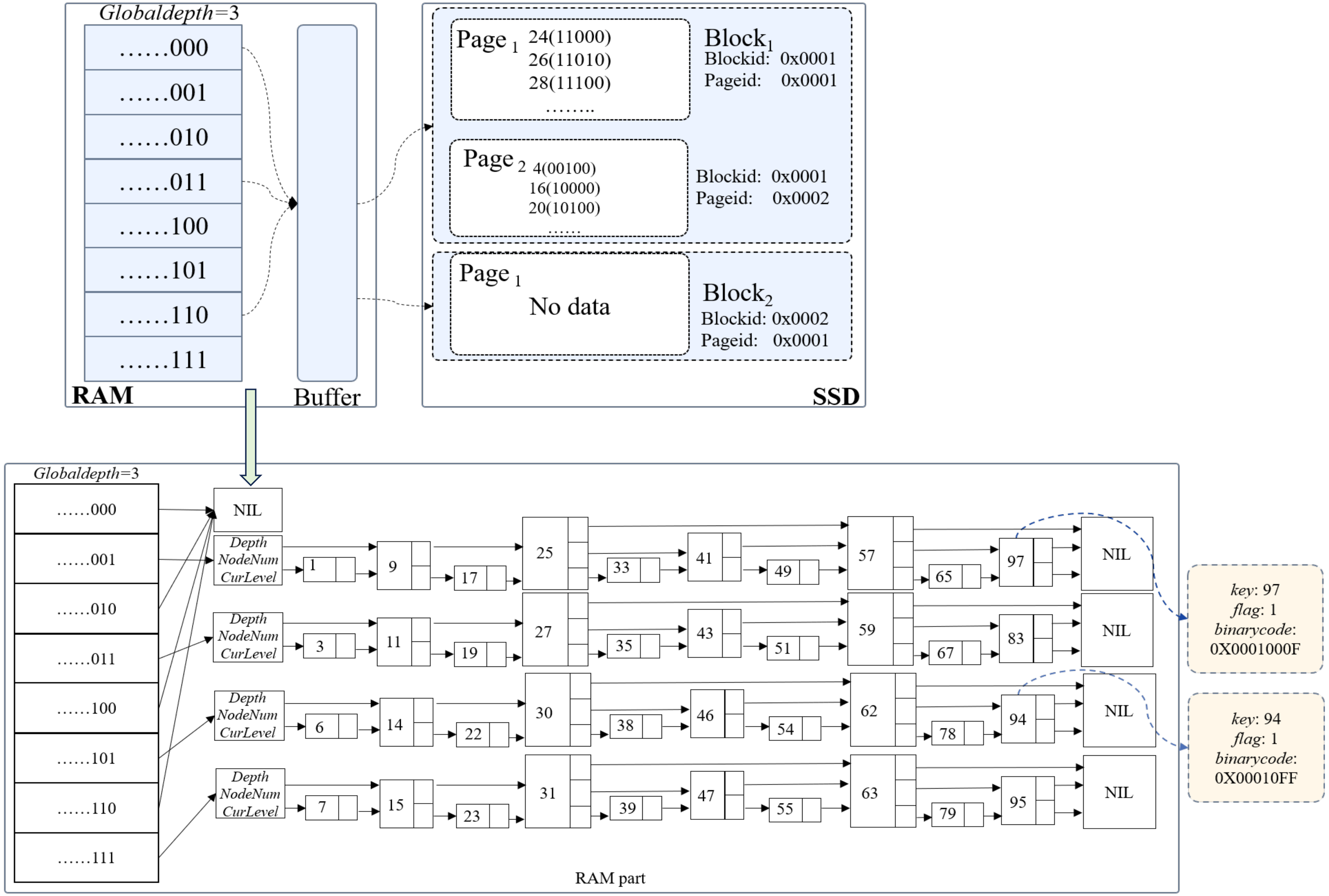
The key value (KV) store on SSD is now facing a serious problem: ***I/O amplification***.

Can we modify existing LSM-trees employed in KV store to solve this problem?

- An intrinsic ***compaction*** process used in LSM trees
- It can be somewhat reduced(WiscKey, Partitioning, etc.)

How to reduce I/O amplification while accessing data efficiently?

