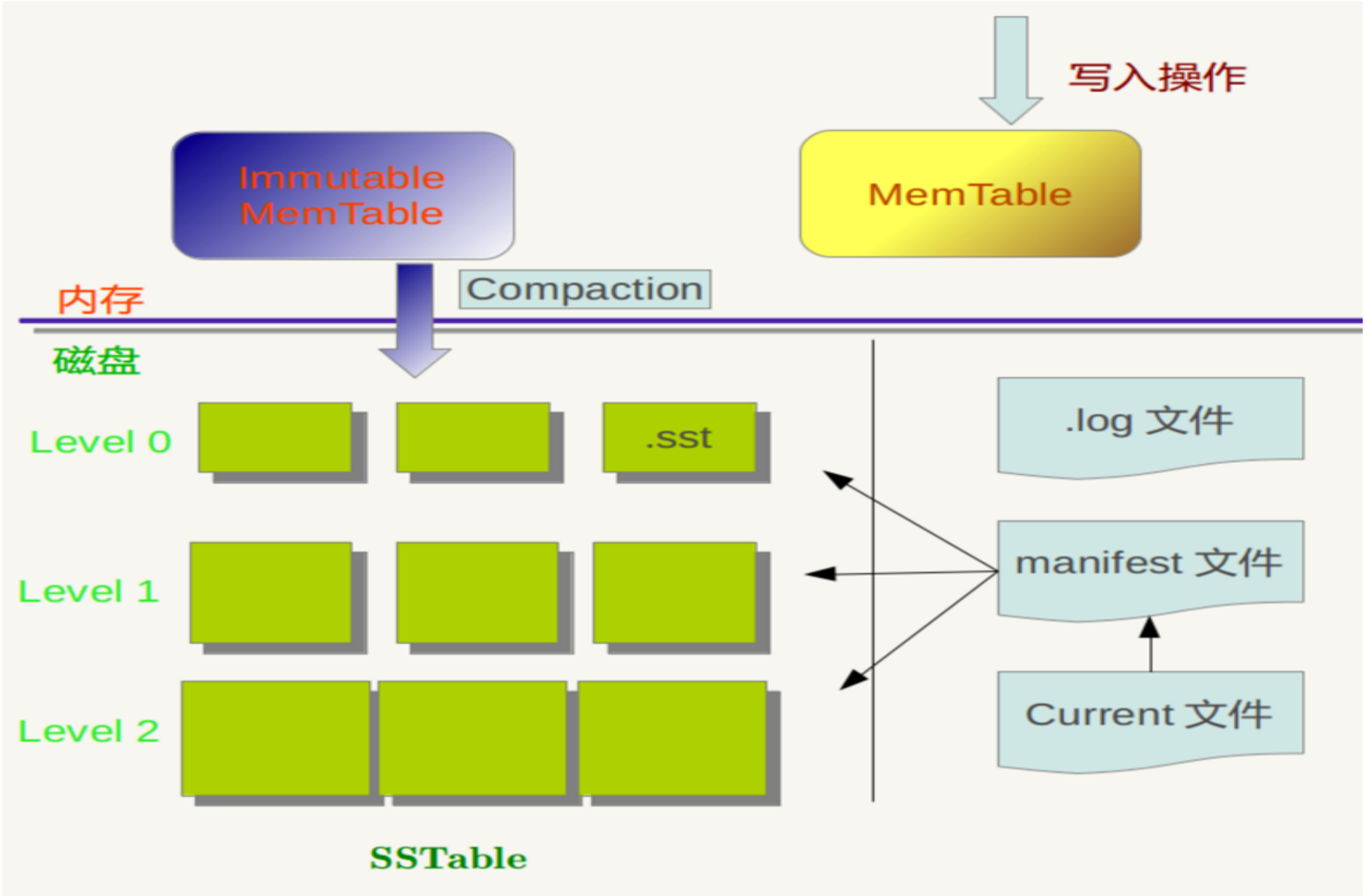
LevelDB

Arch


写入操作
Immutable
MemTable
MemTable
Compaction
内存
磁盘
Level 0
.sst
.log 文件
Level 1
manifest 文件
Level 2
Current 文件
SSTable

Log Structured Merge Tree

1. append only write (just like log)
2. write is extremely fast
3. read may lookup many places
4. delete is a special kind of write
5. space amplification && need compaction

Write

1. write log
2. write memtable

Read

1. memtable -> immutable memtable -> all sstables in level0 overlap the key -> the sstable in level1 overlap the key -> … -> the sstable in level6 overlap the key

2. bloom filter

3. sstable : index block / data block

Delete

1. kTypeValue = 0x1
2. kTypeDeletion = 0x0
3. InternalKey = UserKey + sequence number + type

Snapshot

1. InternalKey = UserKey + sequence number + type

2. all read&&write grab a sequence number first

```
const Snapshot* DBImpl::GetSnapshot() {
  MutexLock l(&mutex_);
  return snapshots_.New(versions_->LastSequence());
}
```

a list

Iterator

1. DBIter(MergingIterator(mem_itr, imm_itr, L0_a_itr, L0_b_itr, … L0_n_itr, L1_itr, L2_itr, … L6_itr))

2. create a iterator will pin all sstables and memtable(include immutable) until delete

Compaction

1. flush immutable memtable to sstable (minor)

2. levelN levelN+1 … ===> LevelN+1 … (major)

Compaction

calculate score for each level

level0 score = files count / 4

level1 leveln score = TotalFileSize / MaxBytesForLevel

(MaxBytesForLevel1-6: 160M 1.6G 16G 160G 1600G 16000G)

pick compaction (best_score > 1)

do compaction (multiway merge)

Compaction

1. snapshot
2. iterator