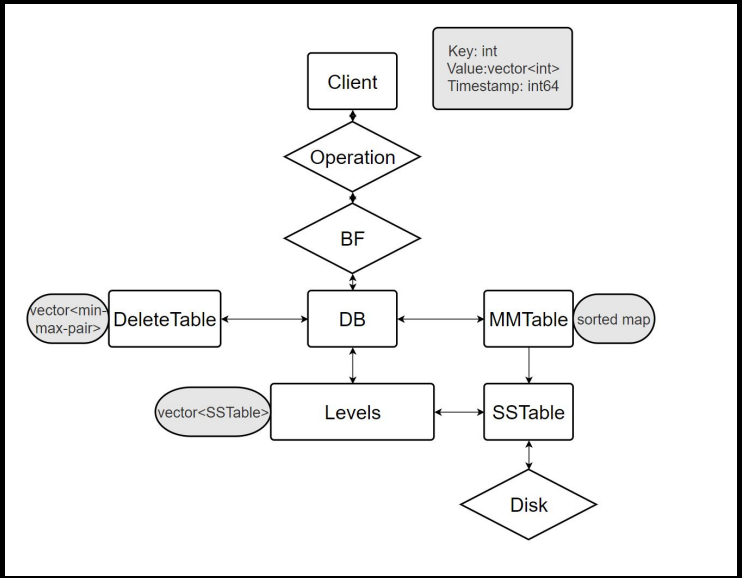
Implementation of LSM

By Hantian (Alan) Liu, Joseph Mitchell, Junchen Liu

Introduction

- Structure
- APIs
- Options

Structure


Key: int
Value:vector<int>
Timestamp: int64
Client
Operation
BF
vector<min-max-pair>
DeleteTable
DB
MMTable
sorted map
vector<SSTable>
Levels
SSTable
Disk

APIs

- Get
- Put
- Scan
- Range Scan
- Delete
- Range Delete

Options

```
namespace Option {
    const bool LEVELING = true;
    const uint64_t MEM_SPACE = (uint64_t) 2 * 1024 * 1024;
    const uint64_t NZ_NUM = 3;
    const uint64_t Z_SPACE = (uint64_t) 8 * 1024 * 1024;
    const uint64_t NZ_SPACES[] = {
        (uint64_t) 32 * 1024 * 1024,
        (uint64_t) 128 * 1024 * 1024,
        UINT64_MAX
    };
    const char *const Z_NAME = "/L0";
    const char *const NZ_NAMES[] = {"/L1", "/L2", "/L3"};
}
```

Challenges

- Range Delete
- Scan
- Persistent(insert/visible/timestamp/index/metadata)

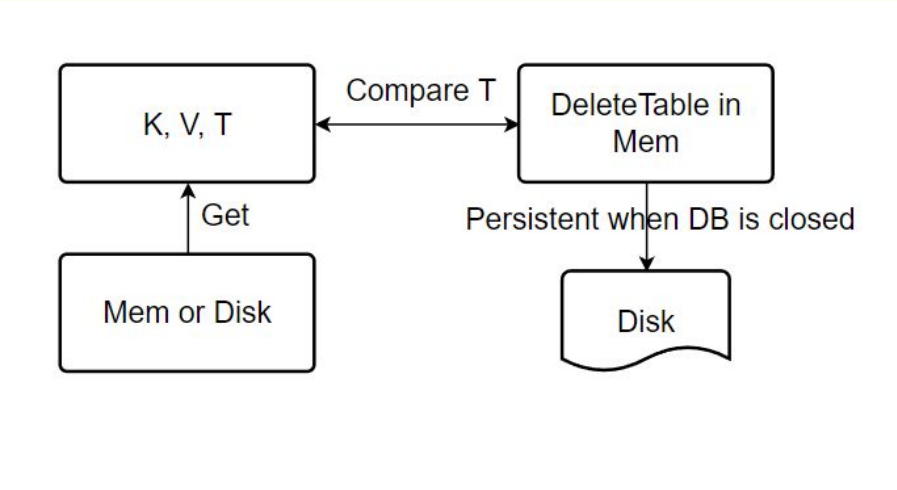
RangeDelete(int min_key, int max_key)

Old Approach:
- Insert tomestones one by one
- Disadvantages: Users Range Delete 1 - 1M

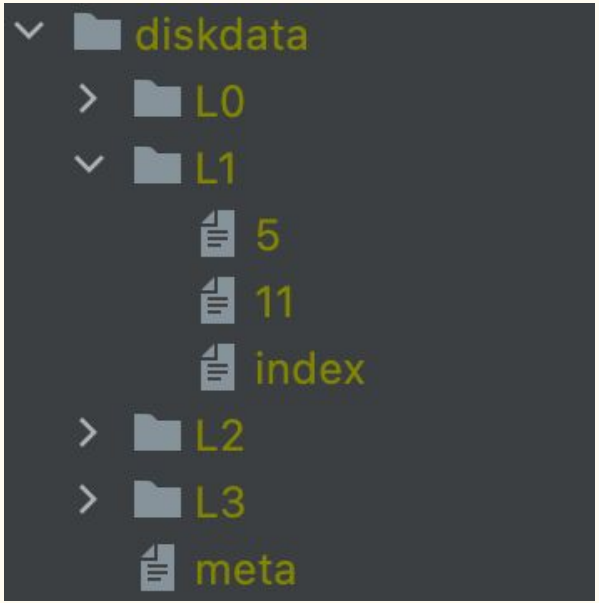
We Use:
- Insert a interval with a timestamp into a vector
- Compair

Problem:
- Scan the vector when GET

Possible Improvement:
Update the Delete Vector



Persistent



Goal:

- Manage the data storage
- Persistent the database when closed

Index in each Level:

- Byte count
- Id of each table in this Level

Metadata of DB:

- No of current SSTable to avoid overlap
- Configuration of current DB

Experiments

- BloomFilter
- Level Threshold
- Compact Strategy

Experiment Parameters

Data

10k entries

5 dimension

Workload

10k operations

10k key range

Default Setting

Memory Table size 1k

Level 0 size 4k

Level 1 size 16k

Bloom Filter Test

BF size = 1024

Runtime 419s

BF size = 10000

Runtime 312s

Level Size

Memory Table size 1k

Level 0 size 4k

Level 1 size 16k

Runtime 410s

Memory Table size 2k

Level 0 size 8k

Runtime 403s

Compact Strategy

Tiering

Runtime 419s

Leveling

Runtime 410s

Multiply dataset size and level size by 100

Runtime more than 1h

Runtime 15 min

Conclusion

There’s a significant bottleneck in the system we need to fix

Large bloom filter can largely increase the performance

Leveling search through one run in one level and performs better when data size is large