IF711 - Atividade 05

Aplicação Cliente/Servidor com gRPC e Go RPC

Equipe 05:
Diego Rocha (djaar)
Ricardo Bizerra (rblf)

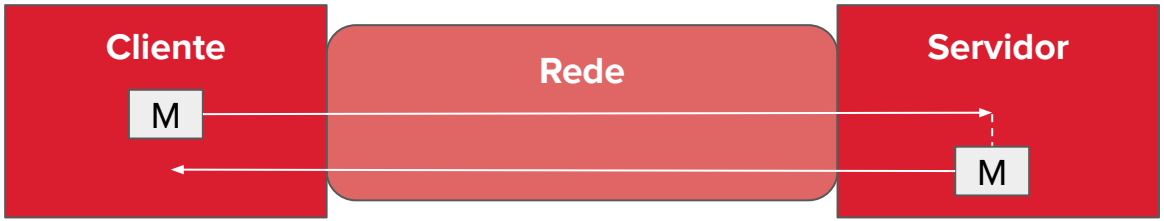
Centro de Informática UFPE
Universidade Federal de Pernambuco

Objetivo

- Implementar um servidor que hospeda um serviço que fornece a operação **Mul(m1,m2)**, onde m1 e m2 são matrizes.
- Realizar avaliação de desempenho utilizando a métrica **RTT (Round Trip Time)**, calculado no cliente.

Serviços implementados

- **Base:** aplicação cliente-servidor de troca de mensagens
  - Implementação via RPC
  - Protocolos gRPC e Go-RPC



Serviços implementados

- Estruturas gerais
  - Requisição e Resposta do servidor

```
3  type Request struct {
4      Operation string
5      A         [][]int
6      B         [][]int
7  }
```

```
3  type Reply struct {
4      R [][]int
5  }
```

Serviços implementados

- Multiplicador de matrizes

```
3   func Multiply(a, b [][]int) [][]int {
8     c := make([][]int, len(a))
9     for i := range c {
10      c[i] = make([]int, len(b[0]))
11    }
12
13    for i := 0; i < len(a); i++ {
14      for j := 0; j < len(b[0]); j++ {
15        for k := 0; k < len(b); k++ {
16          c[i][j] += a[i][k] * b[k][j]
17        }
18      }
19    }
20
21    return c
22  }
```

Serviços implementados

- Cliente Go RPC
  - Conexão com o servidor RPC

```
9   func Client(invocations int, a [][]int, b [][]int) {
12    client, err := rpc.Dial("tcp", "rpc-server:8080")
13
14    if err ≠ nil {
15      panic(err)
16    }
17
18    defer client.Close()
```

Serviços implementados

- Cliente Go RPC
  - Chamada ao procedimento e cálculo de RTT

```
9   func Client(invocations int, a [][]int, b [][]int) {
20    for i := 0; i < invocations; i++ {
21      msgToServer := shared.Request{Operation: "Mul", A: a, B: b}
22
23      startTime := time.Now()
24
25      err := client.Call("Matrix.Multiply", msgToServer, &response)
26
27      if err != nil {
28        panic(err)
29      }
30
31      elapsedTime := float64(time.Since(startTime).Nanoseconds()) / 1000000
32
33      shared.WriteRTTValue("/data/go-rpc-results.txt", elapsedTime)
34    }
35  }
```

Serviços implementados

- Servidor Go RPC
  - Registro do multiplicador de matrizes como serviço

```go
type MatrixService struct{}

func (s *MatrixService) Multiply(req shared.Request, res *shared.Reply) error {
  response := matrix.Multiply(req.A, req.B)

  res.R = response

  return nil
}
```

```go
func Server() {
  matrix_service := new(MatrixService)

  server := rpc.NewServer()
  server.RegisterName("Matrix", matrix_service)
```

