Introducing LSM-tree into PostgreSQL, making it as a data gobbler

By Shichao Jin

From https://zhuanlan.zhihu.com/p/56259025

PGCon 2020

VidarDB
Simplify Your Backend

About Me

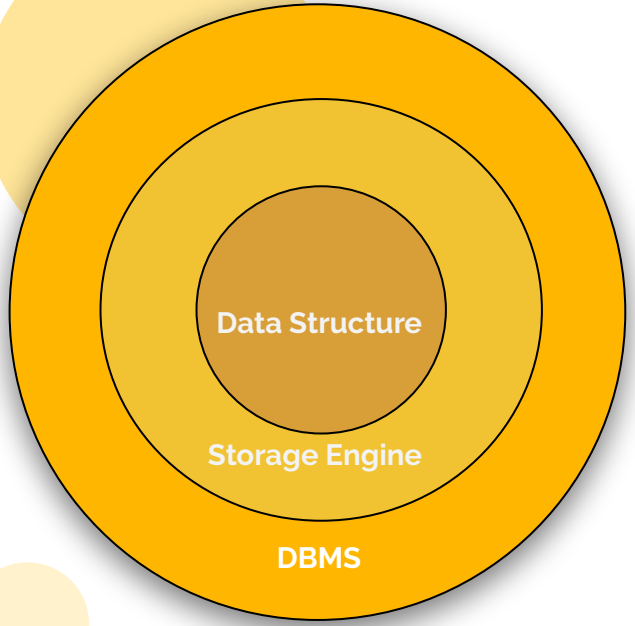
- 8 years experience in DBMS R&D
- Redshift AWS and Facebook RocksDB
- Ex-PhD student at UWaterloo
- Fan of PostgreSQL
- Founder of VidarDB

Outline

- Experiment & Analysis
- LSM-tree Introduction
- Our Implementation

Background

- Storage engine accounts for a big chunk of the performance
- What is “storage engine”? incarnation of a data structure



Background

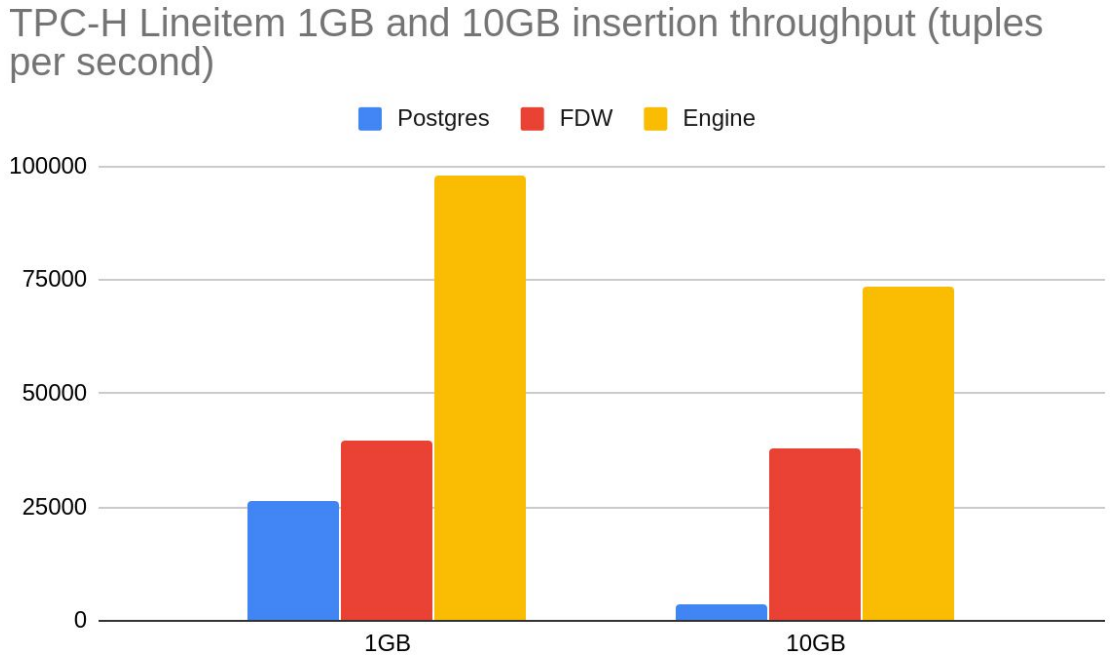
- B+tree/B-tree: InnoDB, BerkeleyDB, WiredTiger
- LSM-tree: LevelDB, **RocksDB**

