Modeling *Fast Delete* in Log-Structured Merge (LSM) Trees

Grace Sun[1,2], Venkat Subramanian[2], Subhadeep Sarkar[2], Manos Athanassoulis[2]

[1]Niskayuna High School, 1626 Balltown Rd, Niskayuna, NY 12309
[2]DiSC lab, Boston University, 111 Cummington Mall, Boston, MA 02215

Introduction

Background

Why use LSM trees?

Fast Ingestion | Competitive Reads | Good Space Utilization

Deletes in LSM trees

- Insert **tombstone** to "delete"
- Tombstones invalidate but do not remove entries
- No guarantees for deletion time
- Invalid entries increase **space amplification**, impairing performance
- ***Fast Delete* (FADE)** persists deletes by enforcing time bounds (*delete persistence thresholds*) on garbage collection (*compactions*)

Objective

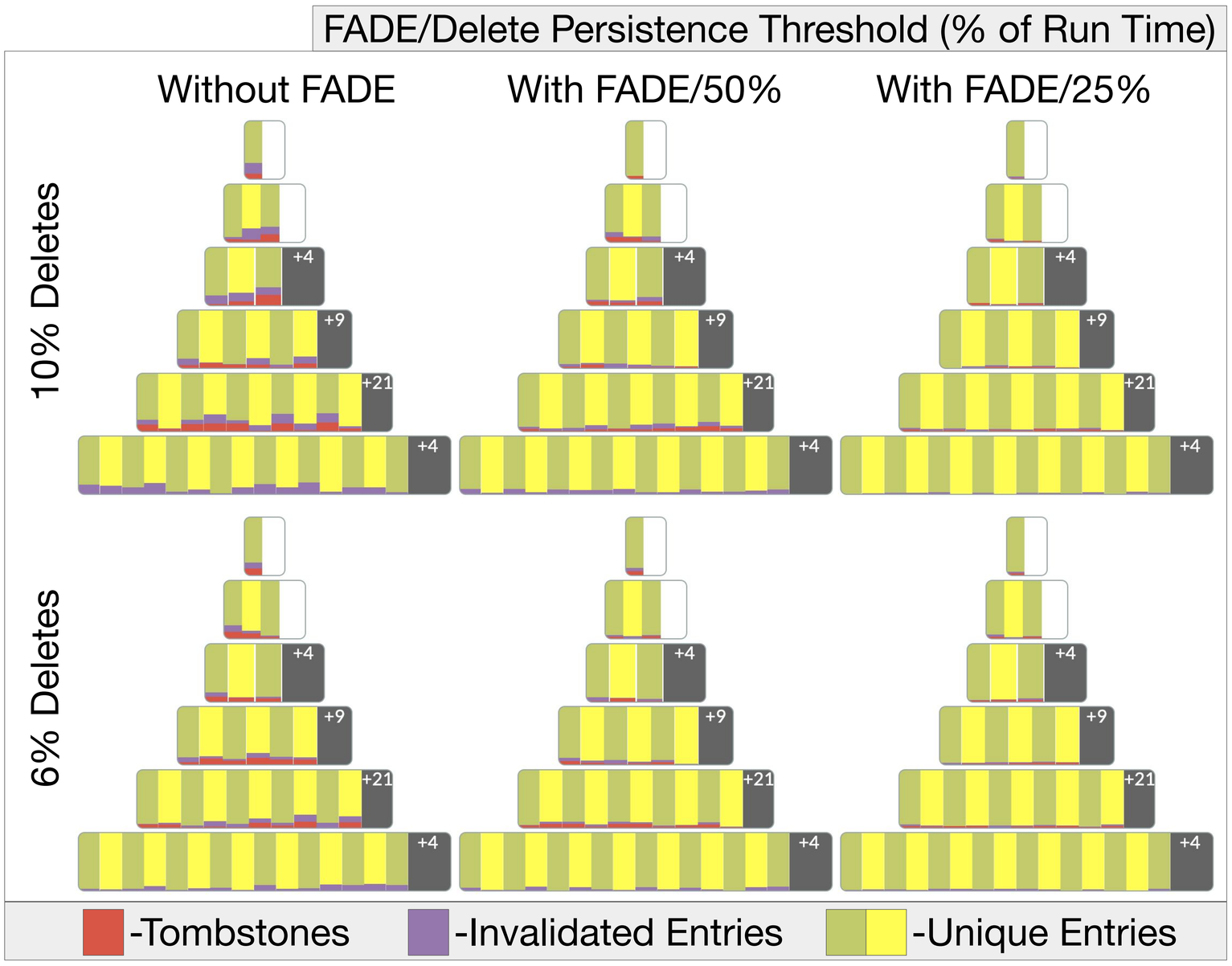
Build web interface visualizing FADE's performance implications as workload varies

