Installation

Here is how to install a key-value store server, client using RPC and fastdb:

1. Install the FastDB Library

```
$ go get github.com/marcelloh/fastdb
```

2. Import FastDB and RPC Packages in Your Go Code

```go
package main

import (
    "log"
    "net"
    "net/rpc"
    "github.com/marcelloh/fastdb"
    "encoding/json"
)
```

3. Create a Server

   1. Implement RPC Methods

```go
func (kv *KeyValue) GetValue(request int, reply *string) error {
    value, ok := kv.db.Get("key-value.db", request)

    if ok {
        *reply = string(value)
    }

    return nil
}

func (kv *KeyValue) SetValue(record *KeyValueArgs, reply *string) error {
    recordData, _ := json.Marshal(record)
    err := kv.db.Set("key-value.db", record.ID, recordData)

    if err != nil {
        log.Fatal("Fail to set key-value", err)
    }

    *reply = "OK!"
    return nil
}
```

   2. Main function

```go
func main() {
    // 1. Init fastDB
    kv := new(KeyValue)
    db, err := fastdb.Open("key-value.db", 100) // "key-value.db" will be create if empty
    kv.db = db

    if err != nil {
        log.Println("Error opening database:", err)
        return
    }
    defer db.Close()

    // 2. Register service to RPC
    rpc.RegisterName("KeyValue", kv)

    listener, err := net.Listen("tcp", ":1234")
    if err != nil {
        log.Fatal("Can not create sever because:", err)
    }

  log.Print("Sever is listening on port 1234")

    // Allow RPC accept call from Client
    for {
        conn, err := listener.Accept()
        if err != nil {
            log.Fatal("Accept error:", err)
        }

        go rpc.ServeConn(conn)
    }
}
```

4. Create a Client

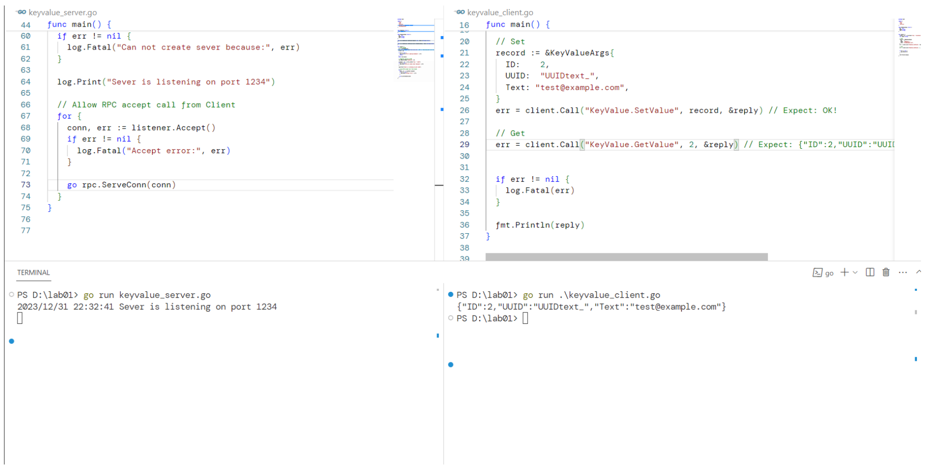
```go
package main

import (
    "fmt"
    "log"
    "net/rpc"
)

type KeyValueArgs struct {
    ID   int
    UUID string
    Text string
```

```
}


func main() {
    client, err := rpc.Dial("tcp", "localhost:1234")
    var reply string

    // Set
    record := &KeyValueArgs{
        ID:    2,
        UUID:  "UUIDtext_",
        Text: "test@example.com",
    }
    err = client.Call("KeyValue.SetValue", record, &reply) // Expect: OK!

    // Get
    err = client.Call("KeyValue.GetValue", 2, &reply) // Expect: {"ID":2,"UUID":"UUIDtext_"


    if err != nil {
        log.Fatal(err)
    }

    fmt.Println(reply)
}
```

4. Run

```
# Start the server
$ go run keyvalue_server.go
> 22:02:54 Sever is listening on port 1234

# Start the client
$ go run keyvalue_client.go
# Expect: {"ID": 2,"UUID":"UUIDtext_","Text":"test@example.com"}
```

Show case



keyvalue_server.go

```go
func main() {
	if err != nil {
		log.Fatal("Can not create sever because:", err)
	}

	log.Print("Sever is listening on port 1234")

	// Allow RPC accept call from Client
	for {
		conn, err := listener.Accept()
		if err != nil {
			log.Fatal("Accept error:", err)
		}

		go rpc.ServeConn(conn)
	}
}
```

keyvalue_client.go

```go
func main() {
	// Set
	record := &KeyValueArgs{
		ID:    2,
		UUID:  "UUIDtext_",
		Text: "test@example.com",
	}
	err = client.Call("KeyValue.SetValue", record, &reply) // Expect: OK!

	// Get
	err = client.Call("KeyValue.GetValue", 2, &reply) // Expect: {"ID":2,"UUID":"UUID


	if err != nil {
		log.Fatal(err)
	}

	fmt.Println(reply)
}
```

```
PS D:\lab01> go run keyvalue_server.go
2023/12/31 22:32:41 Sever is listening on port 1234
```

```
PS D:\lab01> go run .\keyvalue_client.go
{"ID":2,"UUID":"UUIDtext_","Text":"test@example.com"}
PS D:\lab01>
```

Figure 1: Screenshot 2023-12-31 223426