Serviços implementados

- Servidor Go RPC
  - Listen com a porta 8080 e aceite de conexões

```
9   func Server() {
15    ln, err := net.Listen("tcp", "0.0.0.0:8080")
16    if err ≠ nil {
17      panic(err)
18    }
19
20    fmt.Println("Server listening on port 8080")
21
22    for {
23      conn, err := ln.Accept()
24
25      if err ≠ nil {
26        fmt.Println("Connection error:", err)
27        continue
28      }
29
30      go server.ServeConn(conn)
31    }
32  }
```

Serviços implementados

- .proto
  - Define o contrato entre o servidor e o cliente.
  - Define o formato do Request e do Reply

```
1   syntax = "proto3";
2
3   package grpc;
4   option go_package = "/grpc/grpc";
5   service MatrixMul {
6       rpc Mul (Request) returns (Reply) {}
7   }
8
9   message Request {
10      string Op = 1;
11      Matrix m1 = 2;
12      Matrix m2 = 3;
13  }
14
15  message Reply {
16      Matrix m = 1;
17  }
18
19  message Matrix {
20      repeated Row m = 1;
21  }
22
23  message Row {
24      repeated int32 row = 1;
25  }       You, 3 hours ago • feat: grcp client
```

Serviços implementados

- Traduz entre o Matrix pelo protocol buffer e array de inteiros e vice-versa

```go
package sharedGrpc

import (
    pb "exercicio-05-djaar-rblf/grpc/grpc"
)

func ParseMatrix(m [][]int32) *pb.Matrix {
    rows := make([]*pb.Row, 0)
    for i := range m {
        rows = append(rows, &pb.Row{Row: m[i]})
    }
    return &pb.Matrix{M: rows}
}

func UnparseMatrix(m *pb.Matrix) [][]int32 {
    result := make([][]int32, 0)
    matrix := m.GetM()
    for i := range matrix {
        row := matrix[i]
        result = append(result, row.Row)
    }
    return result
}
```

Serviços implementados

- Servidor gRPC
  - Inicia a conexão com os clientes

```
41  func Server() {
42      lis, err := net.Listen("tcp", "0.0.0.0:8080")
43      if err ≠ nil {
44          log.Fatalf("failed to listen: %v", err)
45      }
46      s := grpc.NewServer()
47      pb.RegisterMatrixMulServer(s, &server{})
48      log.Printf("server listening at %v", lis.Addr())
49      if err := s.Serve(lis); err ≠ nil {
50          log.Fatalf("failed to serve: %v", err)
51      }
52  }
53
```

Serviços implementados

- Servidor gRPC
  - Expõe a função Mul para os clientes

```
27  // Mul implements grpc.MatrixMulServer
28  func (s *server) Mul(_ context.Context, in *pb.Request) (*pb.Reply, error) {
29      op := in.GetOp()
30      // log.Printf("Received: %v", op)
31      if op ≠ "Mul" {
32          return nil, &OpError{"Wrong Operation"}
33      }
34      matrix1 := sharedGrpc.UnparseMatrix(in.GetM1())
35      matrix2 := sharedGrpc.UnparseMatrix(in.GetM2())
36      matrix_res := matrix.Multiply32(matrix1, matrix2)
37      matrix_parse := sharedGrpc.ParseMatrix(matrix_res)
38      return &pb.Reply{M: matrix_parse}, nil
39  }
40
```

Serviços implementados

- Cliente gRPC
  - Inicializa conexão com o servidor

```
15  func Client(invocations int, a [][]int32, b [][]int32) {
16      // Set up a connection to the server.
17      conn, err := grpc.NewClient("rpc-server:8080", grpc.WithTransportCredentials(insecure.NewCredentials()))
18      if err ≠ nil {
19          log.Fatalf("did not connect: %v", err)
20      }
21      defer conn.Close()
22      c := pb.NewMatrixMulClient(conn)
23
```

