UFPE

Entrega 5
RPC

Programação concorrente e distribuida

Lembrando...

Data Reader

```go
package datareader

import (
    "encoding/csv"
    "fmt"
    "log"
    "os"
)

type DataReaderRPC struct{}

func (t *DataReaderRPC) GetDataRPC(quantLinhas int, reply *[][]string) error {
    var linhas = make([][]string, quantLinhas)
    fmt.Println()
    f, err := os.Open("../datareader/data.csv")
    if err != nil {
        log.Fatal(err)
    }

    csvReader := csv.NewReader(f)

    for i := 0; i <= quantLinhas; i++ {
        // read csv values using csv.Reader
        data, err := csvReader.Read()
        if err != nil {
            log.Fatal(err)
        }
        if i > 0 {
            linhas[i-1] = append(linhas[i-1], data...)
        }
    }
    f.Close()

    *reply = linhas[:]

    return nil
}
```

SERVER GO RPC

```go
func server() {
    datareader := new(datareader.DataReaderRPC)
    // Cria server RPC
    server := rpc.NewServer()
    err := server.RegisterName("DataReader", datareader)
    if err != nil {
        fmt.Println(err)
        os.Exit(0)
    }
    // Cria listener
    ln, err := net.Listen("tcp", "localhost:1313")
    if err != nil {
        fmt.Println(err)
        os.Exit(0)
    }
    defer func(ln net.Listener) {
        var err = ln.Close()
        if err != nil {
            fmt.Println(err)
        }
    }(ln)
    // aguarda por requests
    fmt.Println("Server is ready ...")
    server.Accept(ln)
}
```

CLIENT GO RPC

```go
func client_RPC() {
    setUpLog()
    var requestTime time.Duration

    // conecta ao servidor
    client, err := rpc.Dial("tcp", "localhost:1313")
    if err != nil {
        log.Fatal(err)
    }
    // fecha conexão
    defer client.Close()

    var reply [][]string
    // Create request
    request := 10000
    for i := 0; i < request; i++ {
        // prepara request & start time
        t1 := time.Now()
        // invoca operação remota
        client.Call("DataReader.GetDataRPC", 2, &reply)

        requestTime = time.Now().Sub(t1)

        if(len(os.Args) > 1 && os.Args[1] == "teste"){
            log.Println(strconv.Itoa(int(requestTime.Nanoseconds())))
        }
    }
}
```

Clientes concorrentes

```
#!/bin/bash
#go run ../../TCP/server.go

go run  client.go &
go run  client.go &
go run  client.go &
go run  client.go &
go run  client.go &
go run  client.go &
go run  client.go "teste " &
go run  client.go &
go run  client.go &
go run  client.go &
go run  client.go &
go run  client.go &
go run  client.go &
go run  client.go &
go run  client.go &
```

RTT

```go
func client_RPC() {
    setUpLog()
    var requestTime time.Duration

    // conecta ao servidor
    client, err := rpc.Dial("tcp", "localhost:1313")
    if err != nil {
        log.Fatal(err)
    }
    // fecha conexão
    defer client.Close()

    var reply [][]string
    // Create request
    request := 10000
    for i := 0; i < request; i++ {
        // prepara request & start time
        t1 := time.Now()
        // invoca operação remota
        client.Call("DataReader.GetDataRPC", 2, &reply)

        requestTime = time.Now().Sub(t1)

        if(len(os.Args) > 1 && os.Args[1] == "teste"){
            log.Println(strconv.Itoa(int(requestTime.Nanoseconds())))
        }
    }
}
```

