基于Raft的k/v service系统设计文档



作者姓名 秦清澳

作者学号 22151174

所在学院 软件学院

提交日期 二○二一 年12月

## 系统总体架构

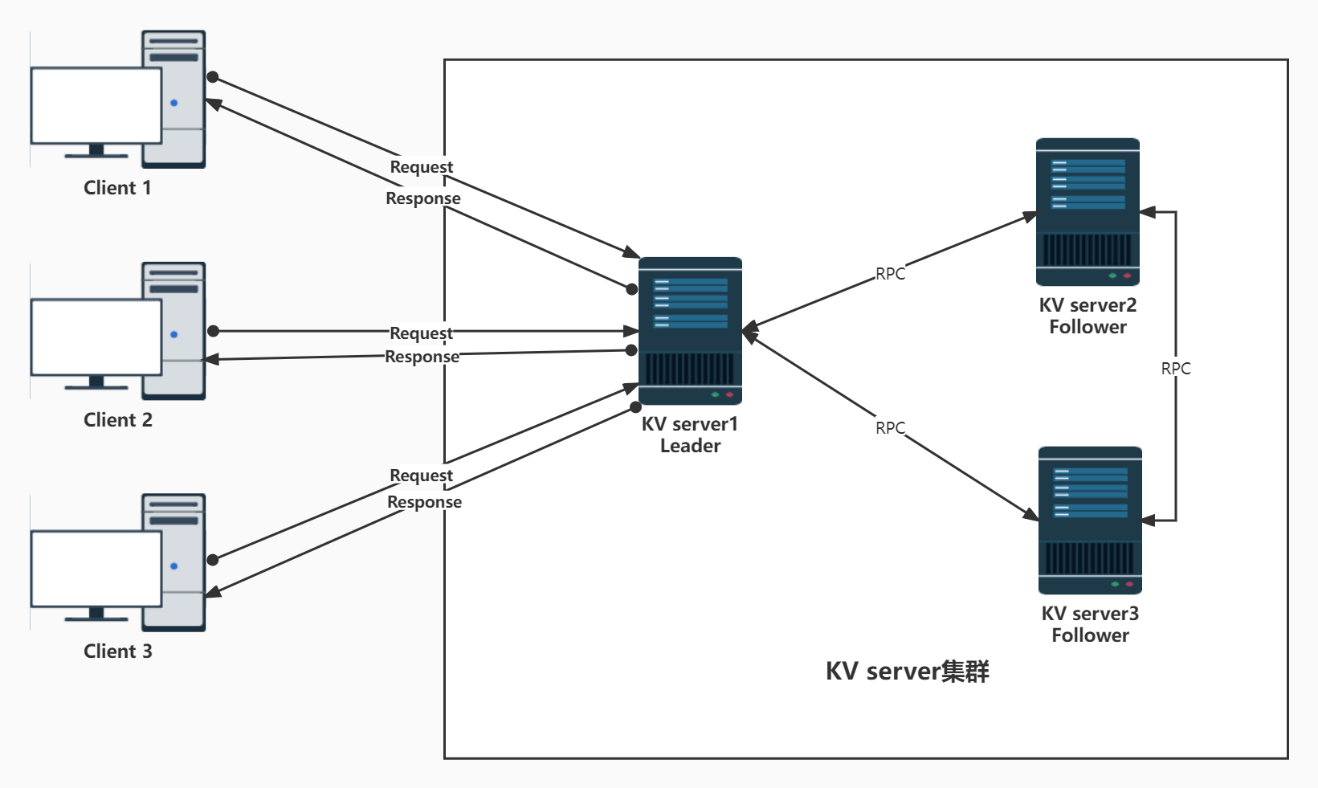


图1 总体架构

系统的总体架构如图1所示，由若干客户端和一个KV Server集群组成。Server集群由一个Leader和若干Follower组成。客户端向集群请求Get/Put/Append三种KV服务，只有Leader会响应给客户端正确的服务。Server集群之间通过Raft协议的RPC保证数据一致性。

## 2. 整体设计

### 2.1 KV Sever cluster设计

图2 KV Server cluster通信设计

KV Server cluster之间的通信如图2所示，Leader通过RPC的方式调用Follwer的InstallSnapshot以及AppendEntries实现快照安装以及日志项复制实现分布式一致性。另外，Follwer之间还可以通过调用RequesVote发起投票从而实现Leader选举。

