TurtleDB: Highly Tunable, High-Performance Key/Value Storage

Tony Astolfi (tastolfi@mathworks.com)
MathWorks/Tufts University

Vidya Silai
MathWorks

Raja R. Sambasivan
Tufts University

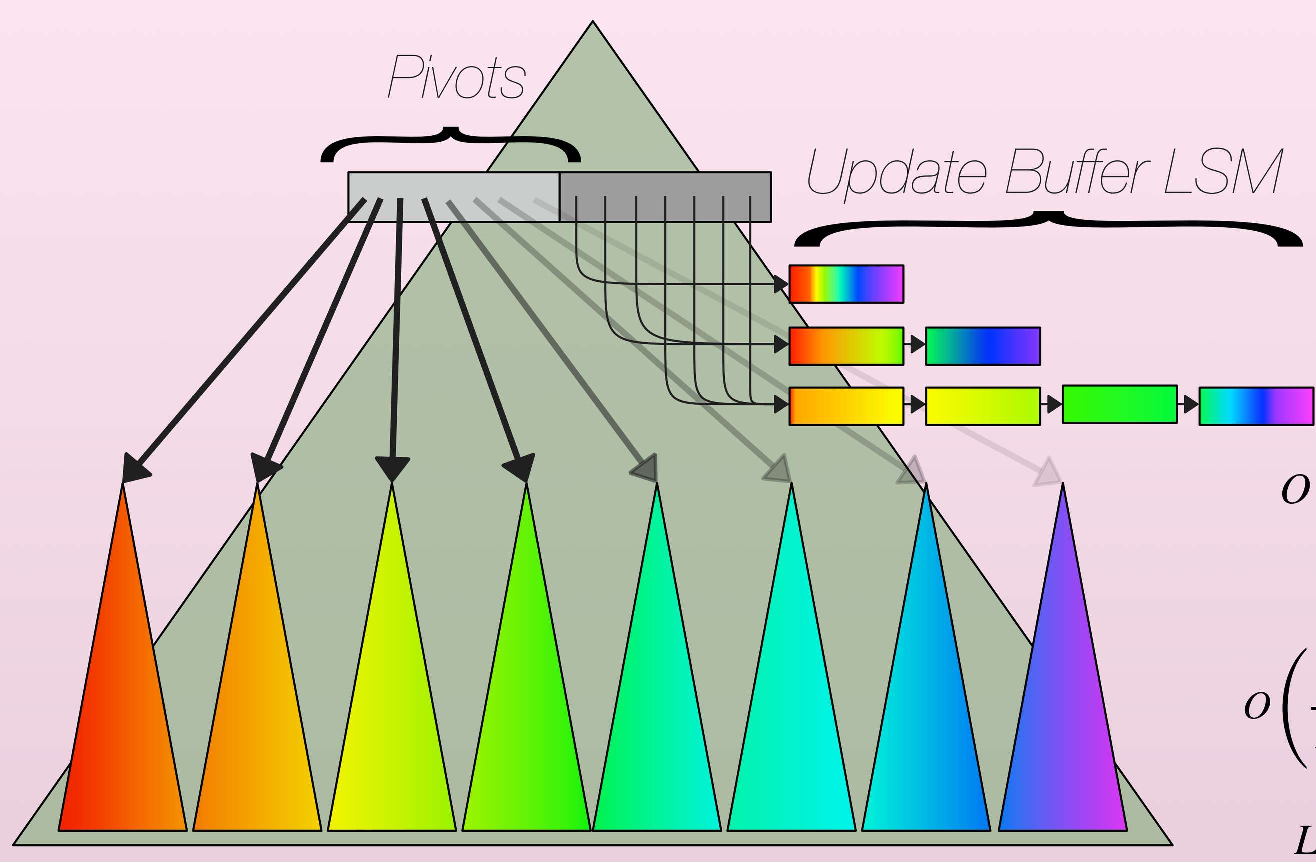
Johes Bater
Tufts University

The Challenge

Key/Value Storage Engines offer a generic, low-level abstraction for data persistence, but delivering on the promise of generality is often hindered by:

- *Data Structures that bake in application-specific trade-offs*
- *Complicated tuning knobs with limited dynamic range*

