

```
1 package main
2
3 import (
4     "fmt"
5     "github.com/marcelloh/fastdb"
6     "log"
7     "net"
8     "net/http"
9     "net/rpc"
10    "os"
11    "strconv"
12    "sync"
13    "time"
14 )
15
16 // Service quản lý database của server
17 type Service struct {
18     db      *fastdb.DB
19     mutex   *sync.Mutex
20 }
21
22 // Khởi tạo database riêng cho từng server
23 func NewService(dbFile string) (*Service, error) {
24     db, err := fastdb.Open(dbFile, 100)
25     if err != nil {
26         log.Fatal(err)
27     }
28
29     return &Service{
30         db:      db,
31         mutex:   new(sync.Mutex),
32     }, nil
33 }
34
35 // == Structs Request & Reply ==
36 type RegisterServerRequest struct {
37     Address string
38 }
39 type RegisterServerReply struct {
40     Success bool
41     Message string
42 }
43
44 type GetRequest struct {
45     Bucket string
46     Key     int
47 }
48 type GetReply struct {
49     Success bool
50     Data    []byte
51     Err     error
52 }
53
```

```
54 type SetRequest struct {
55     Bucket string
56     Key     int
57     Data    []byte
58 }
59 type SetReply struct {
60     Success bool
61     Err      error
62 }
63
64 type DeleteRequest struct {
65     Bucket string
66     Key     int
67 }
68 type DeleteReply struct {
69     Success bool
70     Err      error
71 }
72
73 type GetAllRequest struct {
74     Bucket string
75 }
76 type GetAllReply struct {
77     Success bool
78     Data    map[int][]byte
79     Err      error
80 }
81
82 type GetInfoRequest struct{}
83 type GetInfoReply struct {
84     Success bool
85     Info     string
86     Err      error
87 }
88
89 // == Các phương thức RPC ==
90 func (s *Service) Get(req *GetRequest, reply *GetReply) error {
91     log.Printf("Server received Get request - Bucket: %s, Key: %d",
92         req.Bucket, req.Key)
93
94     data, ok := s.db.Get(req.Bucket, req.Key)
95     if ok {
96         reply.Success = true
97         reply.Data = data
98         log.Printf("Server found data for Key %d: %s", req.Key,
99             string(data))
100     } else {
101         reply.Success = false
102         reply.Err = fmt.Errorf("key not found")
103         log.Printf("Server could not find Key %d", req.Key)
104     }
105     return nil
106 }
107
108 func (s *Service) Set(req *SetRequest, reply *SetReply) error {
```

```
107     log.Printf("Server received Set request - Bucket: %s, Key: %d",
108 req.Bucket, req.Key)
109     s.mutex.Lock()
110     defer s.mutex.Unlock()
111     err := s.db.Set(req.Bucket, req.Key, req.Data)
112     if err != nil {
113         reply.Success = false
114         reply.Err = err
115         log.Printf("Server failed to store data - Key: %d", req.Key)
116         return err
117     }
118
119     reply.Success = true
120     log.Printf("Server successfully stored data - Key: %d", req.Key)
121     return nil
122 }
123
124 func (s *Service) Delete(req *DeleteRequest, reply *DeleteReply) error {
125     s.mutex.Lock()
126     defer s.mutex.Unlock()
127     _, err := s.db.Del(req.Bucket, req.Key)
128     if err != nil {
129         reply.Success = false
130         reply.Err = err
131     } else {
132         reply.Success = true
133     }
134     return nil
135 }
136
137 func (s *Service) GetAll(req *GetAllRequest, reply *GetAllReply) error {
138     data, err := s.db.GetAll(req.Bucket)
139     if err != nil {
140         reply.Success = false
141     } else {
142         reply.Success = true
143     }
144     reply.Data = data
145     return nil
146 }
147
148 func (s *Service) GetInfo(req *GetInfoRequest, reply *GetInfoReply) error {
149     info := s.db.Info()
150     if info != "" {
151         reply.Success = true
152         reply.Info = info
153     } else {
154         reply.Success = false
155         reply.Err = fmt.Errorf("no info available")
156     }
157     return nil
158 }
159
160 // == Server tự động kết nối LoadBalancer ==
```

```

161 func registerWithLoadBalancer(serverAddress string) {
162     // Đợi 1 giây trước khi gửi request để đảm bảo server đã mở cổng
163     time.Sleep(1 * time.Second)
164
165     client, err := rpc.DialHTTP("tcp", "localhost:9000") // LoadBalancer
    chạy trên port 9000
166     if err != nil {
167         log.Printf("⚠ Failed to connect to LoadBalancer: %v", err)
168         return
169     }
170     defer client.Close()
171
172     req := &RegisterServerRequest{Address: serverAddress}
173     reply := &RegisterServerReply{}
174
175     err = client.Call("LoadBalancer.RegisterServer", req, reply)
176     if err != nil {
177         log.Printf("Failed to register with LoadBalancer: %v", err)
178         return
179     }
180
181     log.Printf("Server registered with LoadBalancer: %s", reply.Message)
182 }
183
184 // == Chạy Server ==
185 func main() {
186     // Nhận port từ dòng lệnh
187     if len(os.Args) < 2 {
188         log.Fatal("Usage: go run server_main.go <port>")
189     }
190     port := os.Args[1]
191     _, err := strconv.Atoi(port)
192     if err != nil {
193         log.Fatal("Invalid port number:", port)
194     }
195
196     // Mỗi server có database riêng theo port
197     dbFile := fmt.Sprintf("server_%s.db", port)
198     service, err := NewService(dbFile)
199     if err != nil {
200         log.Fatal("Error creating service:", err)
201     }
202
203     // Đăng ký RPC server
204     err = rpc.Register(service)
205     if err != nil {
206         log.Fatal("Error registering RPC:", err)
207     }
208     rpc.HandleHTTP()
209
210     // Lắng nghe trên port được chỉ định
211     address := ":" + port
212     listener, err := net.Listen("tcp", address)
213     if err != nil {
214         log.Fatal("Listen error:", err)

```

```
215     }
216
217     log.Printf("Server started on port %s with database: %s", port,
dbFile)
218
219     // Chạy server trong goroutine để không block chương trình
220     go func() {
221         err = http.Serve(listener, nil)
222         if err != nil {
223             log.Fatal("HTTP serve error:", err)
224         }
225     }()
226
227     // Đăng ký với LoadBalancer
228     registerWithLoadBalancer("localhost:" + port)
229
230     // Giữ chương trình chạy
231     select {}
232 }
233
```