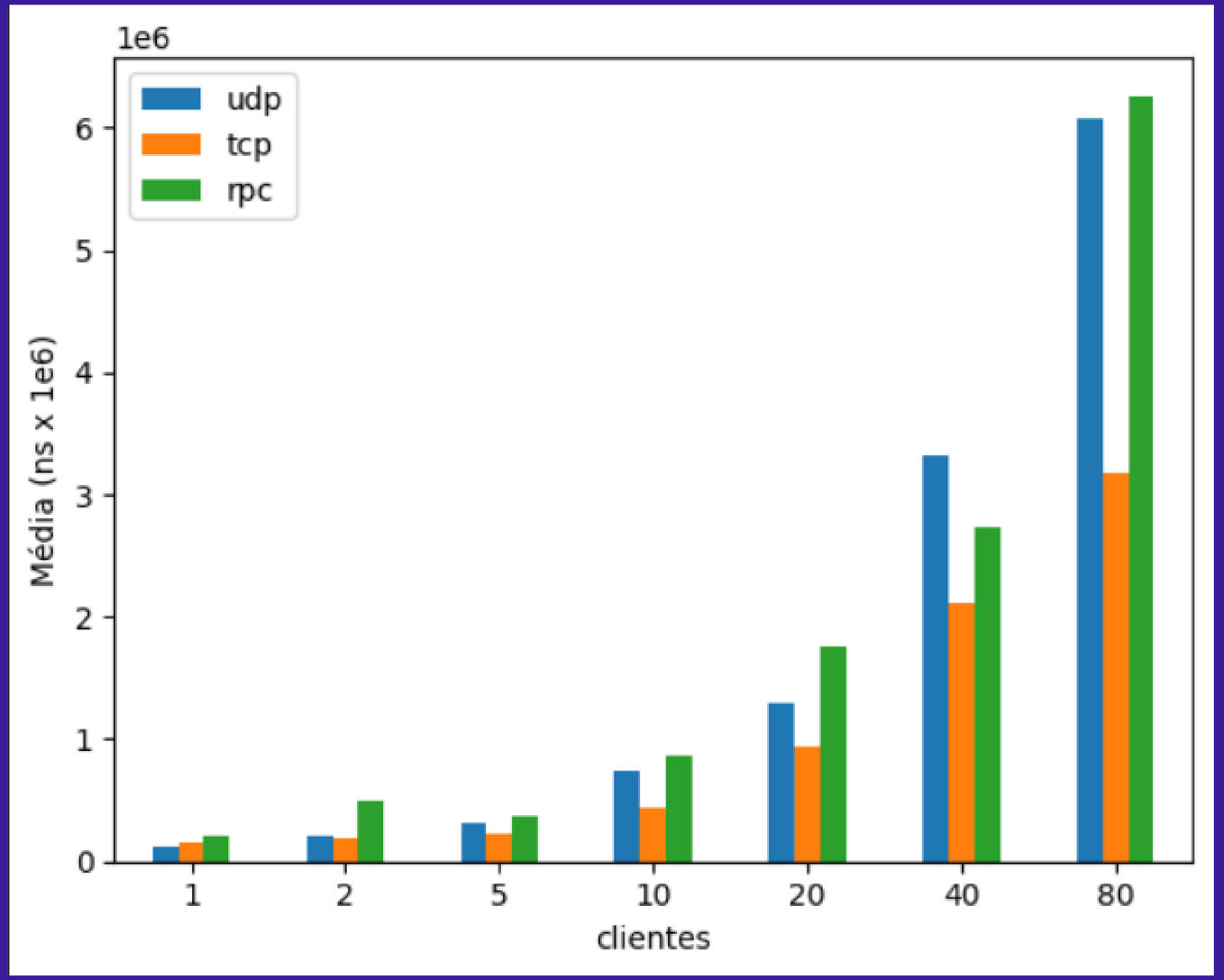
LOGS antes

```
2023/07/17 00:51:05.390690 client.go:86: 783000
2023/07/17 00:51:05.391170 client.go:86: 1723843
2023/07/17 00:51:05.391467 client.go:86: 1405223
2023/07/17 00:51:05.391492 client.go:86: teste 2311006
2023/07/17 00:51:05.391713 client.go:86: 5468708
2023/07/17 00:51:05.392117 client.go:86: 1348512
2023/07/17 00:51:05.392706 client.go:86: 417725
2023/07/17 00:51:05.393193 client.go:86: 440913
2023/07/17 00:51:05.393431 client.go:86: 1850188
2023/07/17 00:51:05.393734 client.go:86: 232015
2023/07/17 00:51:05.393824 client.go:86: 4093237
2023/07/17 00:51:05.394145 client.go:86: 910531
2023/07/17 00:51:05.394454 client.go:86: 4128787
2023/07/17 00:51:05.394933 client.go:86: 1097848
2023/07/17 00:51:05.395256 client.go:86: 224263
2023/07/17 00:51:05.395549 client.go:86: 187666
2023/07/17 00:51:05.395715 client.go:86: 3951527
2023/07/17 00:51:05.396840 client.go:86: 2618939
2023/07/17 00:51:05.397144 client.go:86: 709177
2023/07/17 00:51:05.397545 client.go:86: 632840
2023/07/17 00:51:05.397991 client.go:86: 721469
2023/07/17 00:51:05.398234 client.go:86: 207920
2023/07/17 00:51:05.398452 client.go:86: 192415
2023/07/17 00:51:05.398805 client.go:86: 327490
2023/07/17 00:51:05.399097 client.go:86: 7870438
2023/07/17 00:51:05.399108 client.go:86: 258137
2023/07/17 00:51:05.399320 client.go:86: 186967
2023/07/17 00:51:05.399405 client.go:86: 5512917
2023/07/17 00:51:05.399721 client.go:86: 227196
2023/07/17 00:51:05.400030 client.go:86: 275667
2023/07/17 00:51:05.400287 client.go:86: 208897
2023/07/17 00:51:05.400685 client.go:86: teste 9078428
2023/07/17 00:51:05.401080 client.go:86: 761698
2023/07/17 00:51:05.401062 client.go:86: 644852
2023/07/17 00:51:05.401366 client.go:86: 2223075
2023/07/17 00:51:05.401491 client.go:86: 332100
2023/07/17 00:51:05.401611 client.go:86: 201355
2023/07/17 00:51:05.401713 client.go:86: 575360
2023/07/17 00:51:05.401686 client.go:86: teste 943078
2023/07/17 00:51:05.401778 client.go:86: 2308003
2023/07/17 00:51:05.401956 client.go:86: 7454109
2023/07/17 00:51:05.402227 client.go:86: 399147
2023/07/17 00:51:05.402272 client.go:86: 486311
2023/07/17 00:51:05.402444 client.go:86: 799343
2023/07/17 00:51:05.402571 client.go:86: 1048330
2023/07/17 00:51:05.402787 client.go:86: 302137
```