Experiment Setting

Hardware:
CPU: Xeon E3-1265L, 2.5GHz, 4 cores
Disk: 7200 RPM
RAM: 16GB DRR3

Software:
TPC-H 10GB, **shuffled**!
Ubuntu 18.04, PostgreSQL11.6, RocksDB 6.2, default setting
libpqxx

Experiment Results



Why huge difference?

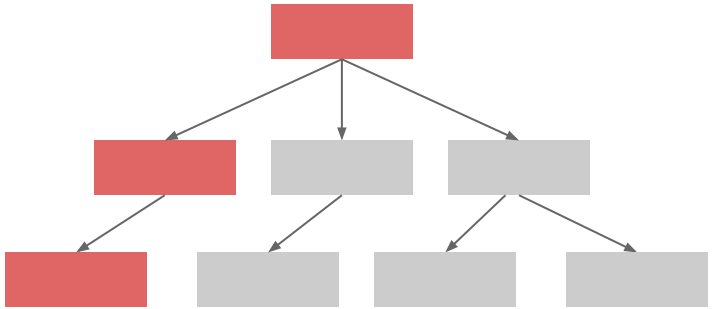
Data structure!

PostgreSQL uses **B+tree**!

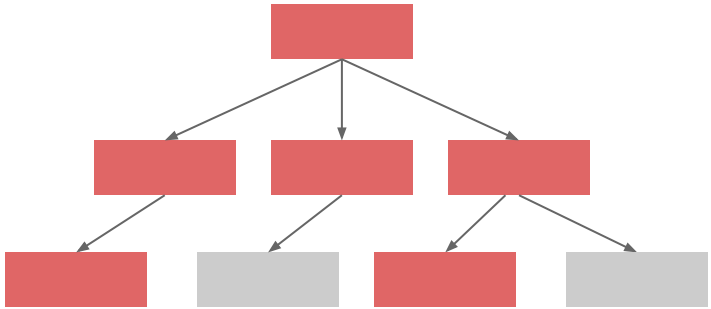
So what's wrong with B+tree?

B+tree Reading Pattern

Rare case: data come in order



Common case: data come in **random** order



Red rectangle means disk page cached!

Problem with B+tree

I only talk about the insertion case here, but the root cause comes from the reading pattern of B+tree. Every insertion of B+tree needs to trigger **random reading** to locate the exact position of B+tree.