Methods

- Build dynamic, file-specific tombstone visual (Fig. 1)
- Populate emulated LSM trees for six workload configurations
- Validate visualization by comparing with experimental results (Fig. 2)
- Parameters varied:
  - Use of FADE (Without FADE, With FADE/50%, With FADE/25%)
  - Proportion of deletes in workload (6%, 10%)

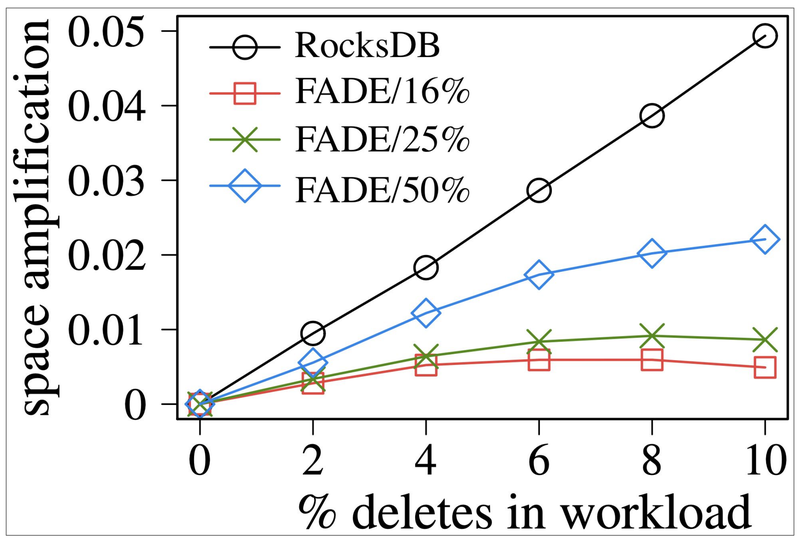
Results

Fig. 1: Tombstone density and distribution (LSM emulation)



Discussion

Fig. 2: Space amp. vs. % deletes [1]



Conclusions

- Without FADE, almost all tombstones remain in the LSM tree
- With FADE, tombstones are persisted, reducing space amplification especially when deletes increase in the workload
- Lower delete persistent thresholds persist more tombstones

Limitations

- Static inputs to vary in future:
  - Presence of updates
  - Support for different key distributions (currently support uniform distribution only)

Acknowledgements

Thank you to Prof. Athanassoulis and Dr. Sarkar for their incredible mentorship, Venkat Subramanian for collaborating on the visualization, the DiSC lab for their steadfast support, and the BU RISE program for giving me this wonderful opportunity.

References

[1] S. Sarkar, T. I. Papon, D. Staratzis, M. Athanassoulis. Lethe: A Tunable Delete-Aware LSM Engine. ACM SIGMOD, 2020.
[2] S. Sarkar, K. Chen, Z. Zhu, M. Athanassoulis. Compactionary: A Dictionary for LSM Compactions. ACM SIGMOD, 2022.
[3] S. Sarkar, D. Staratzis, Z. Zhu, M. Athanassoulis. Constructing and Analyzing the LSM Compaction Design Space. VLDB, 2021.
[4] S. Sarkar, M. Athanassoulis. Dissecting, Designing, and Optimizing LSM- based Data Stores. ACM SIGMOD, 2022.