### 2.2 KV Server设计

图3 KV Server 设计

KV Server 的内部设计如图3所示，提供的k/v service有Get、Put/Append三种，通过State Machine与Raft实例、Persister实例实现。

State Machine通过Start函数向Raft实例下达执行日志项中的命令，Raft实例通过applyCh通道响应命令执行情况。另外，State Machine通过Snapshot函数将当前State Machine的state创建快照，Raft实例调用SaveStateAndSnapshot、函数进行持久化，将Raft协议当前的状态(任期、LeaderID)以及状态机的状态（日志项相关信息）保存到内存（Memory）。

Raft实例也可以通过Persist方法只进行Raft协议状态的持久化。如果server发生网络故障后重连，Raft实例可以调用readPersist函数读取当前Raft协议的状态。

另外，Leader会通过RPC的方式远程调用Follwer的CondInstallSnapshot函数进行快照的安装，也是通过applyCh通道响应。状态机可以通过调用ReadSnapshot函数读取最新快照。而k/v service层可以通过ReadPersist函数读取最新的key/value键值对。

## 3. Raft算法实现

### 3.1 Raft算法简介

Raft算法是坦福大学的Dicgo Ongaro和John Oustcrhout两名学者针对Paxos算法实现困难、难以理解的问题而设计的算法。Raft算法具有和 Paxos 算法相似的功能和相近的性能，但它是基于领导者的设计模式，使得 Raft 算法易于理解，同时更易于工业实现。Raft算法是一种分布式系统共识算法，它设计的一个重要目标就是易于理解和工程实现，基于复制状态机（Replicated State Machine )模型的设计理念和基于Leader节点的日志复制方案简化了整个算法的状态空间，使得该目标得以实现， Raft 算法在实现数据一致性时沿用了**复制状态机**的思想，在一个集群中，如果每个节点所接收到的指令一致，那么经由状态机(State Machine)执行后得到的状态也一定是一致的。

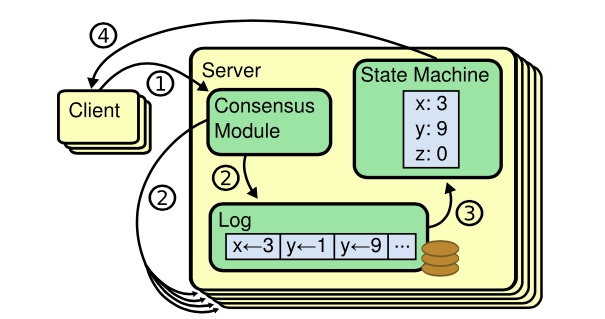


图2 复制状态机模型

### 3.2重要struct

#### 3.2.1 ApplyMsg

当Leader通过RPC的方式调用Follower执行命令(Command)和快照操作时,ApplyMsg被用来封装响应消息传入通道（channel），可以理解为函数的返回值。

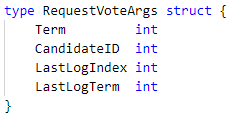
#### 3.2.2. Raft

Raft是一个节点需要维持的所有信息的集合，重要字段包括记录其他对等节点的peers数组、用于持久化的persister对象、当前任期currentTerm等。

### 3.3 Leader 选举实现

#### 3.3.1 重要struct

##### 3.3.1.1 RequestVoteArgs



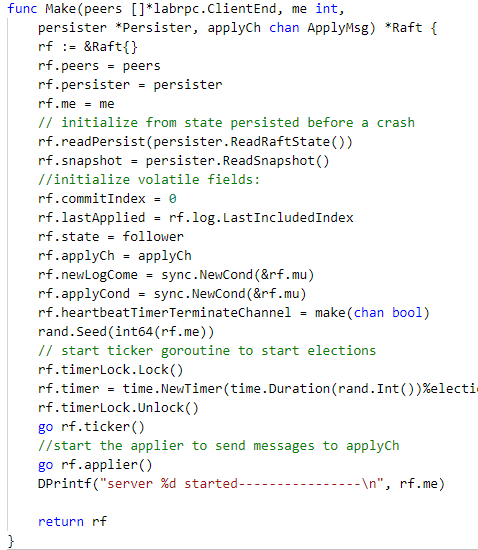
RequestVoteArgs是进行领导者选举投票时传入的参数，包括当前节点保存的领导者的任期Term、候选者IDCandidateID、最后一个日志项索引LastLogIndex以及最后一个日志项的任期LastLogTerm。