Proposed method: REXIO



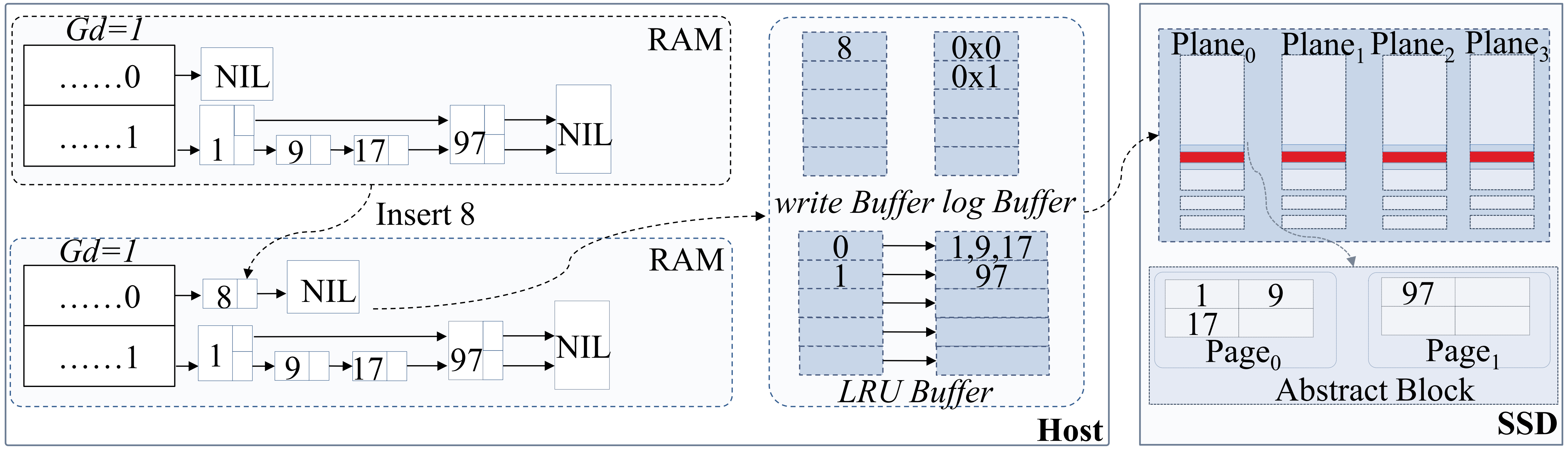
◆We propose an indexing approach for KV stores in r/w heavy workloads that decouples RAM from the SSD, eliminating the extra I/Os.

◆We employ an In-RAM hashing table that stores the keys and addresses of persistent KV pairs to reduce buffer invalidation and avoid data reorganization.

◆ We also introduce the \textit{In-block logging} within the index, designed to transform deletions into sequentially writing \textit{bianrycode}.

◆We conduct experiments on an NVMe SSD. The results demonstrate that our method significantly reduces WA in r/w heavy workloads.

Workflow: An inserting case



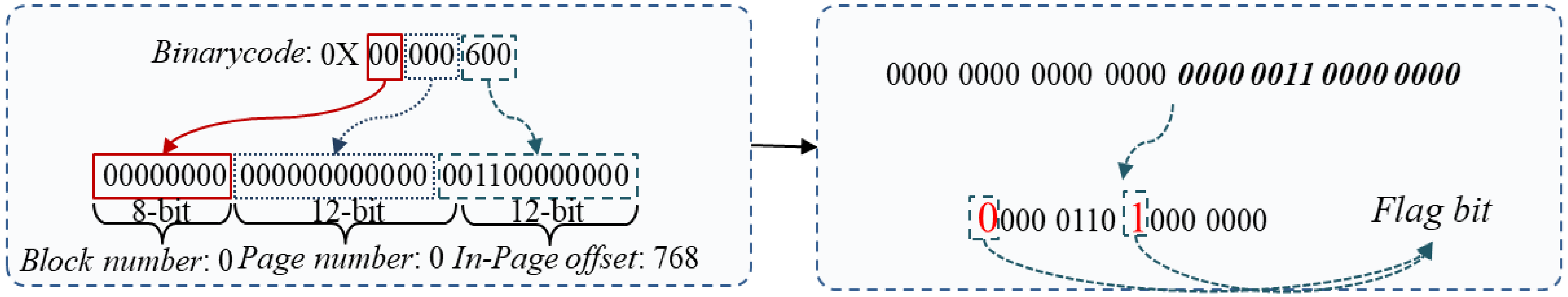
Overall architecture of *NEXIO* one-time I/O operation

- RAM-disk decoupling, eliminating the need for RAM to mirror the disk's storage structure.
- Maintain physical addresses of each KV pair in RAM.
- In-block logging is employed to transform delete and update operations into sequentially write *binarycode*.

