IF711 - Program. Concorrente Distribuída - T01 (2024.2)

Grupo 09

Erbert B. G. Rocha
ebgr@cin.ufpe.br

Centro de Informática UFPE
Universidade Federal de Pernambuco

O Problema

5. Implementar um programa que lê múltiplos arquivos e conta o número de linhas em cada arquivo.

Estrutura gRPC

```
syntax = "proto3";
package grpcarquivo;
option go_package = "./grpcarquivo";

service ArquivoService
   rpc CountLines(Request) returns (Response);

message Request
   string content = 1;

message Response
   int32 lines = 1;
```

Clients

```
func clientGO(client pb.ArquivoServiceClient, wg *sync.WaitGroup, start <-chan struct{}) {
    defer wg.Done()
    <-start

    for i := 0; i < count; i++ {
        start := time.Now()
        callRemote(message, client)
        delta := time.Since(start) / time.Nanosecond
        fmt.Println(strconv.FormatInt(delta.Nanoseconds(), 10))
    }
}
```

Clients

```go
func callRemote(message string, client *rpc.Client) {
   request := Request{Content: message}
   var response int
   err := client.Call("Arquivo.CountLines", request, &response)
   if err != nil { print(err) }
}
```

```go
func callRemote(message string, client pb.ArquivoServiceClient) {
   req := &pb.Request{Content: message}
   _, err := client.CountLines(ctx, req)
   if err != nil { print(err) }
}
```

Clients

```
func main() {
   message = readFile(os.Args[2])
   var wg sync.WaitGroup
   start := make(chan struct{})

   for i := 0; i < clients; i++ {
       conn, err := // Criar Conexão
       defer conn.Close()
       client := pb.NewArquivoServiceClient(conn)
       go clientGO(client, &wg, start, i+1)
       wg.Add(1)
   }
   time.Sleep(1 * time.Second)   close(start)
   wg.Wait()
   fmt.Println()
}
```

Server gRPC

```go
func (s *server) CountLines(ctx context.Context, req *pb.Request) (*pb.Response, error) {
   return &pb.Response{Lines: 1 + int32(strings.Count(req.GetContent(), "\n"))}, nil
}

func main() {
   listener, err := net.Listen("tcp", ":1313")
   if err != nil { print(err) }

   pb.RegisterArquivoServiceServer(grpc.NewServer(), &server{})
   fmt.Println("Servidor gRPC rodando na porta 1313..." )

   err = s.Serve(listener);
   if err != nil { print(err) }
}
```

gRPC

Server goRPC

```go
func (a *Arquivo) CountLines(request *Request, response *int) error {
   *response = (1 + int(strings.Count(request.Content, "\n")));    return nil
}

func main() {
   arquivo := &Arquivo{}
   rpc.Register(arquivo)

   listener, err := net.Listen("tcp", "0.0.0.0:1313")
   if err != nil { print(err) }

   defer listener.Close()
   fmt.Printf("Servidor RPC rodando em ( %s)...\n", address)
   for {
       conn, err := listener.Accept()
       if err != nil { print(err) }
       go rpc.ServeConn(conn)
   }
}
```