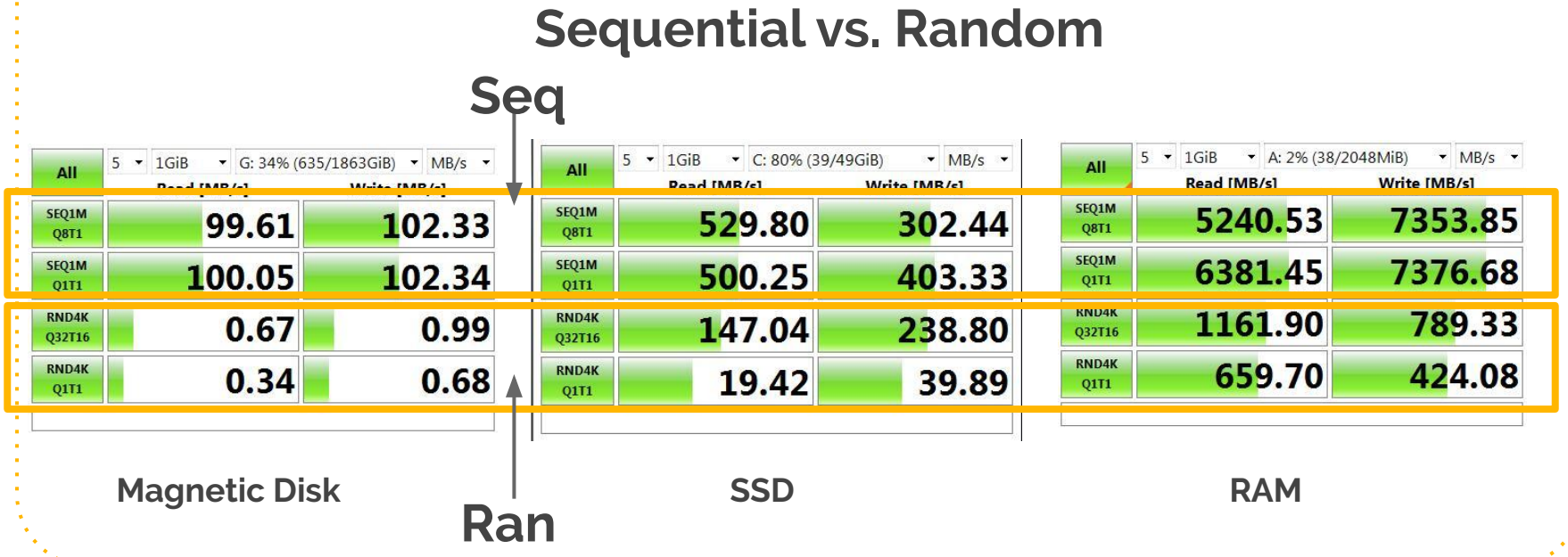
Note: Reading uses **cache**!

Problem with B+tree (Cont.)

Data is large, and usually come not in the order of primary key, the previous cached disk blocks are useless!

B+tree insertion causes too many disk access!

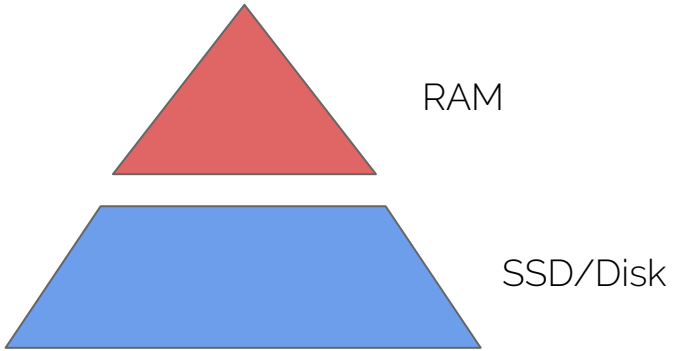
Disk vs. SSD vs. RAM in my laptop



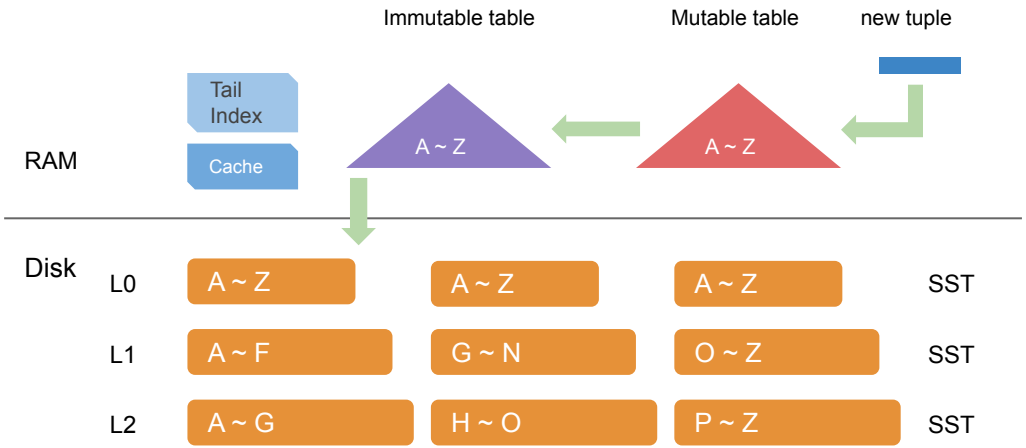
Solution

No touch of disk (excluding WAL) when insertion!
Tiered design.