Serviços implementados

- Cliente gRPC
  - Invoca a função Mul, que realiza uma requisição no servidor

```
15  func Client(invocations int, a [][]int32, b [][]int32) {
24      for i := 0; i < invocations; i++ {
25          startTime := time.Now()
26          // Contact the server and print out its response.
27          ctx, cancel := context.WithTimeout(context.Background(), 10*time.Second)
28          defer cancel()
29          matrix1 := sharedGrpc.ParseMatrix(a)
30          matrix2 := sharedGrpc.ParseMatrix(b)
31          r, err := c.Mul(ctx, &pb.Request{Op: "Mul", M1: matrix1, M2: matrix2})
32          if err ≠ nil {
33              log.Fatalf("could not multiply: %v", err)
34          }
35          _ = sharedGrpc.UnparseMatrix(r.GetM())
36          //fmt.Printf("%v\n", matrix_res)
37
38          // Tempo em milisegundos mais preciso
39          elapsedTime := float64(time.Since(startTime).Nanoseconds()) / 1000000
40
41          shared.WriteRTTValue("/data/grpc-results.txt", elapsedTime)
42      }
43  }
44
```

Métrica

- RTT médio, mediano e desvio padrão
  - Round Trip Time médio em 10000 execuções do programa, para cada instância de cliente

Parâmetros

Parâmetro	Tipo	Valor
Número de invocações	Carga de Trabalho	10.000 invocações
Processador	Sistema	9th Gen Intel® Core™ i3-9100F @ 3.60GHz × 4 núcleos
Memória RAM	Sistema	16 GB
Sistema operacional	Sistema	Kubuntu 22.04
Linguagem de programação	Sistema	Go 1.23.6
Interfaces de rede	Sistema	Ligada
Fonte de alimentação	Sistema	Rede elétrica
Processos em execução	Sistema	Apenas os estritamente necessários à realização do experimento

Fatores

Fator	Nível
Dimensão da matriz	20x20
Valores da matriz	Valor aleatório entre 0 e 99 para cada célula
Número de clientes simultâneos	1, 10 e 20
Protocolos de rede	gRPC e Go-RPC
Tipo de rede	Localhost
GOMAXPROCS	4
Serialização	Nativa de cada protocolo