Turtle Trees: A novel Bε+Tree variant



Read Cost:

$$O\left(\alpha \cdot \log_2\left(\frac{N}{L}\right)\right)$$

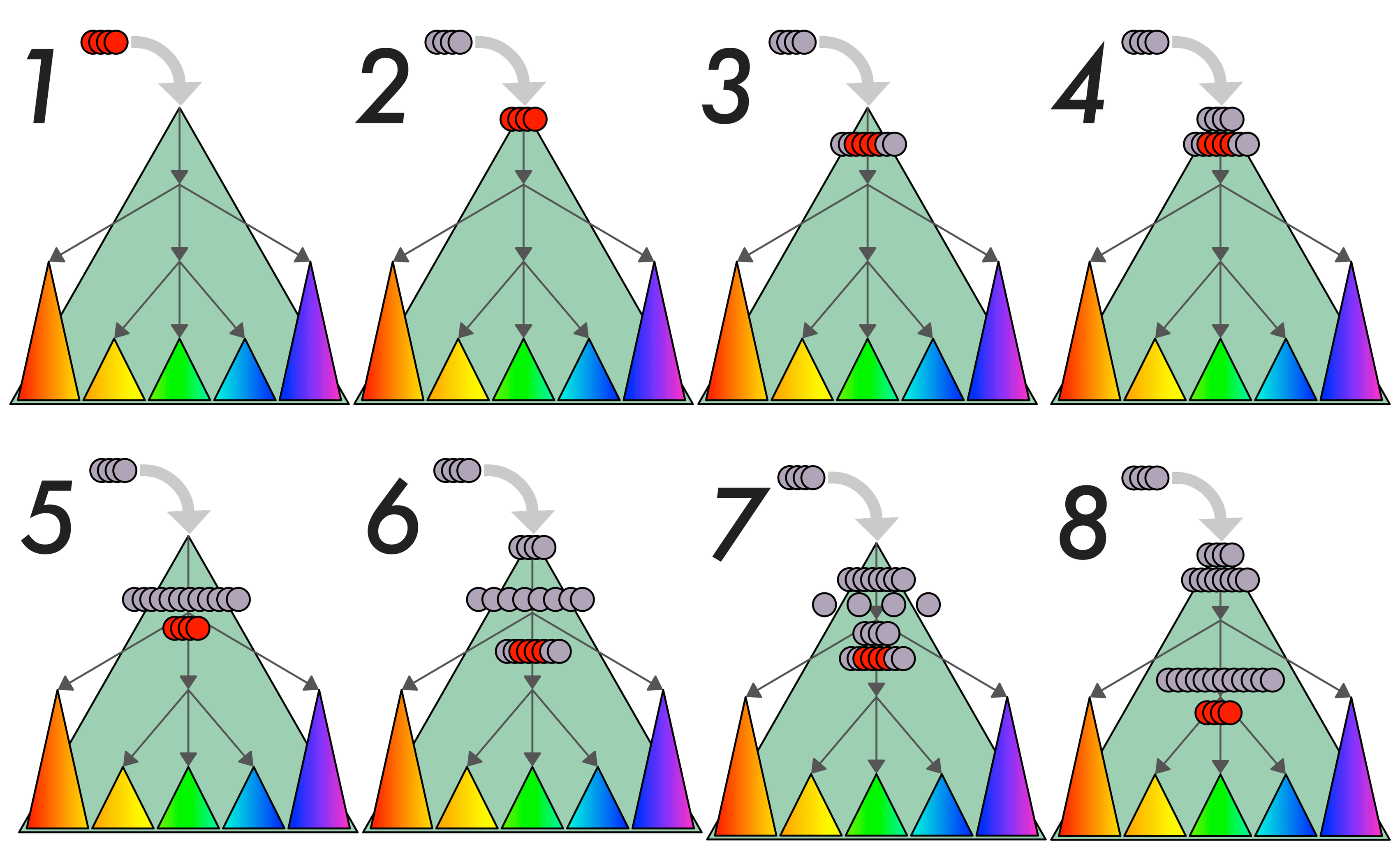
Write Cost:

$$O\left(\frac{1}{L} \cdot \log_2\left(\frac{N}{L \cdot \chi \cdot \alpha}\right)\right)$$

L = Leaf Page Size

Tuning Parameter: Checkpoint Distance (χ)

Key Idea: *Batch Updates / Compactions happen in-memory, up to limit χ, allowing updates to "skip" the first $\log_2(\chi)$ levels of the tree, reducing write amplification…*