Improved method: VLB + Separate storage of key and value

- Variable length binarycode

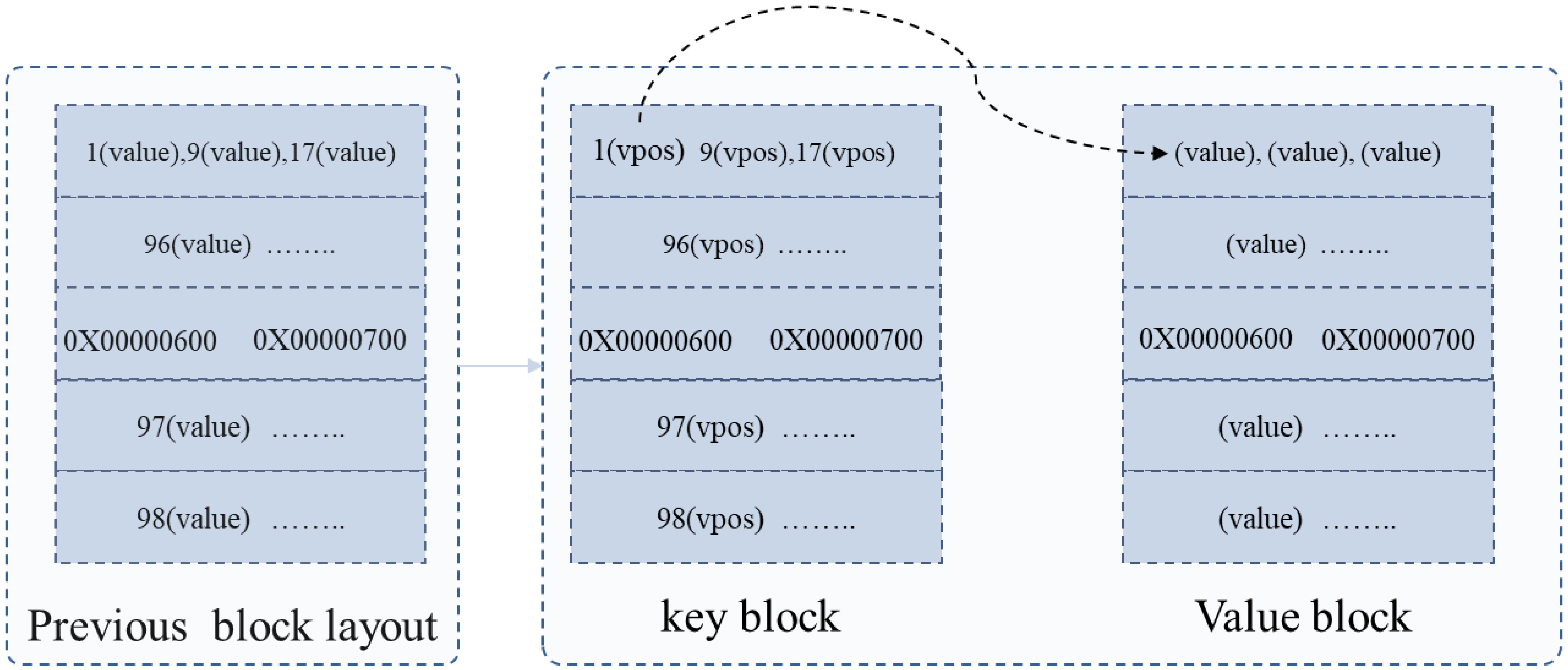
Using the highest bit as a marker:
'0' extends the data entry;
'1' ends the binary code.