Equipamento testado

Copy

System Details

Hardware Information

Model
Acer Aspire A315-42G

Memory
8.0 GiB

Processor
AMD Ryzen™ 7 3700U with Radeon™ Vega Mobile Gfx × 8

Graphics
AMD Radeon™ Vega 10 Graphics

Graphics 1
AMD Radeon™ 540X Series

Disk Capacity
256.1 GB

Software Information

Firmware Version
V1.10

OS Name
Ubuntu 24.04.1 LTS

OS Type
64-bit

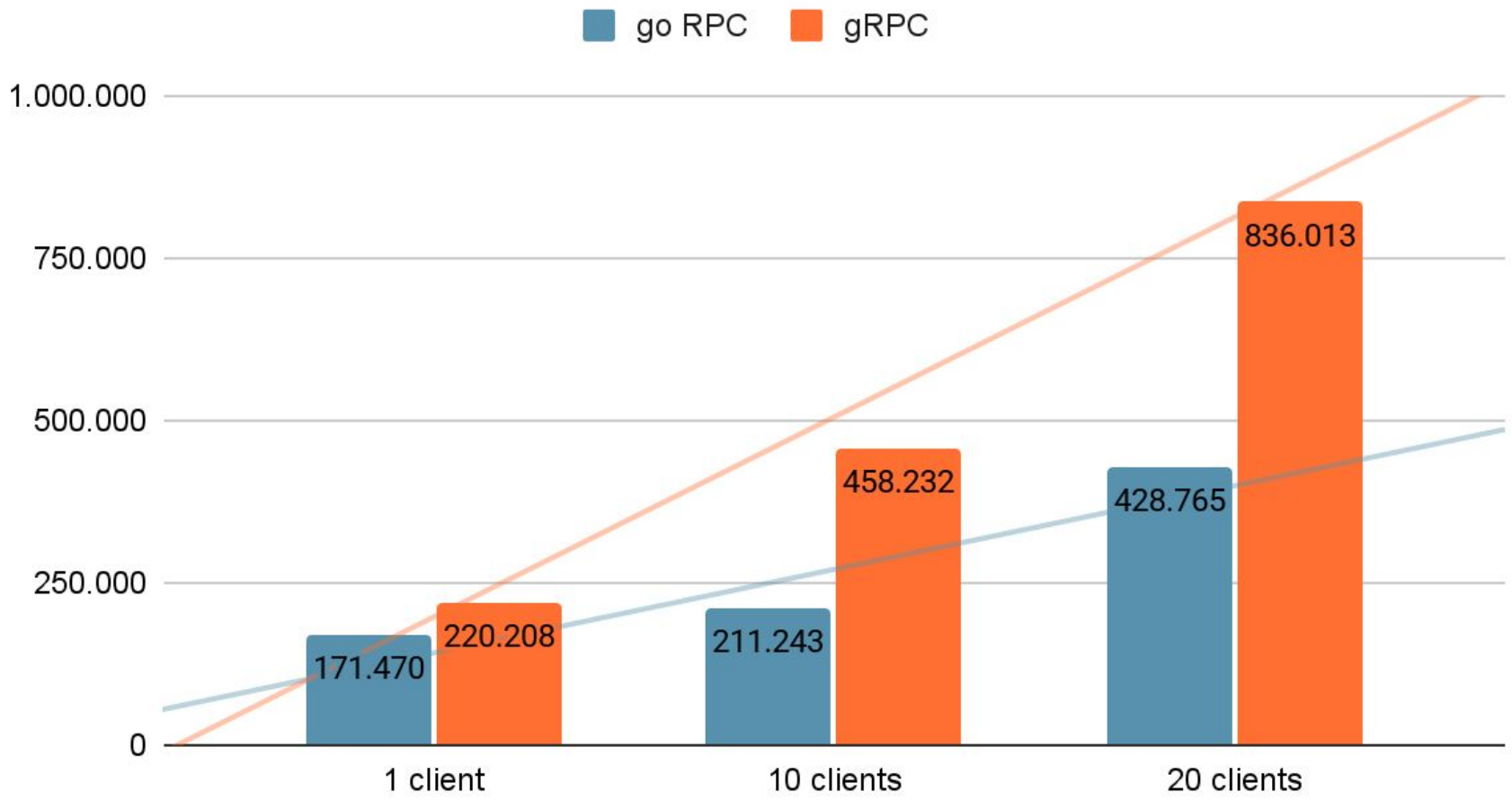
GNOME Version
46

Windowing System
Wayland

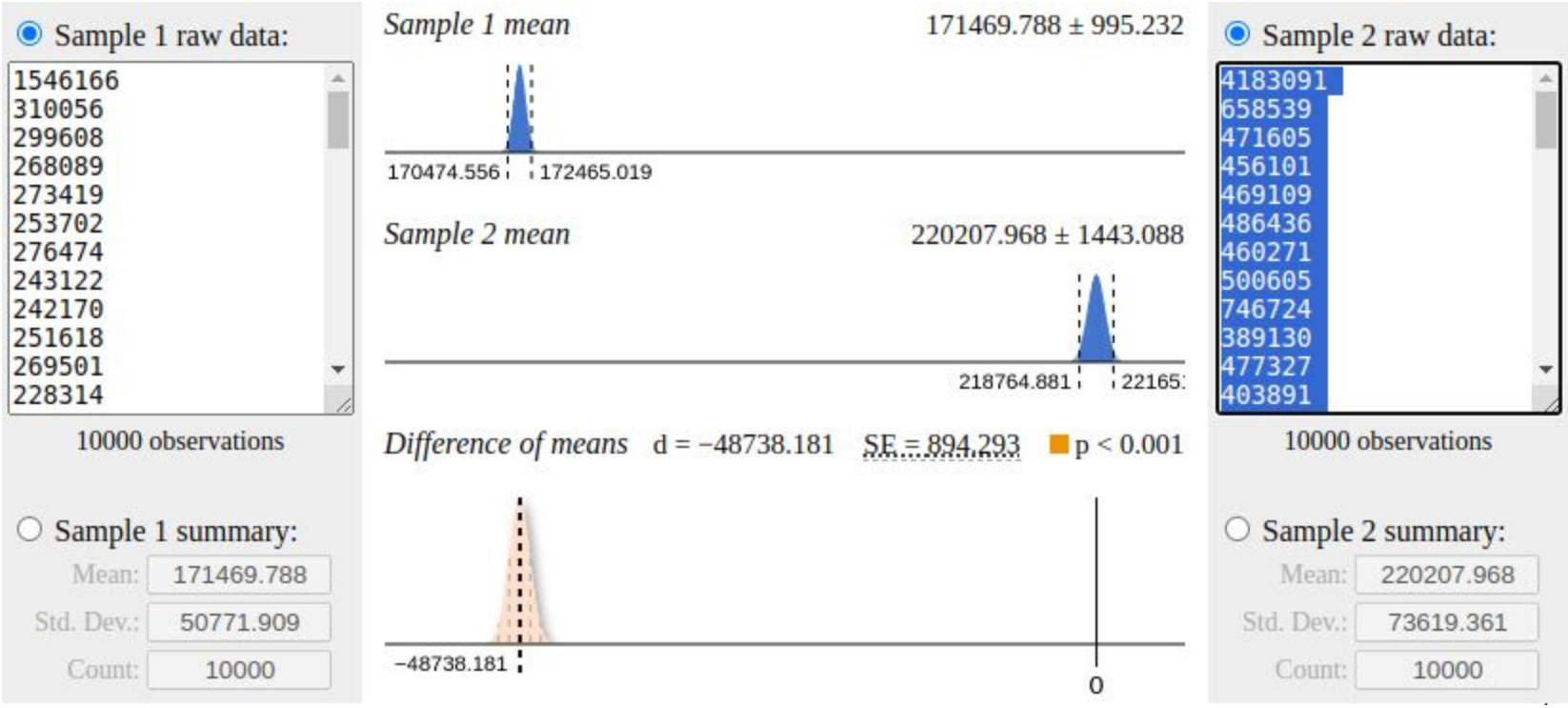
Kernel Version
Linux 6.8.0-52-generic

versão do go: go1.22.2 linux/amd64
alimentação: *tomada*
conectividade: *modo avião*