LSM-tree (Log-structured merge-tree) comes to rescue!

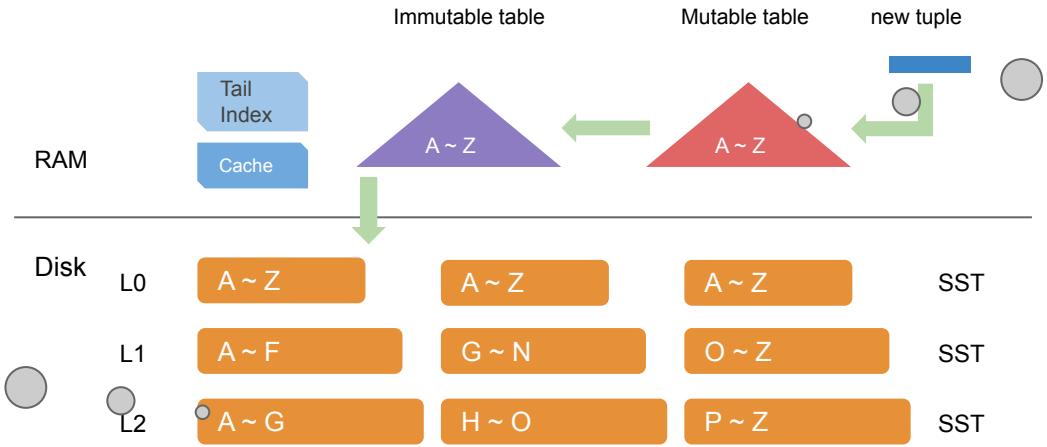


LSM-tree



SST: Sorted String Table

LSM-tree

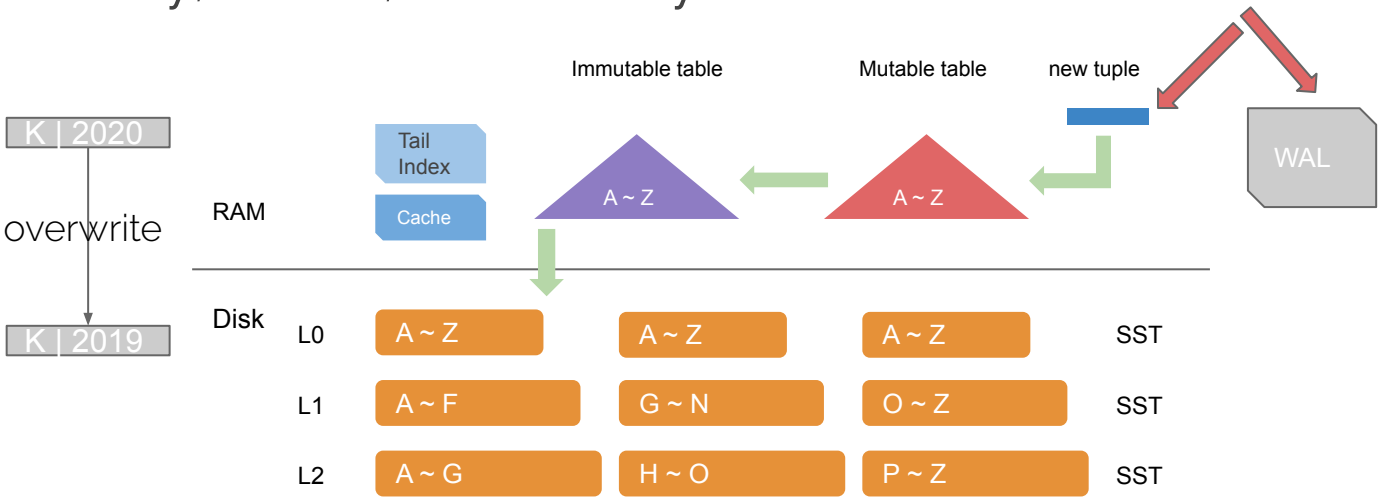


Skiplist
O(logN)