Técnica de avaliação

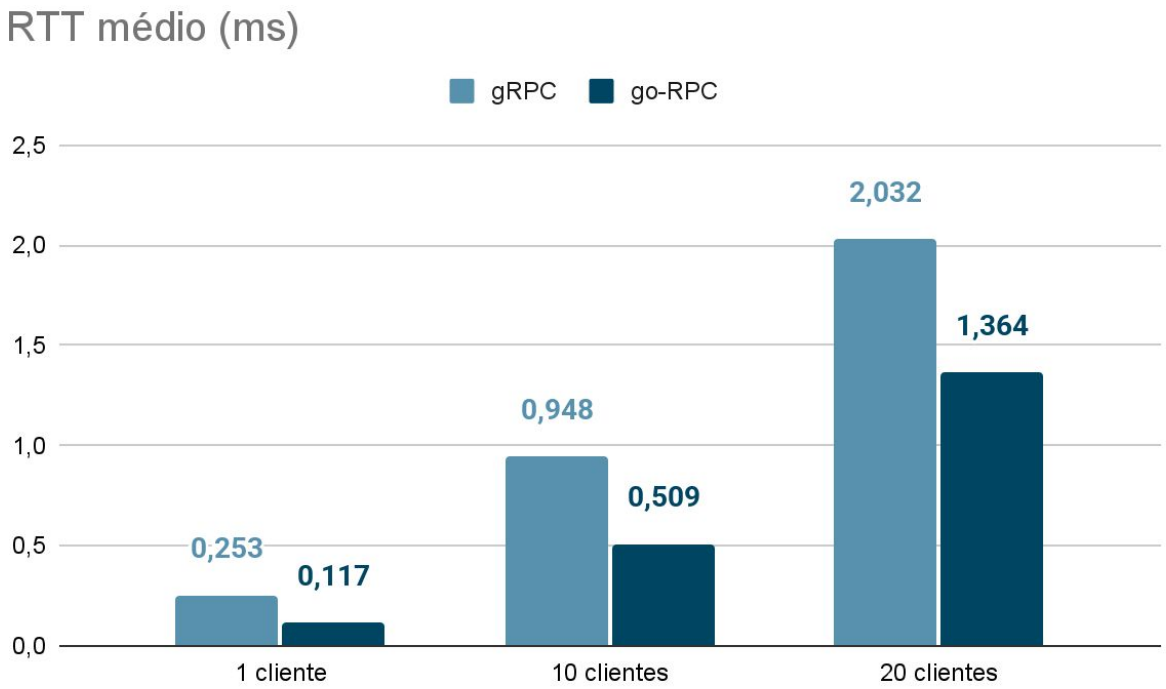
- Medição

Projeção do experimento

- Os experimentos serão realizados variando
  - Protocolo RPC
  - Número de clientes

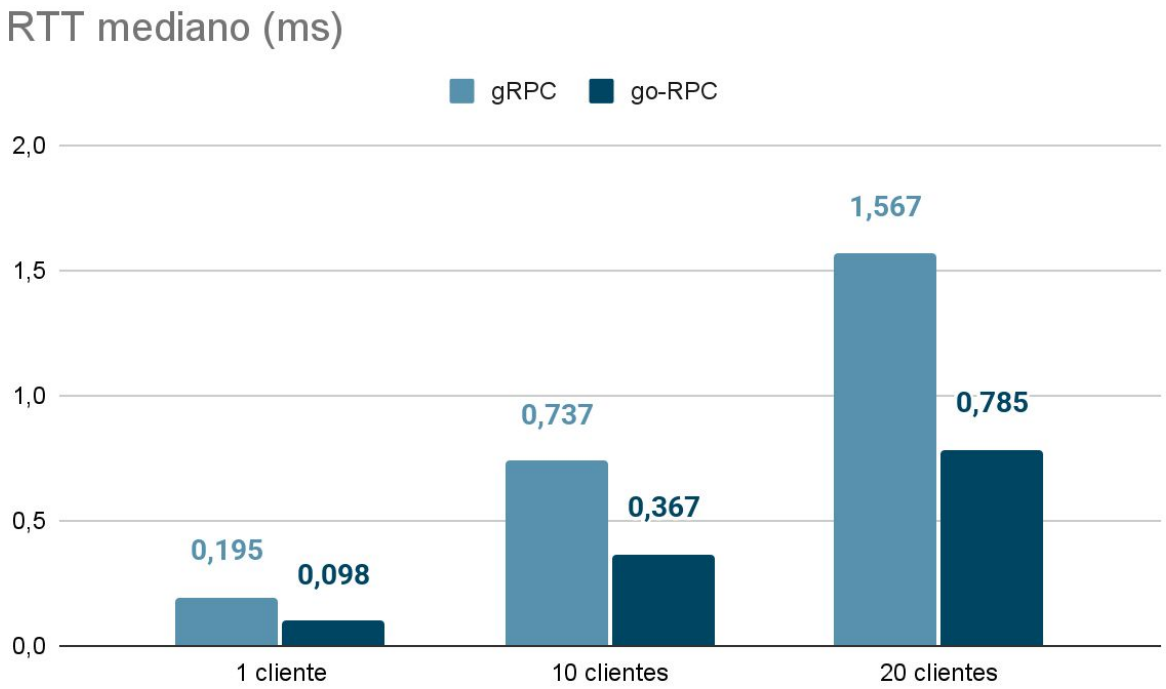
Resultados

Rede: Localhost



Resultados

Rede: Localhost



Resultados

Rede: Localhost