##### 3.3.1.2 RequestVoteReply

RequestVoteReply用来封装节点投票的信息，包括当前节点保存的领导者的任期Term，以及是否为发起者投票VoteGranted。

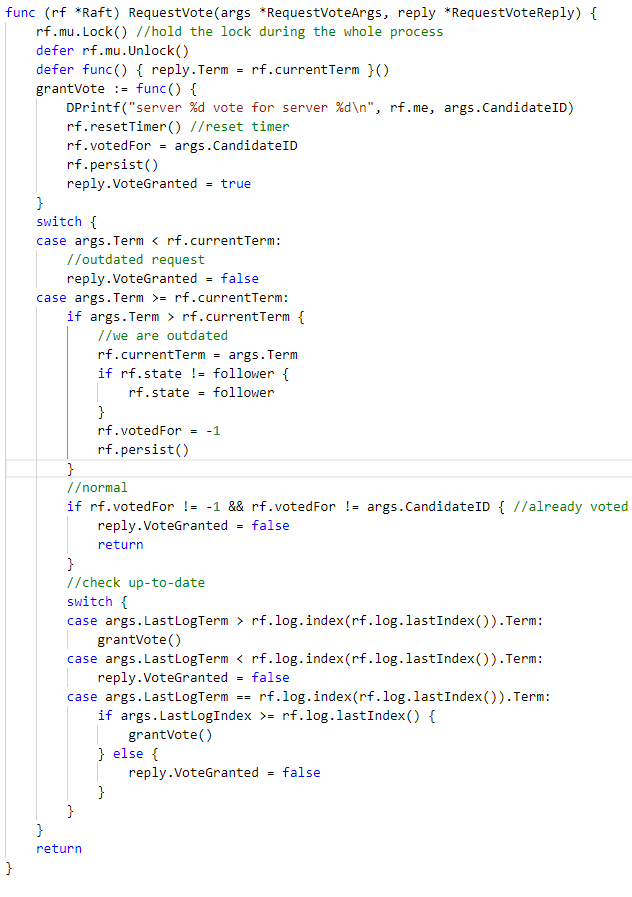
#### 3.3.2 重要函数

##### 3.3.2.1 Make(peers,me,persister,applyCh)



Make函数的参数包括其他节点peers、自己的编号me、持久化对象persister、接受信息的通道applyCh。该函数主要用来初始化一个Raft实例，并发起选举。

##### 3.3.2.2 RequestVote(requestVoteArgs, requestVoteReply)



RequestVote函数的参数包括请求投票的相关参数RequestVoteArgs、回应请求的相关参数RequestVoteReply。该函数主要用来响应发起投票的节点的选举请求。

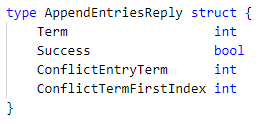
### 3.4 日志复制实现

#### 3.4.1 重要struct

##### 3.4.1.1 LogEntry & logType

LogEntry封装单个日志项，包括的提交该日志项Leader的任期Term以及对应的状态机命令Command。而logType则包含了当前状态机包含的所有日志项数组Entries以及最好一个日志项的索引LastIncludedIndex和任期LastIncludedTerm。

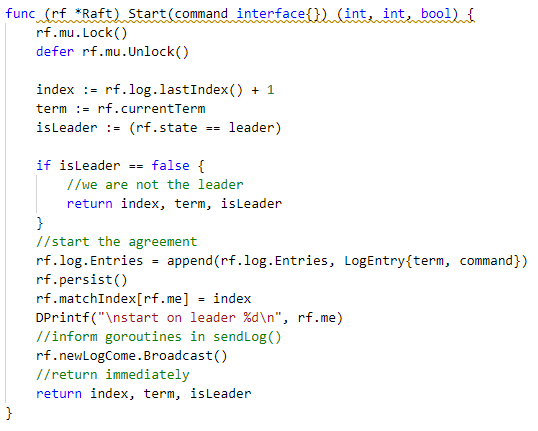
##### 3.4.1.2 AppendEntriesArgs

AppendEntriesArgs是添加日志项所用的参数，包括Leader的任期Term、Leader的编号LeaderID、前一条日志项的索引PrevLogIndex和任期PrevLogTerm、要添加的所有日志项数组Entries、Leader已提交的最后一个日志项索引LeaderCommit。

AppendEntriesReply是添加日志项返回信息的封装，包括Leader的任期Term、成功与否Success、冲突的日志项的任期ConflictEntryTerm、冲突的日志项的索引ConflictTermFirstIndex。

