Go培训第九天

tony

Outline

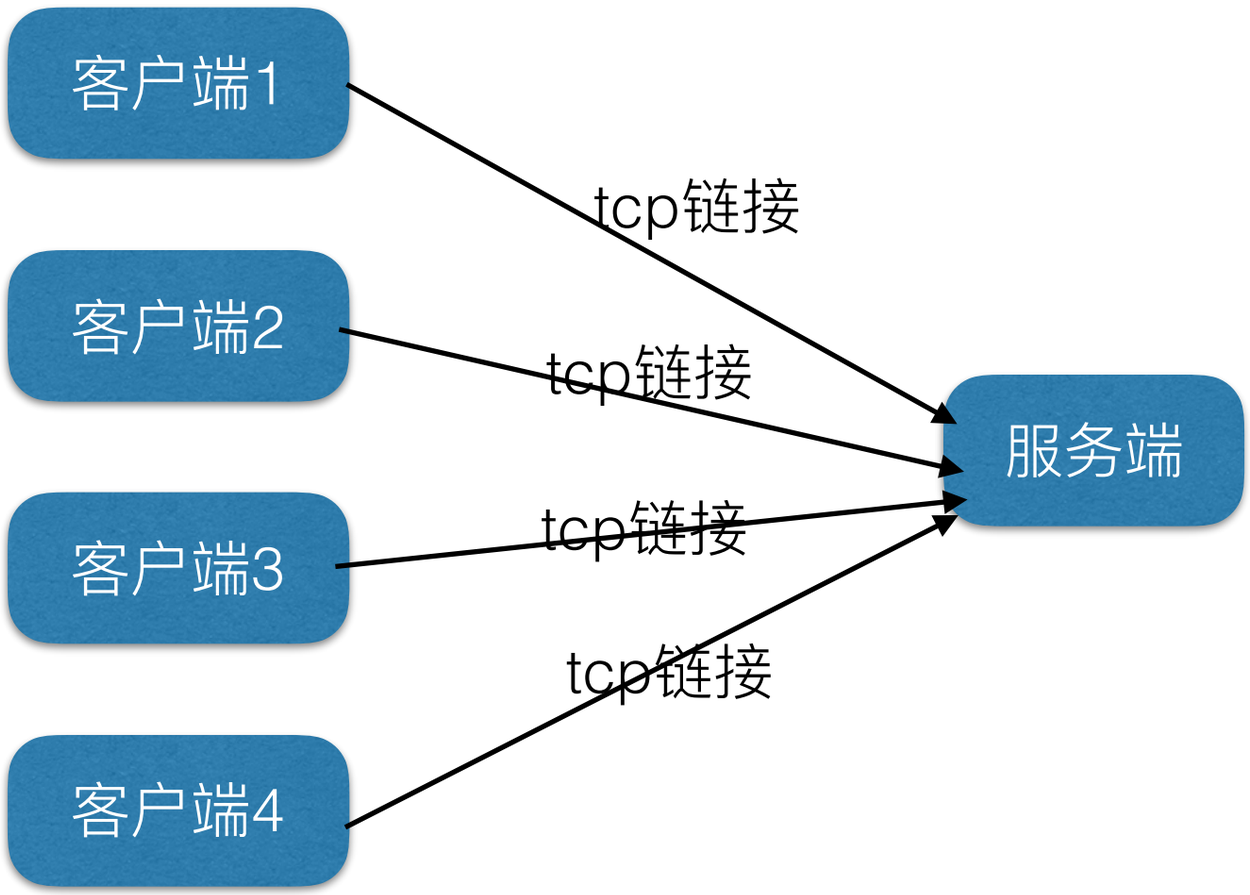
1. Tcp编程

2. redis使用

3. 课后作业

tcp编程

1. 客户端和服务器



socket编程

2. 服务端的处理流程

a. 监听端口

b. 接收客户端的链接

c. 创建goroutine，处理该链接

3. 客户端的处理流程

a. 建立与服务端的链接

b. 进行数据收发

c. 关闭链接

4. 服务端代码

```
package main

import (
	"fmt"
	"net"
)
func main() {
	fmt.Println("start server...")
	listen, err := net.Listen("tcp", "0.0.0.0:50000")
	if err != nil {
		fmt.Println("listen failed, err:", err)
		return
	}
	for {
		conn, err := listen.Accept()
		if err != nil {
			fmt.Println("accept failed, err:", err)
			continue
		}
		go process(conn)
	}
}
func process(conn net.Conn) {
	defer conn.Close()
	for {
		buf := make([]byte, 512)
		_, err := conn.Read(buf)
		if err != nil {
			fmt.Println("read err:", err)
			return
		}
		fmt.Println("read: ", string(buf))
	}
}
```

