1、完成lru（最近最少使用）算法，使用双端链表，Get的时候将key放到队头，Add的时候先将key放到队头，然后看有没有超过最大值限制，超过了从队尾依次去掉旧的key，直到总内存小于最大值限制。

2、接着将lru做成cache，添加了并发锁，来保证并发性。引入group概念，主要用来区分不同的数据库表。在group中的get逻辑为，首先查找内存中是否有这个key，如果有就直接返回内存值，如果没有就从本地加载。

3、设计HTTP客户端，远程节点之间会使用HTTP来进行请求。通过重载ServeHTTP函数来处理从不同的group里获取key相关值。

4、接着是一致性哈希，一致性哈希是为了选取合适的远程节点，如果没有一致性哈希，那么选取远程节点就是随机的，会有额外的开销，同时为了解决数据倾斜问题，可以引入虚拟节点，再使用哈希表将虚拟节点与真实key挂钩。一致性哈希是，将节点放在环上，然后来了key以后计算key的哈希值，顺时针找到第一个节点就是他需要访问的远程节点。

5、接着是改造HTTP客户端。抽象出两个接口，一个是远程节点选择接口PeerPicker，另一个是客户端查找数据接口PeerGetter。改造HTTP客户端，加入PeerPicker接口和PeerGetter哈希表，将Get方法设置为访问对应url获取值，将PeerPicker使用一致性哈希算法来选择远程节点。

6、修改group的整体的逻辑，先在内存中查找是否有这个key，如果有就直接返回，如果没有那么选取一个远程节点，访问这个远程节点看他内存中是否有这个key如果有直接返回，否则再去本地查找。

7、最后一步是对访问远程节点进行优化，使用singleflight，实现Do函数，Do函数作用是针对相同的key，无论Do被调用多少次，函数fn（比如远程节点从数据库读取数据函数）只会调用一次，等待fn调用完后，返回统一值。实现Do函数后，将加载函数（就是本地内存中没有数据调用的那个函数）改成使用Do函数来实现。这样可以一定程度上防止缓存击穿。