Sorted
kv pairs

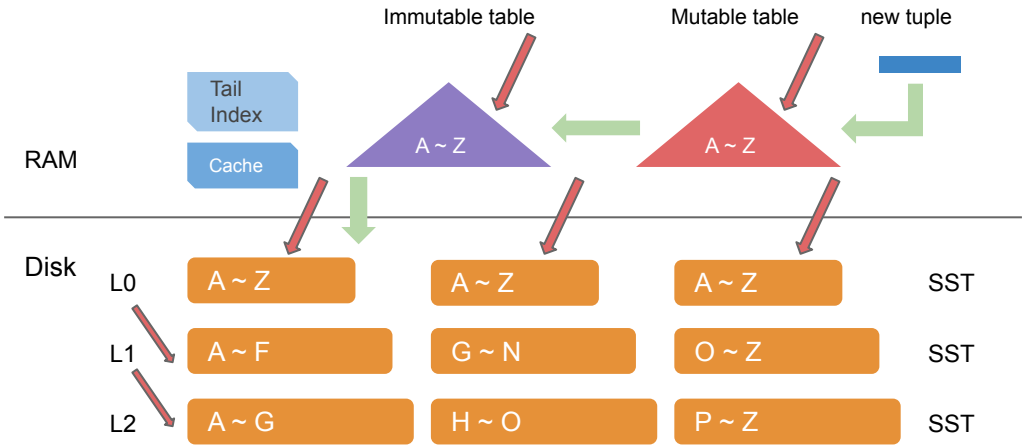
Basic Operations

Put(key, value), Delete(key)



