BoltDB

An embedded K/V database for Go

Huseyin Ekemen

Golang & DevOps Developer

https://www.nextpax.com

Co-Founder @ pdaccess:

https://www.pdaccess.com

5 years in Golang

19 Years with Java

What we have been looking for:

- Single File Database (like sqlite)
- Should be pure go based.
- Has to consist about crash and concurrency cases.
- Reasonably good performance.(Both write & read)

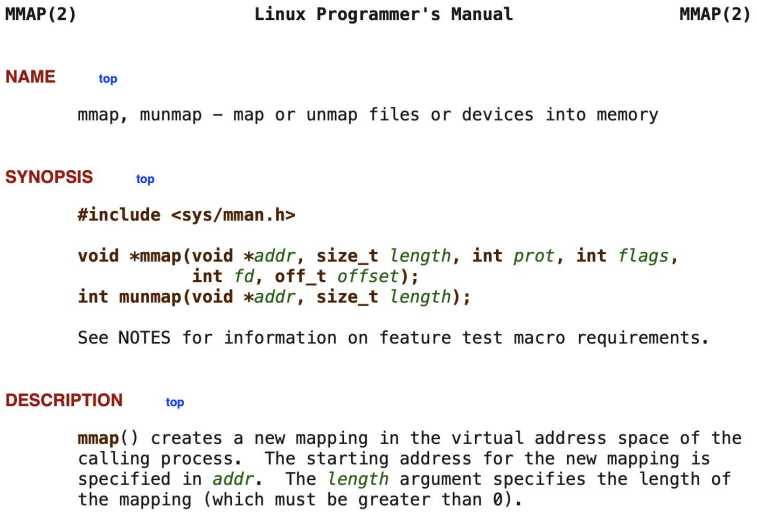
BoltDB:

- LMDB(The Lightning Memory-Mapped Database)

- It is very small code base.
- It is embedded. Run inside your binary
- Single File DataStore
- Simple and stable api.
- ACID based (lock-free MVCC

  using a single writer and multiple

  readers)



MMAP(2) Linux Programmer's Manual MMAP(2)

NAME top

mmap, munmap - map or unmap files or devices into memory

SYNOPSIS top

```
#include <sys/mman.h>

void *mmap(void *addr, size_t length, int prot, int flags,
           int fd, off_t offset);
int munmap(void *addr, size_t length);
```

See NOTES for information on feature test macro requirements.

DESCRIPTION top

mmap() creates a new mapping in the virtual address space of the calling process. The starting address for the new mapping is specified in *addr*. The *length* argument specifies the length of the mapping (which must be greater than 0).

Concept:

- File == Database
- Bucket == Table
- Key-Value == map[[]byte][]byte
- Transaction == Lock

- ReadOnly: Multiple read-only transaction is ok
- ReadWrite: Only one readwrite transaction is allowed.
- Batch: Multiple write together.

DB: Open

```
db, err := bolt.Open("my.db", 0600,
&bolt.Options{Timeout: 1 * time.Second})

// my.db is the filename of the database.
// there are different options for bolt.Open
// After usage you should close the db instance
// db.Close()
```

Transaction: Write

```
db.Update(func(tx *bolt.Tx) error {
    b, err := tx.CreateBucket([]byte("MyLogBucket"))
    if err != nil {
        return fmt.Errorf("create bucket: %s", err)
    }
    return nil
})

// There is a method named CreateBucketIfNotExists()
```

Transaction: Write

```
db.Update(func(tx *bolt.Tx) error {
    b := tx.Bucket([]byte("MyLogBucket"))
    err := b.Put([]byte("user"), []byte("user1"))
    return err
 })


 // For Auto increment

 // id, _ := b.NextSequence()
```

Transaction: Read

```
db.View(func(tx *bolt.Tx) error {
    b := tx.Bucket([]byte("MyLogBucket"))
    v := b.Get([]byte("user"))
    fmt.Printf("The User is: %s\n", v)
    return nil
})
```

Transaction: Read

```
db.View(func(tx *bolt.Tx) error {
     // Assume bucket exists and has keys
     b := tx.Bucket([]byte("MyLogBucket"))

     c := b.Cursor()

     for k, v := c.First(); k != nil; k, v = c.Next() {
          fmt.Printf("key=%s, value=%s\n", k, v)
     }

     return nil
})

// The following functions are available on the cursor:
// First()  Move to the first key.
// Last()   Move to the last key.
// Seek()   Move to a specific key.
// Next()   Move to the next key.
// Prev()   Move to the previous key.
```

Transaction: Batch

```
db.Batch(func(tx *bolt.Tx) error {
    b := tx.Bucket([]byte("MyLogBucket"))
    b.Put([]byte("user"), []byte("user1"))
    b.Put([]byte("device"), []byte("device1"))

    return err
 })


 // Each update require file write

 // Single batch can help here.
```

More

- Dump stats easily

```
// Grab the initial stats.
prev := db.Stats()

for {
        // Wait for 10s.
        time.Sleep(10 * time.Second)

        // Grab the current stats and diff them.
        stats := db.Stats()
        diff := stats.Sub(&prev)

        // Encode stats to JSON and print to STDERR.
        json.NewEncoder(os.Stderr).Encode(diff)

        // Save stats for the next loop.
        prev = stats
}
```

- Data is stored as []byte type.
- You can open database as read-only mode many times.
- Only one write-based access to the database file.
- Single func backup: `_, err := tx.WriteTo(w)`
- You can use Seek() for Range & prefix scan
- ForEach() to loop over a bucket.

Links

- https://github.com/boltdb/bolt main project. It is abundant but still ok.
- https://github.com/etcd-io/bbolt successor of boltdb with same interface. You can continue with this one. (has live project.)
- http://www.lmdb.tech/media/20130406-LOADays-LMDB.pdf LMDB concept.