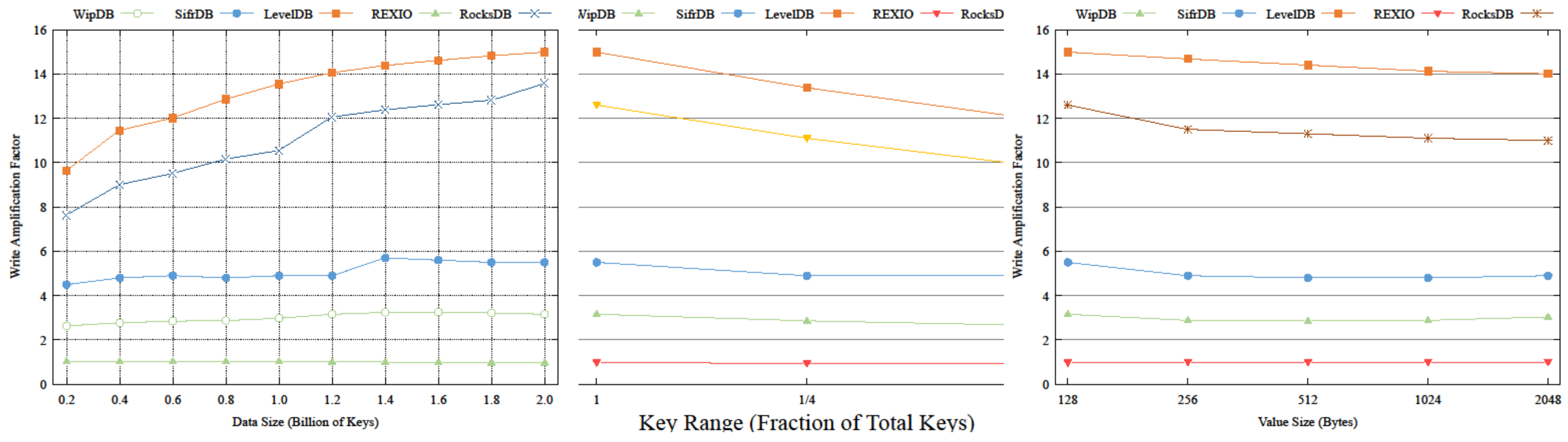
Fixed Length Binarycode (FLB) Variable Length Binarycode (VLB)