Dynamic Trade-offs via Two Tuning Knobs

Checkpoint Distance χ

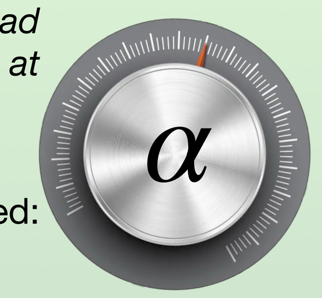
Apply multiple batches to the previous Turtle Tree checkpoint before flushing new/dirty pages (up to χ bytes)

- Decreases Write Cost by $O(log(\chi))$
- Spend Memory during ingest, not queries (like filters)
- **Does not affect read/space costs**

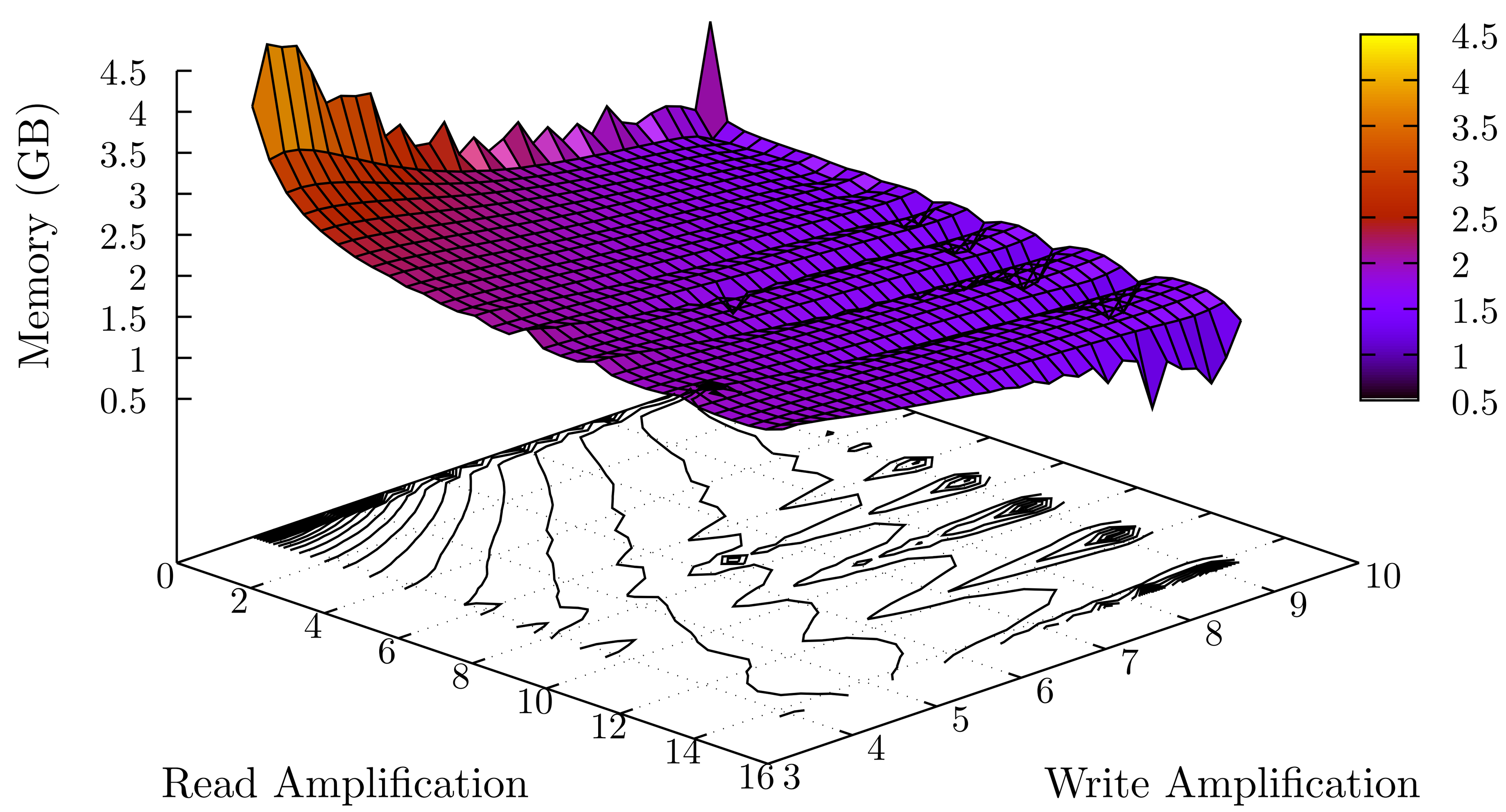
Filter False Positive Rate α

Increase bits per key to decrease read amplification; incurs memory cost at query-time, plus write cost at ingest