#### 3.4.2 重要函数

##### 3.4.2.1 Start(command)

 Start函数是Leader用来将k/v service的相应操作做华为日志项并使用状态机执行相应命令。

##### 3.4.2.2 AppendEntries(appendEntriesArgs, appendEntriesReply)

AppendEntries函数的参数包括添加日志项的相关信息参数AppendEntriesArgs、响应参数AppendEntriesReply。Leader通过RPC的方式调用该函数来添加日志项，保证数据的一致性。

### 3.5 持久化

#### 3.5.1 重要函数

##### 3.5.1.1 persist()

persist函数负责将Raft协议的状态持久化。这里的Raft协议的状态主要包括当前领导者的任期CurrentTerm、最后一次投出得票votedFor、日志项log。

##### 3.5.1.2 readPersist(data)

readPersist函数负责将保存的Raft状态还原。

### 3.6 日志压缩

#### 3.6.1 重要struct

##### InstallSnapshotArgs & InstallSnapshotReply

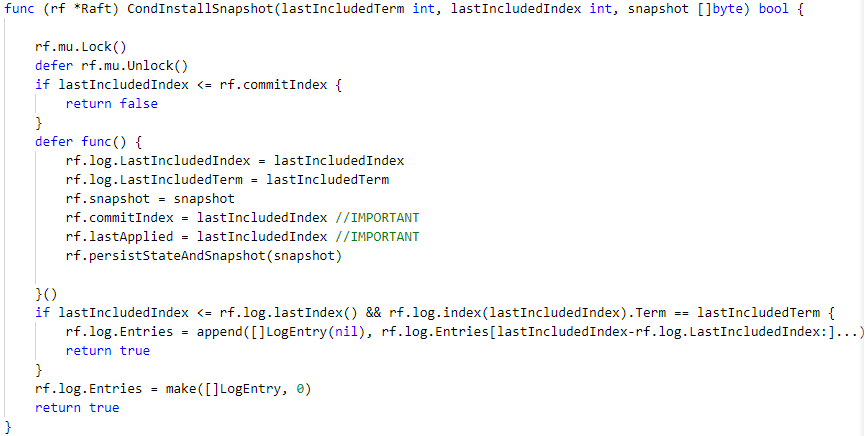
InstallSnapshotArgs是Leader让Follwer安装快照时传入的参数。主要包括任期Term、Leader编号LeaderID、快照包含的最后一个日志项索引LastIncludedIndex和任期LastIncludedTerm、持久化的数据Data。InstallSnapshotReply作为返回值只包含了任期Term。

#### 3.6.2 重要函数

##### 3.6.2.1 Snapshot(index，snapshot)

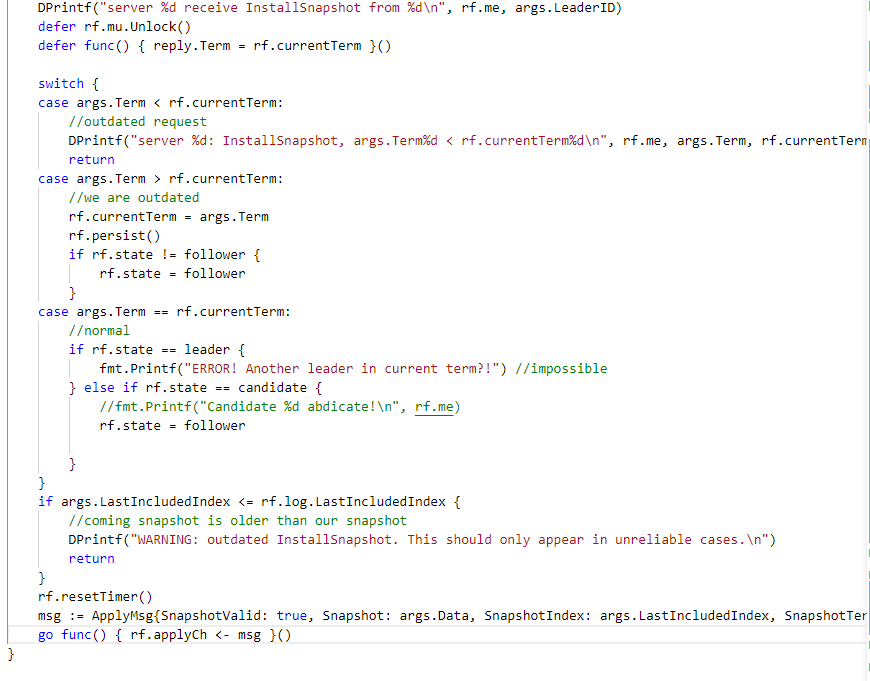
Snapshot是通过创建快照的方式压缩日志项，参数主要包括现在的最新的日志项索引index以及之前创建的快照snapshot。

