ShardKV 相关结构体及方法

结构体

ShardKV

```go
type ShardKV struct {
        mu           sync.Mutex
        me           int
        rf           *raft.Raft
        applyCh      chan raft.ApplyMsg
        make_end     func(string) *labrpc.ClientEnd
        gid          int
        ctrlers      []*labrpc.ClientEnd
        maxraftstate int // snapshot if log grows this big
        persister    *raft.Persister

        // Your definitions here.
        dead   int32 // set by Kill()
        shards map[int]*Shard

        shardsMigrateWG *sync.WaitGroup

        // Leader回复给客户端的响应（日志Index， CommandReply）
        notifyChans map[int]chan ApplyChanResult
        lastApplied int

        // use for query
        ctrlerClerk   *shardctrler.Clerk
        lastConfig    shardctrler.Config
        currentConfig shardctrler.Config
}
```

group

```go
type group struct {
        gid       int
        servers   []*ShardKV
        saved     []*raft.Persister
```

```
        endnames  [][]string
        mendnames [][]string
}
```

Shard

```
type Shard struct {
    Num int
    KV map[string]string
    LastClientOperation map[int64]ClientOperation
    Status ShardStatus
}
```

Command 日志（日志类型）

https://zhuanlan.zhihu.com/p/463146544

为了实现上述题意，可以定义了五种类型的日志，这样 apply 协程就可以根据不同地类型来强转 `Data` 来进一步操作：

- Operation：客户端传来的读写操作日志，有 Put，Get，Append 等请求。
- Configuration：配置更新日志，包含一个配置。
- InsertShards：分片更新日志，包含至少一个分片的数据和配置版本。
- DeleteShards：分片删除日志，包含至少一个分片的 id 和配置版本。
- EmptyEntry：空日志，`Data` 为空，使得状态机达到最新。

```
type Command struct {
    CommandType int
    Data interface{}
}
```

```
const (
    ExecuteTimeout = 1

    OpGet = 1
    OpPut = 2
    OpAppend = 3

    CommandConfiguration = 20
```

```
    CommandKVOperation = 21
    CommandInstallShard = 22
    CommandDeleteShard = 23
)
```