- Standard LSM technique
- Any AMQ Filter type can be used: Bloom, Cuckoo, Quotient, etc.
- $O(log(1/\alpha))$ bits per key required



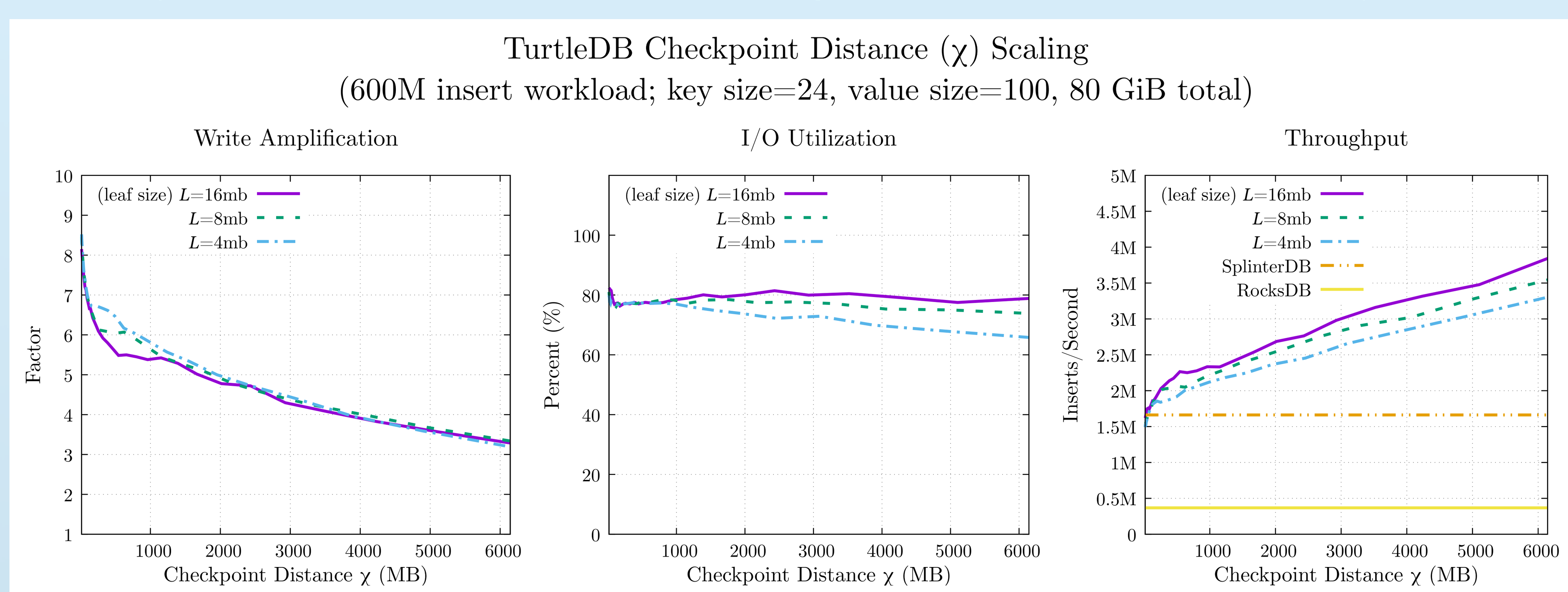
Read/Write/Memory Trade-off Space



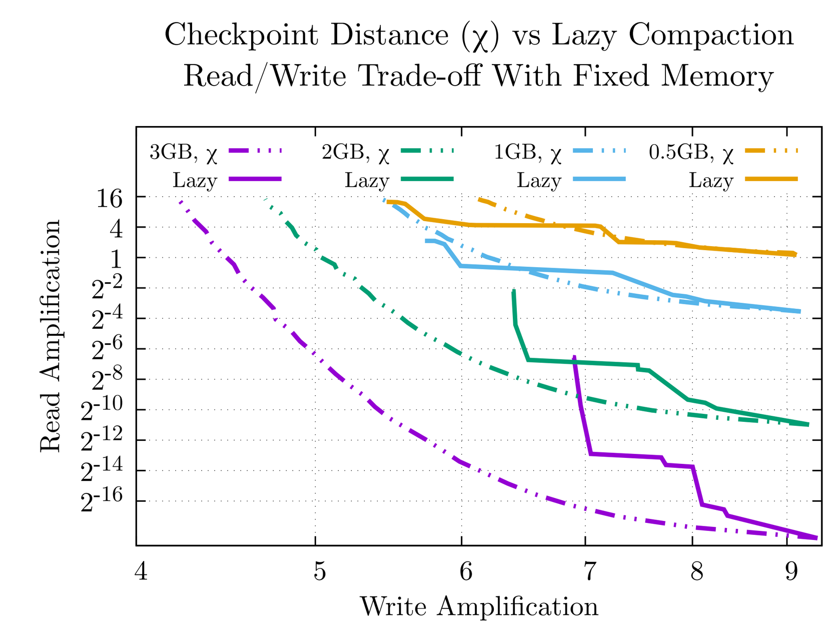
On-Demand Write Optimization

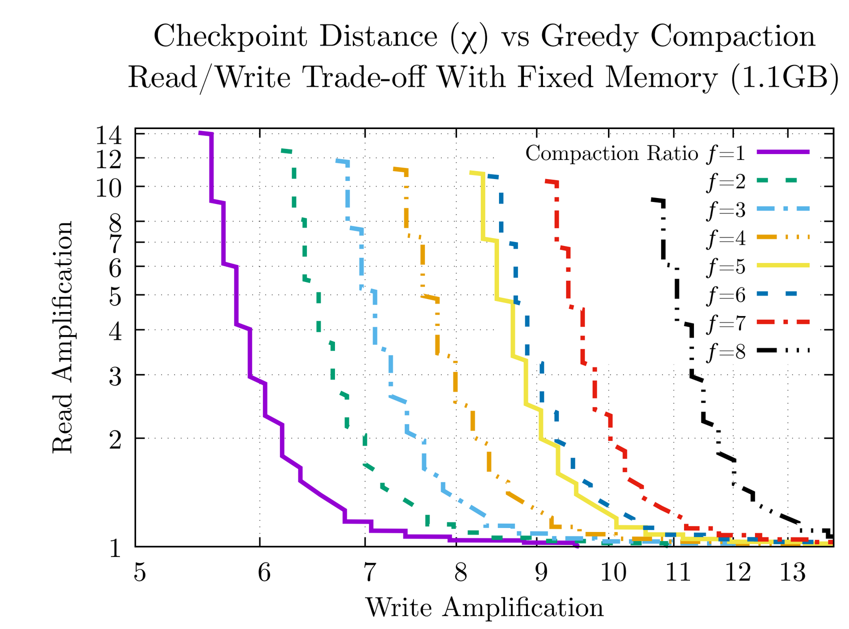
- *Size-Tiered LSM- and Bε-Trees optimize writes at the cost of extra memory for filters for the entire lifetime of the data (even after write workload finishes)*