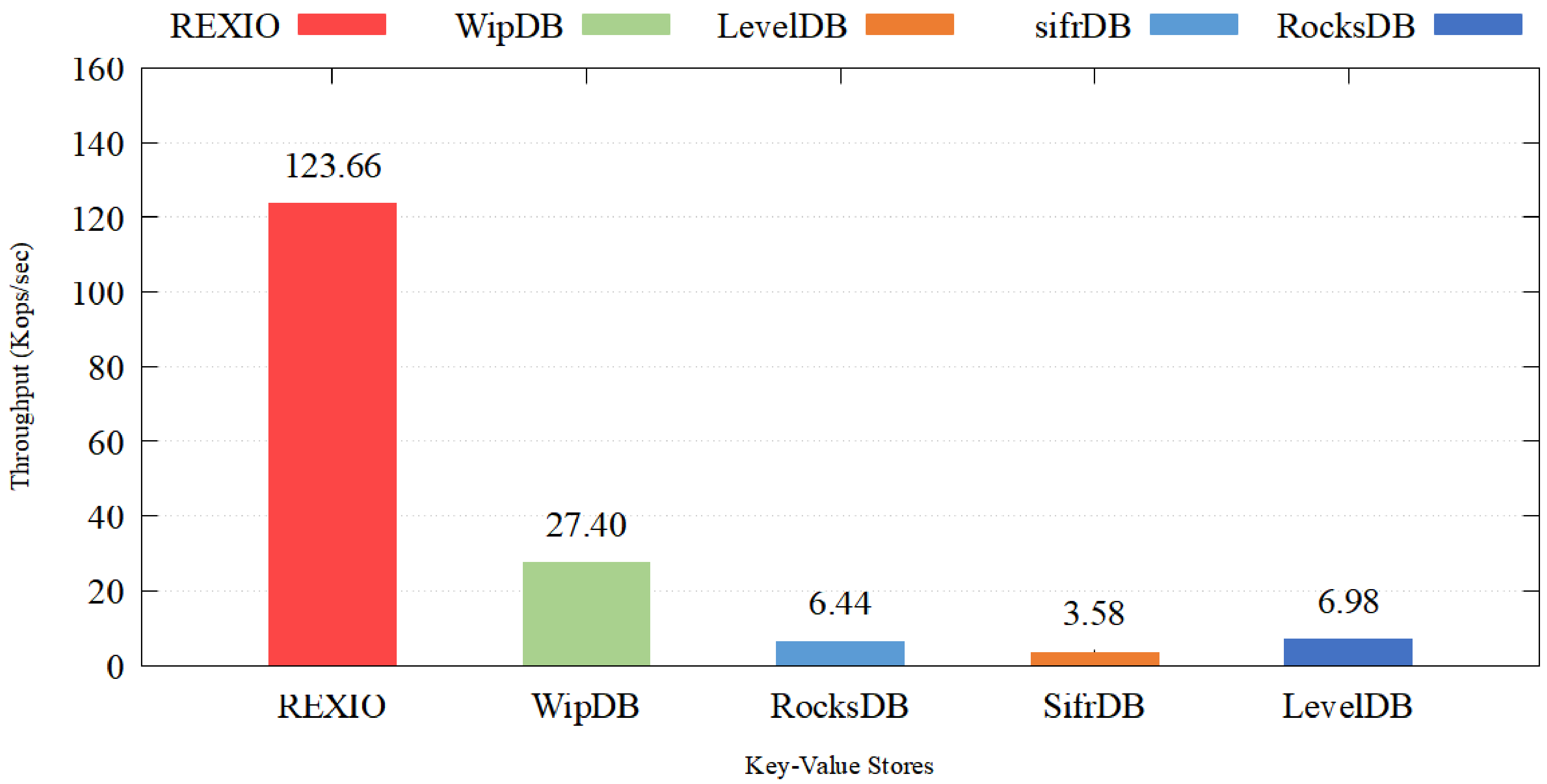
- Separate storage of key and value

Storing keys and values in different physical blocks in the SSD



04 Results



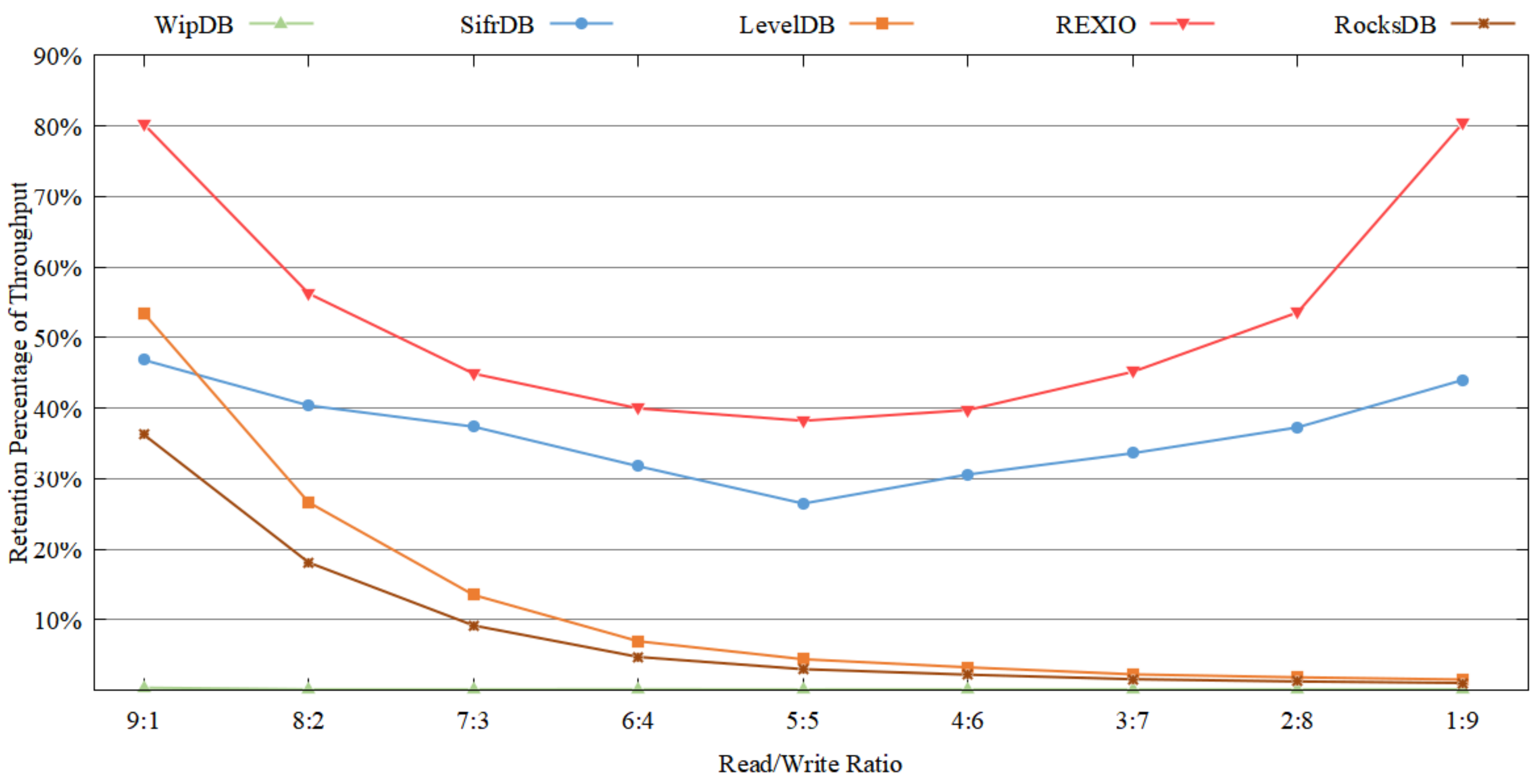


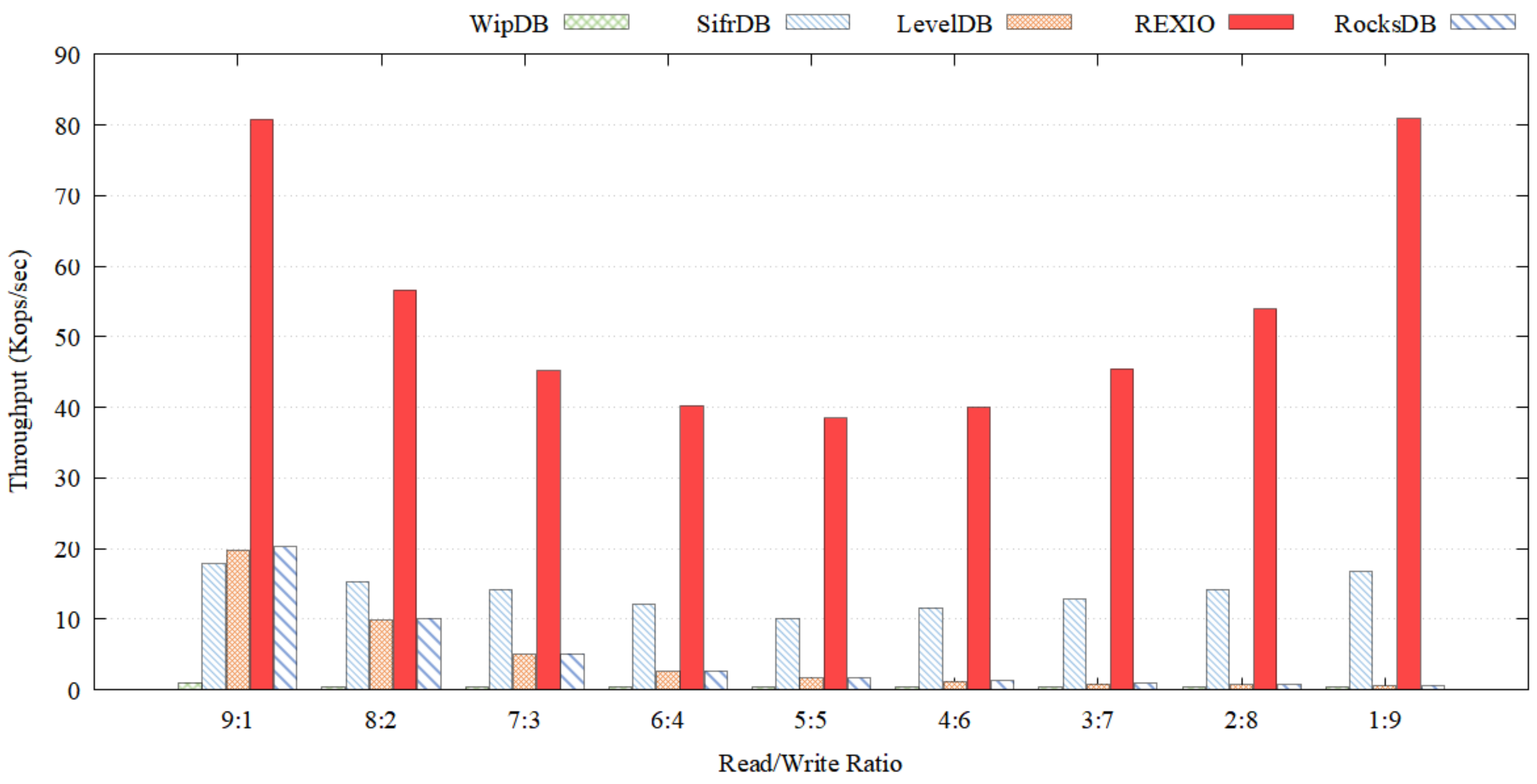
- **WAF Across Data Sizes**:
  - •**WipDB**: Peaks at 3.25 WAF around 1.6 billion KV pairs, then slightly decreases.
  - •**SifrDB**: WAF steadily rises, stabilizing at 5.70.
  - •**LevelDB & RocksDB**: WAF progressively increases to 14.99 and 12.60, respectively.
  - •**REXIO**: Maintains the lowest WAF, decreasing from 1.04 to 0.98 (68.3% lower than WipDB), due to separate key-value storage.