5. 客户端代码

```
package main

import (
	"bufio"
	"fmt"
	"net"
	"os"
	"strings"
)

func main() {

	conn, err := net.Dial("tcp", "localhost:50000")
	if err != nil {
		fmt.Println("Error dialing", err.Error())
		return
	}

	defer conn.Close()
	inputReader := bufio.NewReader(os.Stdin)
	for {
		input, _ := inputReader.ReadString('\n')
		trimmedInput := strings.Trim(input, "\r\n")
		if trimmedInput == "Q" {
			return
		}
		_, err = conn.Write([]byte(trimmedInput))
		if err != nil {
			return
		}
	}
}
```

socket编程

6. 发送http请求

```
package main

import (
	"fmt"
	"io"
	"net"
)
func main() {

	conn, err := net.Dial("tcp", "www.baidu.com:80")
	if err != nil {
		fmt.Println("Error dialing", err.Error())
		return
	}
	defer conn.Close()
	msg := "GET / HTTP/1.1\r\n"
	msg += "Host: www.baidu.com\r\n"
	msg += "Connection: close\r\n"
	msg += "\r\n\r\n"

	_, err = io.WriteString(conn, msg)
	if err != nil {
		fmt.Println("write string failed, ", err)
		return
	}
	buf := make([]byte, 4096)
	for {
		count, err := conn.Read(buf)
		if err != nil {
			break
		}
		fmt.Println(string(buf[0:count]))
	}
}
```

redis

7. redis

redis是个开源的高性能的key-value的内存数据库，可以把它当成远程的数据结构。

支持的value类型非常多，比如string、list（链表）、set（集合）、hash表等等

redis性能非常高，单机能够达到15w qps，通常适合做缓存。

redis

8. redis使用

使用第三方开源的redis库: github.com/garyburd/redigo/redis

```
import(
        “github.com/garyburd/redigo/redis"
)
```

redis

9. 链接redis

```
package main

import (
	"fmt"
	"github.com/garyburd/redigo/redis"
)

func main() {
	c, err := redis.Dial("tcp", "localhost:6379")
	if err != nil {
		fmt.Println("conn redis failed,", err)
		return
	}

	defer c.Close()
}
```

10. Set 接口

```
package main

import (
	"fmt"
	"github.com/garyburd/redigo/redis"
)

func main() {
	c, err := redis.Dial("tcp", "localhost:6379")
	if err != nil {
		fmt.Println("conn redis failed,", err)
		return
	}
	defer c.Close()
	_, err = c.Do("Set", "abc", 100)
	if err != nil {
		fmt.Println(err)
		return
	}

	r, err := redis.Int(c.Do("Get", "abc"))
	if err != nil {
		fmt.Println("get abc failed,", err)
		return
	}

	fmt.Println(r)
}
```

11. Hash表

```
package main

import (
	"fmt"
	"github.com/garyburd/redigo/redis"
)

func main() {
	c, err := redis.Dial("tcp", "localhost:6379")
	if err != nil {
		fmt.Println("conn redis failed,", err)
		return
	}
	defer c.Close()
	_, err = c.Do("HSet", "books", "abc", 100)
	if err != nil {
		fmt.Println(err)
		return
	}

	r, err := redis.Int(c.Do("HGet", "books", "abc"))
	if err != nil {
		fmt.Println("get abc failed,", err)
		return
	}

	fmt.Println(r)
}
```

11. 批量Set

```
package main

import (
	"fmt"
	"github.com/garyburd/redigo/redis"
)

func main() {
	c, err := redis.Dial("tcp", "localhost:6379")
	if err != nil {
		fmt.Println("conn redis failed,", err)
		return
	}

	defer c.Close()
	_, err = c.Do("MSet", "abc", 100, "efg", 300)
	if err != nil {
		fmt.Println(err)
		return
	}

	r, err := redis.Ints(c.Do("MGet", "abc", "efg"))
	if err != nil {
		fmt.Println("get abc failed,", err)
		return
	}

	for _, v := range r {
		fmt.Println(v)
	}
}
```

11. 过期时间

```
package main

import (
	"fmt"
	"github.com/garyburd/redigo/redis"
)

func main() {
	c, err := redis.Dial("tcp", "localhost:6379")
	if err != nil {
		fmt.Println("conn redis failed,", err)
		return
	}

	defer c.Close()
	_, err = c.Do("expire", "abc", 10)
	if err != nil {
		fmt.Println(err)
		return
	}
}
```

12. 队列操作

```
package main

import (
	"fmt"
	"github.com/garyburd/redigo/redis"
)

func main() {
	c, err := redis.Dial("tcp", "localhost:6379")
	if err != nil {
		fmt.Println("conn redis failed,", err)
		return
	}
	defer c.Close()
	_, err = c.Do("lpush", "book_list", "abc", "ceg", 300)
	if err != nil {
		fmt.Println(err)
		return
	}

	r, err := redis.String(c.Do("lpop", "book_list"))
	if err != nil {
		fmt.Println("get abc failed,", err)
		return
	}

	fmt.Println(r)
}
```

课后工作

1. 完善之前讲的图书管理系统

    a. 使用redis存储数据