LOGS agora

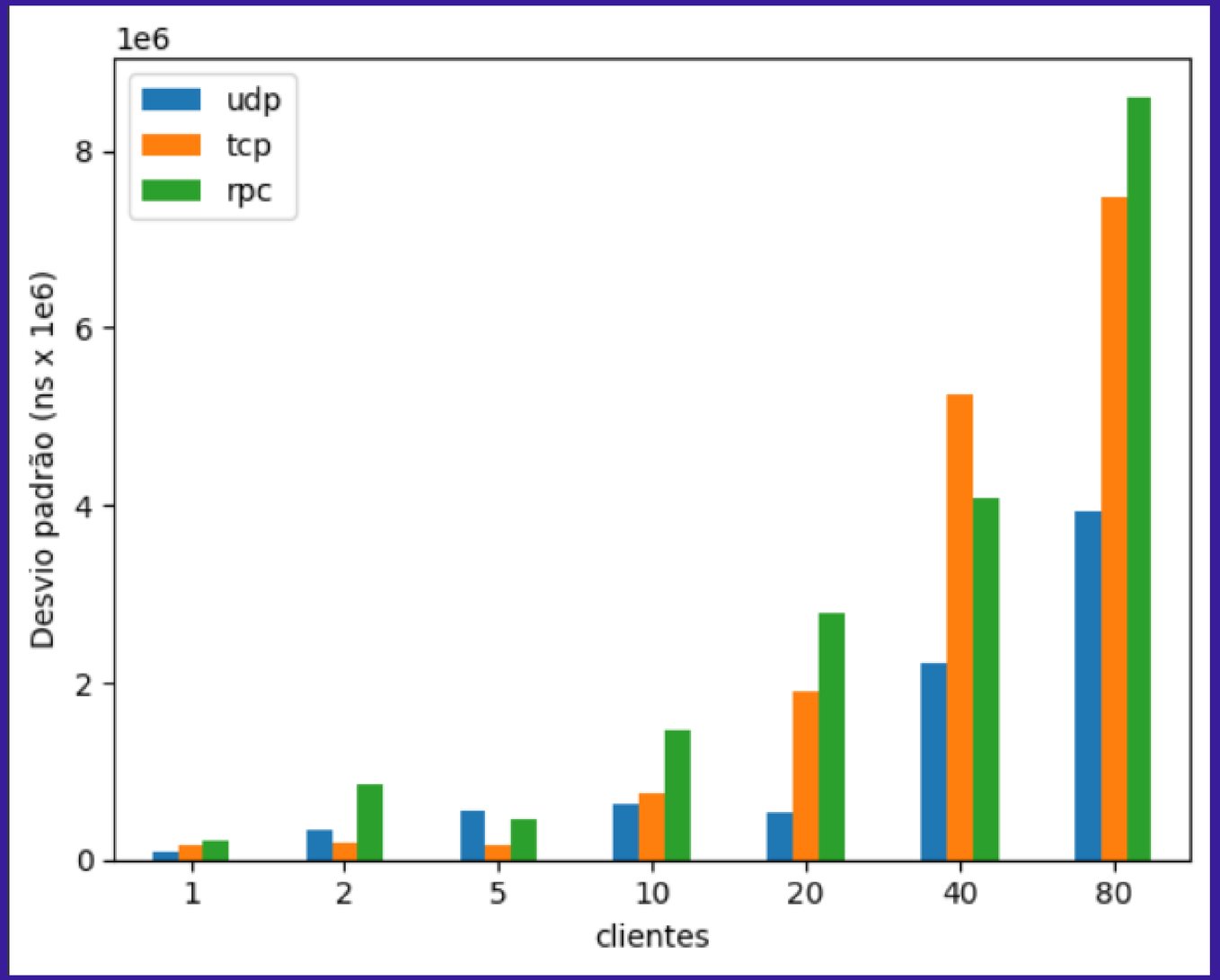
```
 2  431757
 3  311909
 4  418207
 5  302760
 6  514798
 7  2478305
 8  409967
 9  348925
10  349135
11  374207
12  301643
13  305624
14  318684
15  411153
16  228939
17  319383
18  624727
19  370366
20  324481
21  529464
22  252195
23  210152
24  286278
25  466538
26  330836
27  333491
28  293053
29  409687
30  274684
31  278525
```

Avaliações comparativas de desempenho



https://colab.research.google.com/drive/1DYFri6aCgxuSkZPrR12UcgNm-igwaGG3?usp=sharing

Avaliações comparativas de desempenho



https://colab.research.google.com/drive/1DYFri6aCgxuSkZPrR12UcgNm-igwaGG3?usp=sharing