- *Turtle Trees enable write optimization by increasing χ, spending memory only while updating*

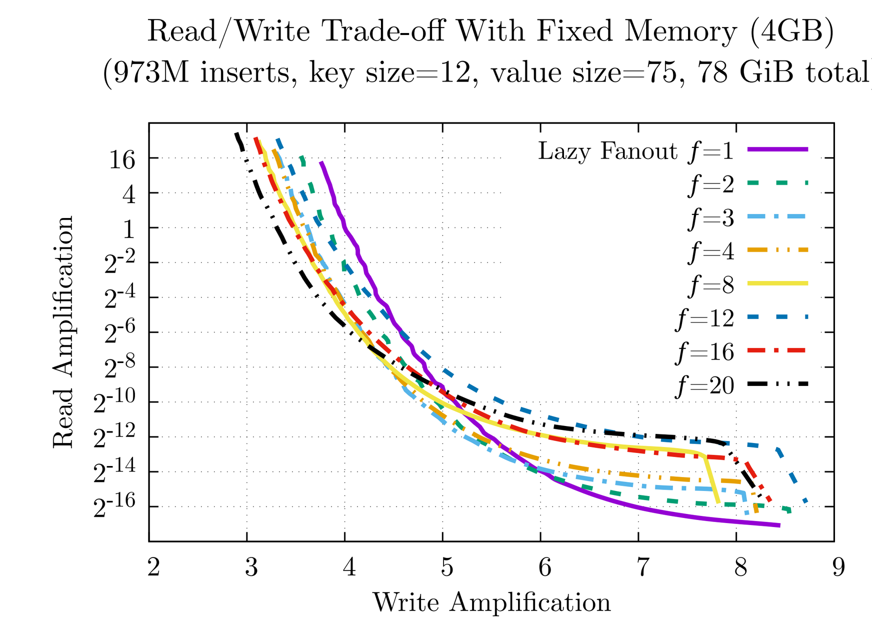
(Benchmarks run on AMD 7970x (32 core/64 thread) workstation with Linux 6.3, using Intel Optane 905p 1.5tb SSD; I/O utilization is percent of rated device bandwidth of 2.2GB/s; keys were inserted in uniform random order)