- **WAF Across Value Sizes**:
  - •**LevelDB & RocksDB**: WAF decreases as value size increases, reflecting their compaction strategy.
  - •**SifrDB**: Initially decreases but stabilizes around 4.9.
  - •**WipDB**: Drops to 2.858 at 512-byte values before increasing again, showing inefficiency with larger values.
  - •**REXIO**: Consistently maintains low WAF, decreasing slightly as value sizes increase.

- **WAF Across Key Ranges**:
  - •**WipDB**: WAF decreases from 3.16 to 2.67 as key range narrows, improving I/O efficiency.
  - •**SifrDB**: Slight decrease to 4.90, then stabilizes.
  - •**LevelDB & RocksDB**: WAF decreases to 12.07 and 9.95 with narrower key ranges.
  - •**REXIO**: Maintains consistently low WAF, showing robustness across varying key ranges.

- **Throughput Comparison**:
  - •**REXIO**: Achieves the highest throughput at 123.66 Kops/sec when writing a 268.2 GB dataset.
  - •**WipDB**: Manages 27.40 Kops/sec, significantly behind REXIO.
  - •**LevelDB & RocksDB**: Similar throughputs of 6.98 Kops/sec and 6.44 Kops/sec, respectively.
  - •**SifrDB**: Lowest throughput at 3.58 Kops/sec.





◆ **WipDB Throughput Retention**:
- Initial retention of 0.35% in read-intensive conditions, dropping significantly across all r/w ratios, maintaining just 0.15%.

◆ **SifrDB Throughput Retention**:
- Initial retention of 46%, dropping to 31% in mixed r/w operations, recovering to 49% as write operations increase.

◆ **LevelDB and RocksDB Throughput Retention**:

•LevelDB: Starts at 53%, declines, and stabilizes around 1.5%.
•RocksDB: Starts at 36%, eventually dropping to 1%.

◆ **REXIO Throughput Retention**:

•Drops to 40% in the most intensive 5:5 r/w condition, then increases gradually, reaching 80%.

◆ **Analysis**:

•Intensive read operations consume more time, impacting throughput retention.
•REXIO's decoupled design makes buffer size increases more effective compared to LSM-based KV stores.

◆ **Throughput Comparison**:

•REXIO starts with 80 Kops/sec in 1:9 r/w ratio and 81 Kops/sec in 9:1 w/r ratio.
•REXIO's absolute throughput is 3.4x higher than SifrDB at a 5:5 r/w ratio.

•**Conclusion**: REXIO's mechanisms show superior performance, especially with large-value writes.

中国科学院大学
University of Chinese Academy of Sciences

广东省智能科学与技术研究院
Guangdong Institute of Intelligence Science and Technology

Thank you for your listening!