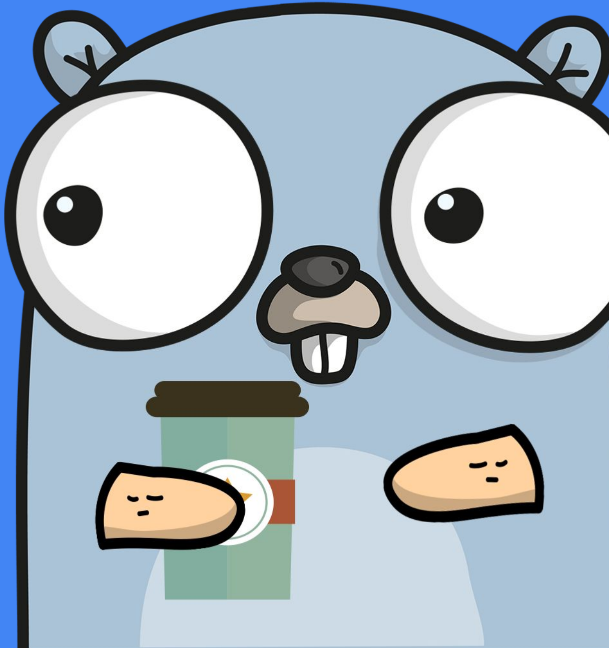
Intro To Pebble

https://cockroachdb.slack.com/archives/CQVRDNE23

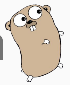
https://github.com/cockroachdb/pebble

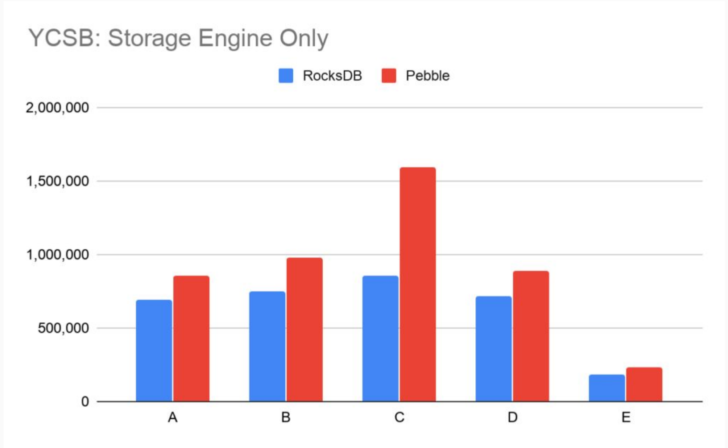


What

- Key-value storage written in by CockroachLabs
- Inspired by RocksDB, LevelDB, used in CockroachDB
- Has RocksDB API

Why

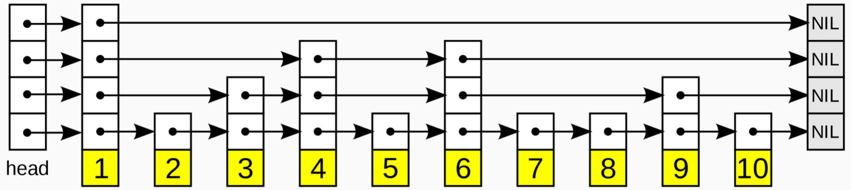
- Active community
- Written in 
- Efficient

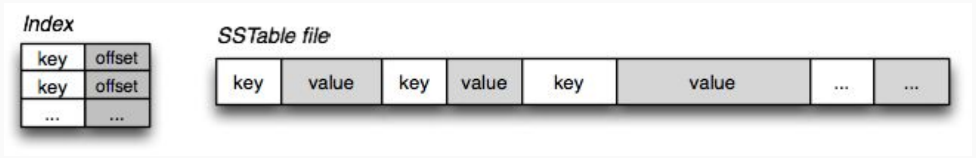


https://cockroachlabs.com/blog/pebble-rocksdb-kv-store/#new-storage-engine-performance

How

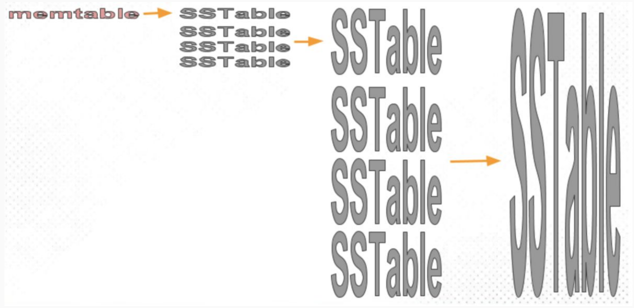
- Log Structured Merge Tree
- Memtables (skiplists) and SSTables
- Snappy compression library