Comparing χ tuning to Greedy & Lazy Compaction Policies

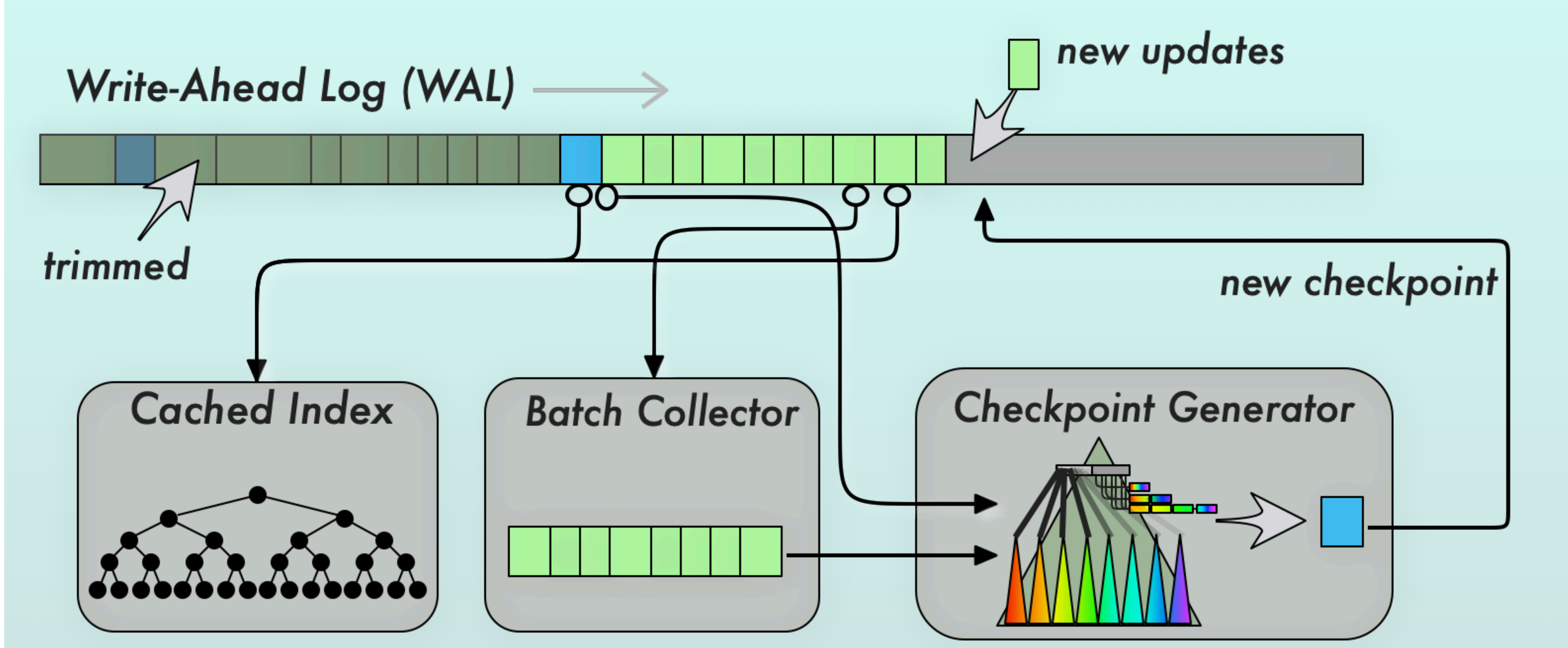






- Checkpoint Distance (χ) outperforms Greedy Compaction (level size ratio) tuning across entire dynamic range
- χ also outperforms Lazy Compaction at medium to high memory budgets, and allows on-demand write optimization
- When χ is combined with Lazy Compaction in Bε+-Trees, the dynamic range of Write Optimization can be extended

System Architecture



Benchmark Results: TurtleDB vs RocksDB

Average throughput of various workloads for multiple key/value sizes (6/100, 16/32, 32/512)