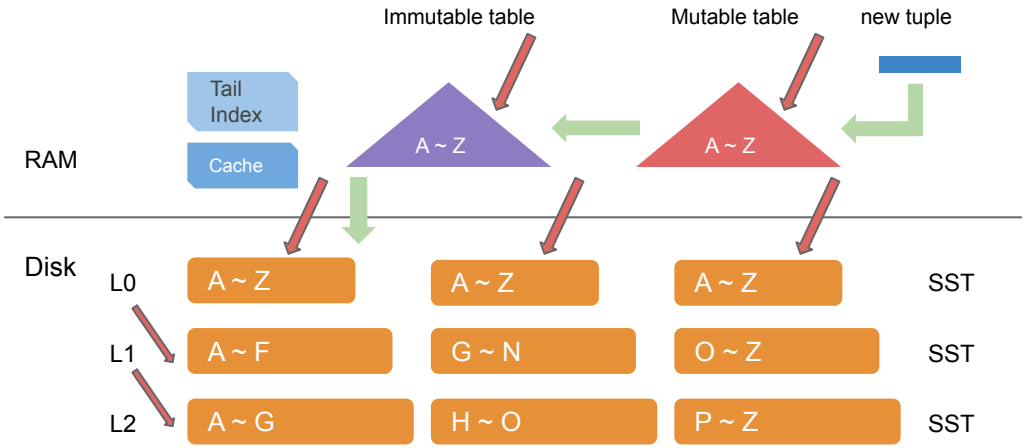
Basic Operations

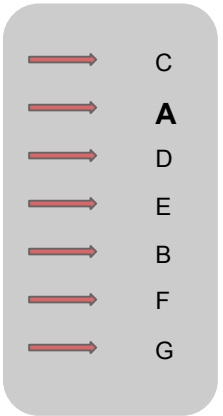
Get(Key), return value



Basic Operations

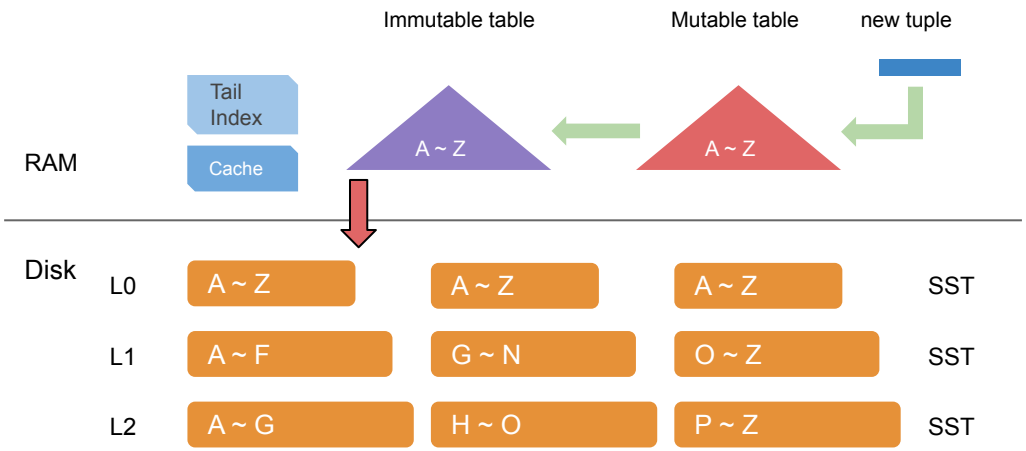
Iterator(): seek & next





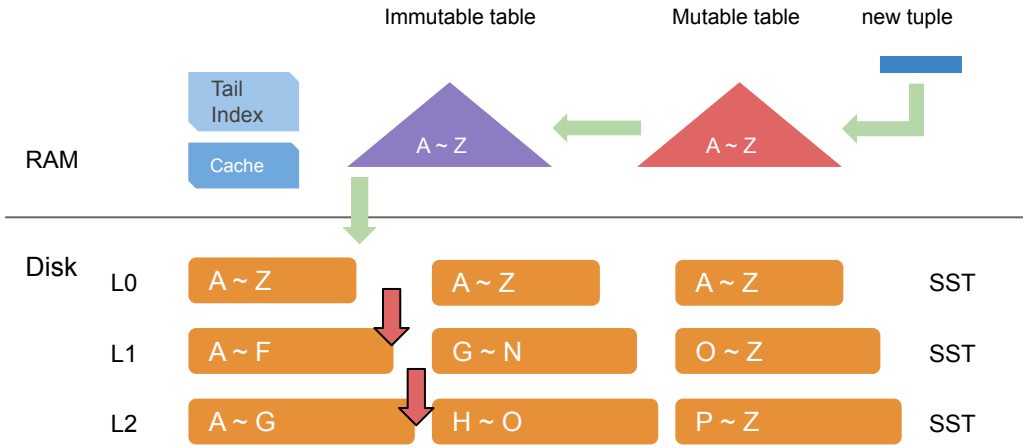
Advanced Operations

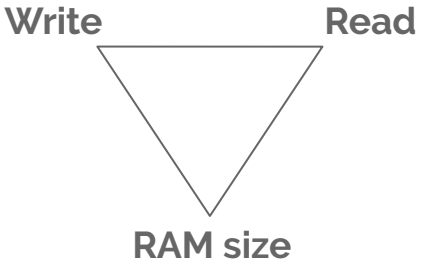
Flush(): RAM→ L0, usually by the background thread



Advanced Operations

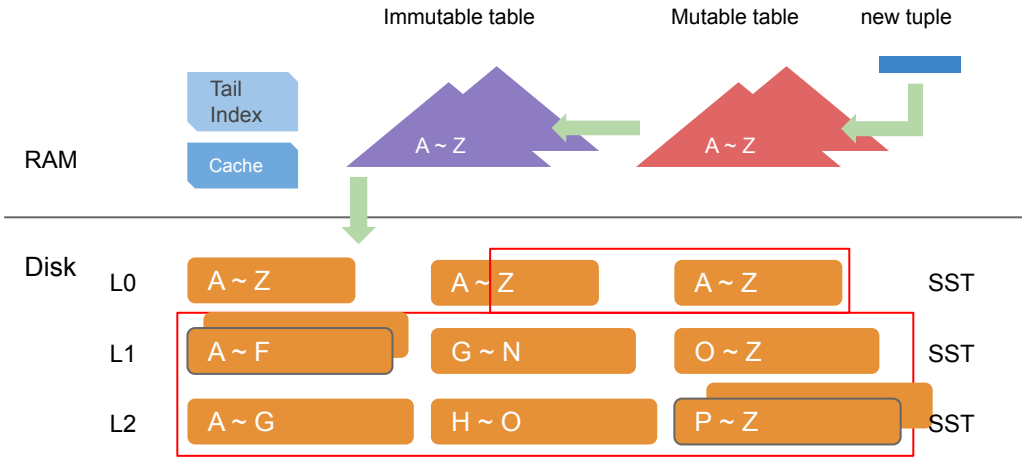
Compact(k1, k2): L0→ L1, L1→ L2, usually called by the background thread