##### 3.6.2.2 CondInstallSnapshot(lastIncludedTerm, lastIncludedIndex, snapshot)



CondInstallSnapshot的参数包括最后的日志项任期lastIncludedTerm和索引lastIncludedIndex、之前的快照snapshot。该函数用来切换到最新快照。

##### 3.6.2.3 InstallSnapshot(installSnapshotArgs,installSnapshotReply)



InstallSnapshot的参数包括要安装的快照信息installSnapshotArgs、用于返回的安装结果installSnapshotReply。Leader通过RPC调用Follower的InstallSnapshot函数来保证数据源一致性。

## 4. KV Service实现

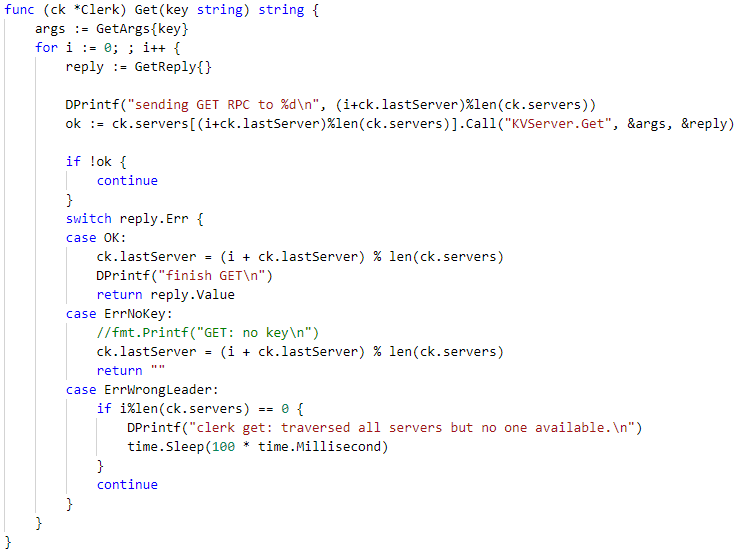
### 4.1 KV Service简介

KV Service基于Raft算法实现，提供容错key/value存储服务。key/value服务是一个可复制的状态机，由几个使用Raft算法进行复制的KVserver组成。只要大多数服务器处于活动状态并能够通信，那么KV Service就能继续处理客户端请求。服务支持三种操作:Put(key, value)、Append(key, arg)和Get(key)。维护了一个简单的k/v键值对数据库。key和value都是字符串。Put()替换数据库中特定key的value，Append(key, arg)将arg附加到kay的value上，Get()获取key的当前对应的value。对于一个不存在的key的Get应该返回一个空字符串。对一个不存在的key执行Append操作类似于Put。每个客户端通过一个具有Put/Append/Get方法与服务器会话。客户端与服务器通过RPC交互。

### 4.2 Client端实现

#### 4.1.1 重要函数

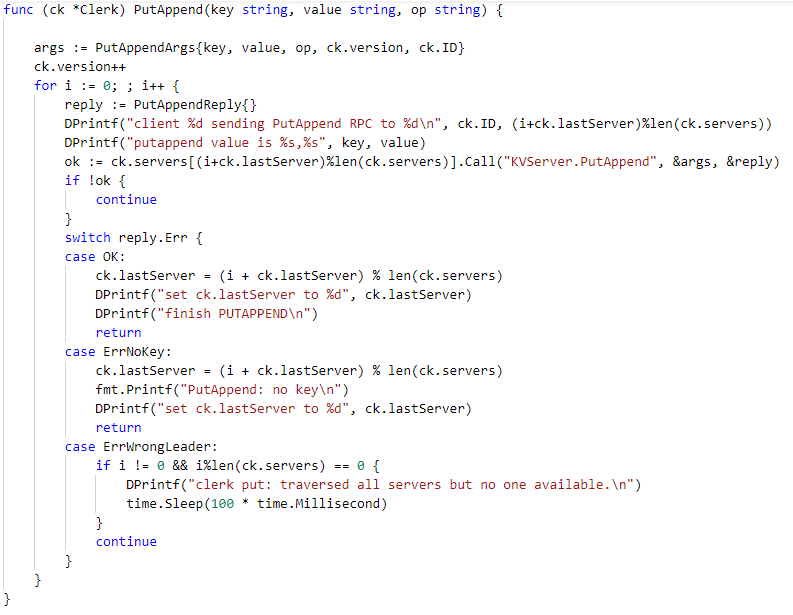
##### 4.1.1.1 Get(key)



