Diversity of LSM tree shapes

Mark Callaghan
Facebook
mcallaghan@fb.com

ABSTRACT

There are two popular approaches for persisting writes in a log-structured merge tree[1] – leveled and tiered compaction. Both are constraints on the shape of an LSM tree. There exist LSM tree shapes that provide better efficiency for some workloads but are not allowed by leveled or tiered compaction. This talk explains runs-per-level compaction which supports leveled, tiered and a hybrid of leveled and tiered. The work is part of a larger effort to support run-time optimization of LSM trees.

1. Overview

Access method efficiency has many dimensions. The RUM Conjecture[2] explains three of them – read, write and space. Leveled and tiered compaction cover different regions of the three-dimensional efficiency space. While those regions are sufficient for many workloads, there are interesting regions of the efficiency space that cannot be reached with either leveled or tiered.

Leveled and tiered compaction impose constraints on the shape of an LSM tree that determine the work that must be done for point reads, range reads, inserts and deletes. Leveled compaction is best for space efficiency and tiered compaction is best for write efficiency. But being best in one dimension comes at a cost in other dimensions and for some workloads that cost is too much.

As part of a project to make the LSM tree shape adaptive at run-time a new compaction algorithm, runs-per-level, has been created. The runs-per-level algorithm is a hybrid of tiered and leveled compaction. The smallest N levels of the tree use tiered while the remaining levels use leveled or a variant of leveled. When N is 0 then it implements leveled compaction. When N is the number of levels in the LSM tree then it implements tiered compaction. Otherwise it implements a hybrid of tiered and leveled. It can be more efficient than leveled and tiered for some workloads.

2. Related Work

Dostoevsky[3] is a new algorithm that is a variant of leveled compaction. Compared to leveled it can trade more read amplification for less write amplification to improve performance for some workloads. The paper also provides a complete performance model to understand read, write and space efficiency with an LSM.

Data Calculator[4] enumerates the search space for access methods in terms of read, write and space efficiency. This framework makes it possible to compare existing access methods and discover new ones.

3. REFERENCES

[1] O'Neil, P., Cheng, E., Gawlick, D., O'Neil, E. The log-structured merge-tree (LSM-tree). *Acta Informatica*. 33, 4 (June 1996), 351-385.

[2] Athanassoulis, M., Kester, M., Maas, L., Stoica, R., Idreos, S., Ailamaki, A., Callaghan, M. Designing Access Methods: The RUM Conjecture. In *Proceedings of the International Conference on Extending Database Technology*, 2016.

[3] Dayan, N., Idreos, S. Dostoevsky: Better Space-Time Trade-Offs for LSM-Tree Based Key-Value Stores via Adaptive Removal of Superfluous Merging. In *ACM SIGMOD International Conference on Management of Data*, 2018.

[4] Idreos, S., Zoumpatianos, K., Hentschel, B., Kester, M., Guo, D. The Data Calculator: Data Structure Design and Cost Synthesis from First Principles, and Learned Cost Models, In *ACM SIGMOD International Conference on Management of Data*, 2018.