How - LSM & Compaction

- Memtables flushed to SSTables
- SSTables (= runs) ordered in levels
- Each run in a given level has similar size & is ordered
- When level is full, runs are merged, sorted, and moved to upper level



```go
package main

import (
    "fmt"
    "log"

    "github.com/cockroachdb/pebble"
)

func main() {
    db, err := pebble.Open("/path/to/db", &pebble.Options{})
    if err != nil {
        log.Fatal(err)
    }
    key := []byte("hello")
    if err := db.Set(key, []byte("world"), pebble.Sync); err != nil {
        log.Fatal(err)
    }
    value, closer, err := db.Get(key)
    if err != nil {
        log.Fatal(err)
    }
    fmt.Printf("%s %s\n", key, value)
    if err := closer.Close(); err != nil {
        log.Fatal(err)
    }
    if err := db.Close(); err != nil {
        log.Fatal(err)
    }
    // NOTE: See regatta/pebble/pebble.go:OpenDB for real-world usage.
}
```

Features

- Level-based compaction
- Indexed batches
- Range search
- Prefix search
- Range deletion (with range deletion tombstones),
- Bloom filters
- Snapshots...

Features - Range Search/Delete

- Query a range of ordered key-value pairs
- `GET/DELETE [begin, end)`

```
[
  {"key": "bar",  "value": "1"},
  {"key": "baz",  "value": "2"},
  {"key": "foo",  "value": "3"},
  {"key": "quux", "value": "4"},
  {"key": "qux",  "value": "5"}
]
```

$\xrightarrow{\texttt{GET [a, quux)}}$

```
[
  {"key": "bar",  "value": "1"},
  {"key": "baz",  "value": "2"},
  {"key": "foo",  "value": "3"}
]
```

Features - Range Search/Delete - How

```
options := &pebble.IterOptions{
    LowerBound: lowerBound,
    UpperBound: upperBound,
}
iter := db.NewIter(options)

for iter.First(); iter.Valid(); iter.Next() {
     fmt.Printf(
        "key: %+v, value: %+v\n",
        iter.Key(),
        iter.Value(),
    )
}
```

// See regatta/storage/table/sm.go:iterator for real-world usage.

Features - Prefix Search

- Special case of range search
- When searching for prefix `prefix`, use range search `GET [prefix, prefix+1)`

```
[
  {"key": "bar",  "value": "1"},
  {"key": "baz",  "value": "2"},
  {"key": "foo",  "value": "3"},
  {"key": "quux", "value": "4"},
  {"key": "qux",  "value": "5"}
]
```

$$\xrightarrow{\texttt{GET [ba, bb)}}$$

```
[
  {"key": "bar",  "value": "1"},
  {"key": "baz",  "value": "2"}
]
```

Features - Prefix Search - How

```
// NOTE: See slide #9.
```

Features - Bloom Filters

- Space-efficiently "decide" whether element either:
    - Possibly in set
    - Definitely not in set
- Used when `SeekPrefixGE` or `Get` is called, not used by default!

"A Bloom filter with a 1% error and an optimal value of k, in contrast, requires only about 9.6 bits per element, regardless of the size of the elements."

Features - Bloom Filters - How

```
pebble.LevelOptions{
     Compression:    pebble.SnappyCompression,
     BlockSize:      blockSize,
     TargetFileSize: int64(sz),
     FilterPolicy:   bloom.FilterPolicy(10), // <- magic constant!
     FilterType:     pebble.TableFilter,
}
// NOTE: see regatta/pebble.go:OpenDB for real-world usage.
```

Features - Bloom Filters - How (standalone)

```go
import (
    "fmt"

    "github.com/cockroachdb/pebble"
    "github.com/cockroachdb/pebble/bloom"
)

func main() {
    fp := bloom.FilterPolicy(10)
    w := fp.NewWriter(pebble.TableFilter)

    w.AddKey([]byte{100})
    a := w.Finish(nil)

    contains := fp.MayContain(pebble.TableFilter, a, []byte{100})
    fmt.Printf("contains: %v\n", contains)
}
```

Go Competition

- Badger - https://github.com/dgraph-io/badger
- Bbolt - https://github.com/etcd-io/bbolt
- Pogreb - https://github.com/akrylysov/pogreb

- Benchmarks - https://github.com/smallnest/kvbench (Pebble not measured correctly 🙃)*

* https://cockroachdb.slack.com/archives/CQVRDNE23/p1625591005015400

Further Reading

- LSM Compaction Mechanism (blogpost)
- LSM (whitepaper)
- CockroachDB - Introducing Pebble
- SSTable and Log Structured Storage
- Pebble Package Documentation

That's all folks!