Get函数的参数只有一个关键字字符串key。该函数使得客户端通过RPC的方式请求服务器的Get服务。

##### 4.1.1.2 PutAppend(key, value, op)

PutAppend函数的参数包括关键字字符串key、值字符串value、具体操作op。该函数与Get函数类似，客户端调用该函数通过RPC的方式请求服务器的Put或者Append服务。



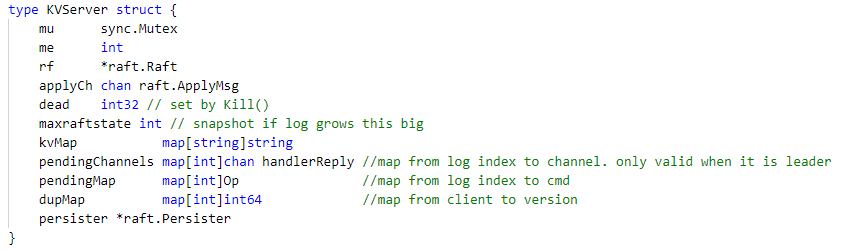
### 4.3 Server端实现

#### 4.3.1重要struct

##### 4.3.1.1 Op

Op封装了客户端请求服务的相关操作信息，包括操作类型Type、关键字Key、值Value、版本号Version、编号ID、领导者编号Leader。

##### 4.3.1.2 KVServer



KV Server封装了一个Server的所有相关信息，主要包括Raft实例rf、通信通道applyCh、最大Raft状态maxraftstate、实现kv存储的kvMap、日志索引到处理结果的映射pendingChannels、日志索引到命令的映射pendingMap、客户端到版本号的映射dupMap、持久化对象persister。

#### 4.3.2 重要函数

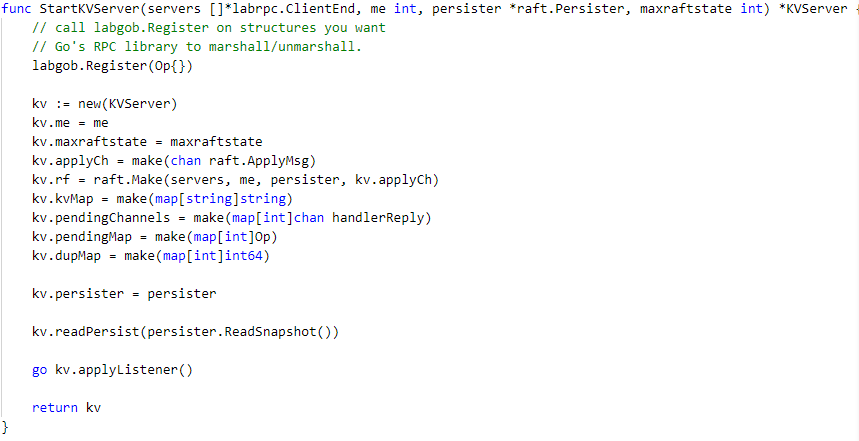
##### 4.3.2.1 Get(args, reply)

Get函数的参数包括Get操作的相关参数信息args、用于返回结果的reply。该函数的主要功能就是使服务器通过Raft进行Get操作，获得key对应的value，并改变状态机的状态，添加日志项。

##### 4.3.2.2 PutAppend(args, reply)

PutAppend函数的参数包括Put或Append操作的相关参数信息args、用于返回结果的reply。该函数的主要功能就是使服务器通过Raft进行Put或Append操作，改变key对应的value，并改变状态机的状态，添加日志项。要注意的是该函数实现了操作去重的功能、相同操作只会进行一次。

##### 4.3.2.3 StartKVServer(servers, me, persister, maxraftstatet)



StartKVServer的参数包括其他服务器servers、自己的编号me、持久化对象persister、允许Raft状态所占最大空间maxraftstate。该函数主要负责新建一个KV Server实例，并初始化相关参数以及新建相应的Raft实例。另外，还会从快照中读取最新的状态并开启监听器监听客户端的请求。

##### 4.3.2.4 applyListener()

applyListener函数主要负责监听客户端的请求并由Leader对其服务，另外该函数还进行了是否Leader判断、命令是否有效、快照操作。