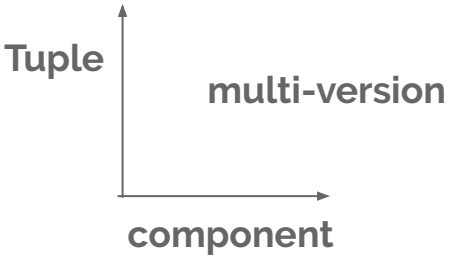




Advanced Operations

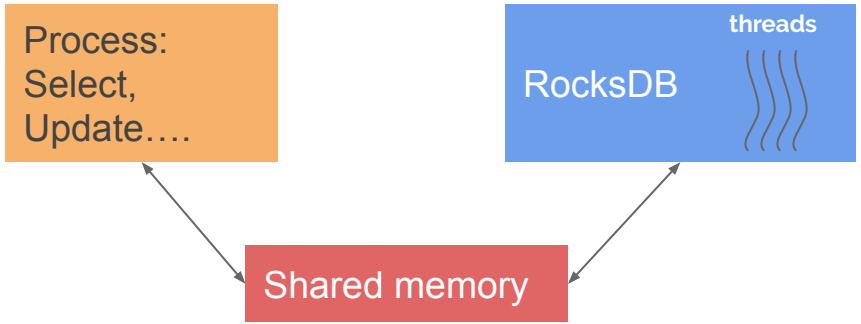
Snapshot(seq): sequence number, which is always increasing





Implementation

- Foreign data wrapper (FDW)
- RocksDB is based on the multi-thread model
- Shared memory



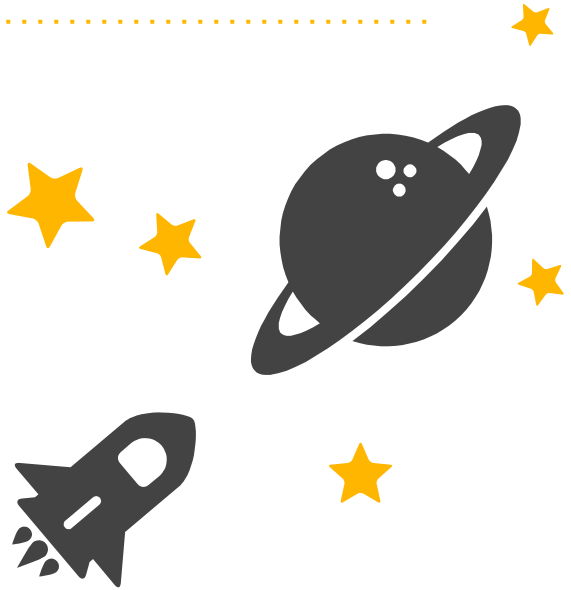
More To Do

- Engine recognizes the data types of PostgreSQL
- Transaction?
- Migrate to pluggable storage engine, table AM

Motivation of designing a new data structure

1. random R/W of RAM is **2000X** faster than magnetic disk, but **120X** more expensive
2. Ratio of RAM/disk size is **40~200X** larger than 1970s, when B+tree is designed
3. New storage choice, NVRAM
4. Requires expertise to choose different engines, data systems

Thanks!



Shichao@vidardb.com

Twitter: @jsc0218

github.com/vidardb/vidardb

github.com/vidardb/PostgresForeignDataWrapper