go RPC e gRPC

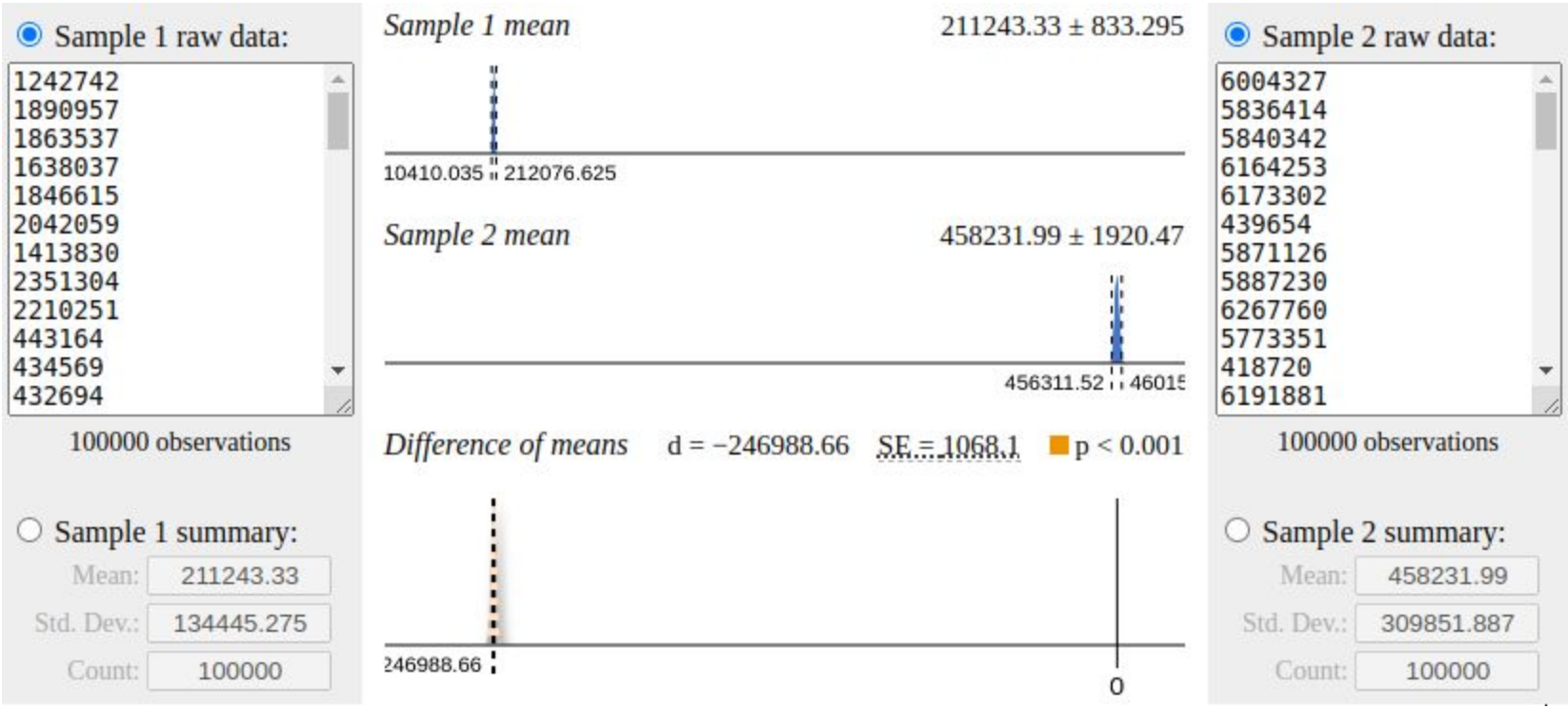

go RPC
gRPC
1.000.000
750.000
500.000
250.000
0
171.470
220.208
211.243
458.232
428.765
836.013
1 client
10 clients
20 clients

t-test(goRPC e gRPC) - 1 Cliente



https://www.evanmiller.org/ab-testing/t-test.html#!171469.7875/50771.909106/10000;220207.9683/73619.360874/10000@95

t-test(goRPC e gRPC) - 10 Clientes



https://www.evanmiller.org/ab-testing/t-test.html#!211243.32971/134445.275075/100000;458231.98997/309851.886753/100000@95

t-test(goRPC e gRPC) - 20 Clientes

https://www.evanmiller.org/ab-testing/t-test.html#!428765.41202/341656.18045/200000;836012.874025/578092.705673/200000@95

bibliografia

- **A basic tutorial introduction to gRPC in Go.**
  - https://grpc.io/docs/languages/go/basics/
- **Two Saple T-Test**
  - https://www.evanmiller.org/ab-testing/t-test.html
- **Códigos apresentados**
  - https://github.com/erbert-gadelha/go-files