Exercício 04 - Mensageria

Victor Nicéas e Lucas Mota

Visão geral

Usamos o rabbitMQ.

Cliente/Servidor.

Aplicação calculadora.

Usamos duas filas, uma para request e a outra para response e um broker de mensagens.

Cliente

```
conn, err := amqp.Dial("amqp://guest:guest@localhost:5672/")
ch, err := conn.Channel()

requestQueue, err := ch.QueueDeclare(
    "request", false, false, false, false, nil)

replyQueue, err := ch.QueueDeclare(
    "response", false, false, false, false, nil)

QueueDeclare(name string, durable,
autoDelete, exclusive, noWait bool, args Table) (Queue, error)
```

Cliente

```
start := time.Now()

msgRequest := shared.Request{Op: "add", P1: i, P2: i}
msgRequestBytes, err := json.Marshal(msgRequest)

err = ch.Publish("", requestQueue.Name, false, false,

amqp.Publishing{ContentType: "text/plain", Body:
msgRequestBytes})

x := <-msgsFromServer

json.Unmarshal(x.Body, &result)
executionTime := time.Since(start)
```

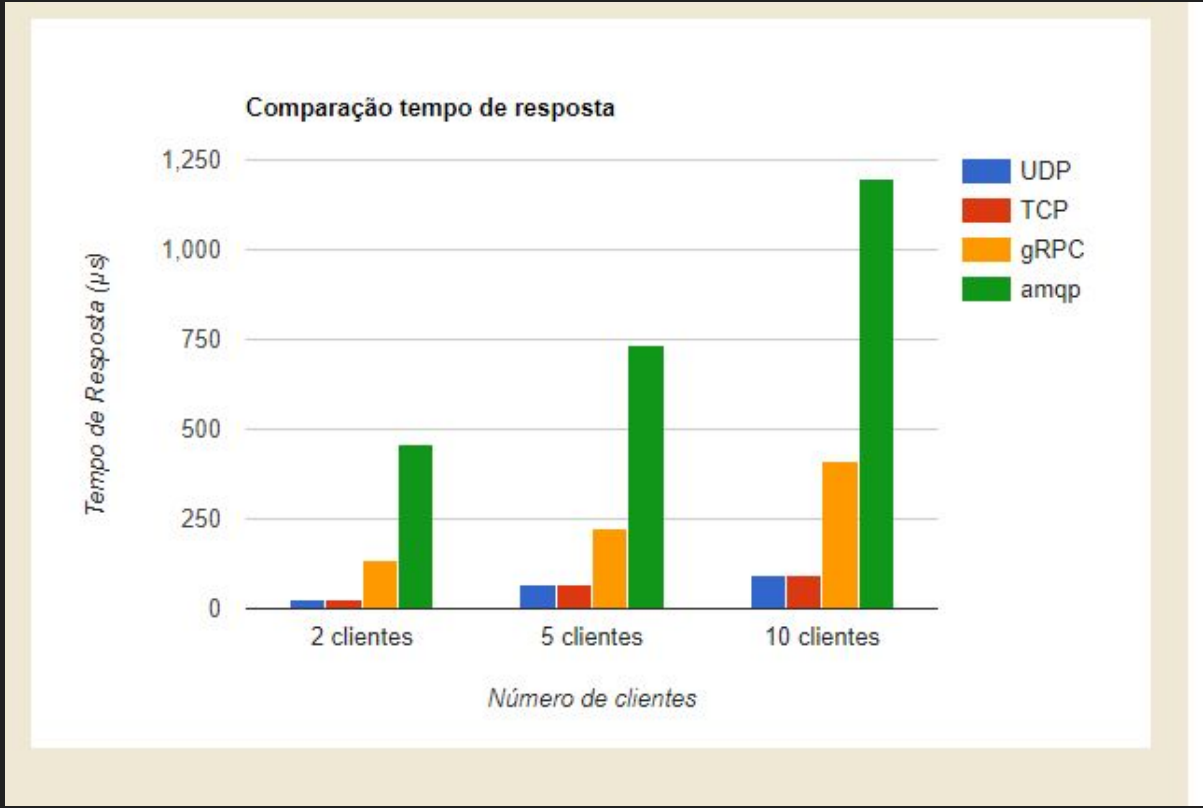
Servidor

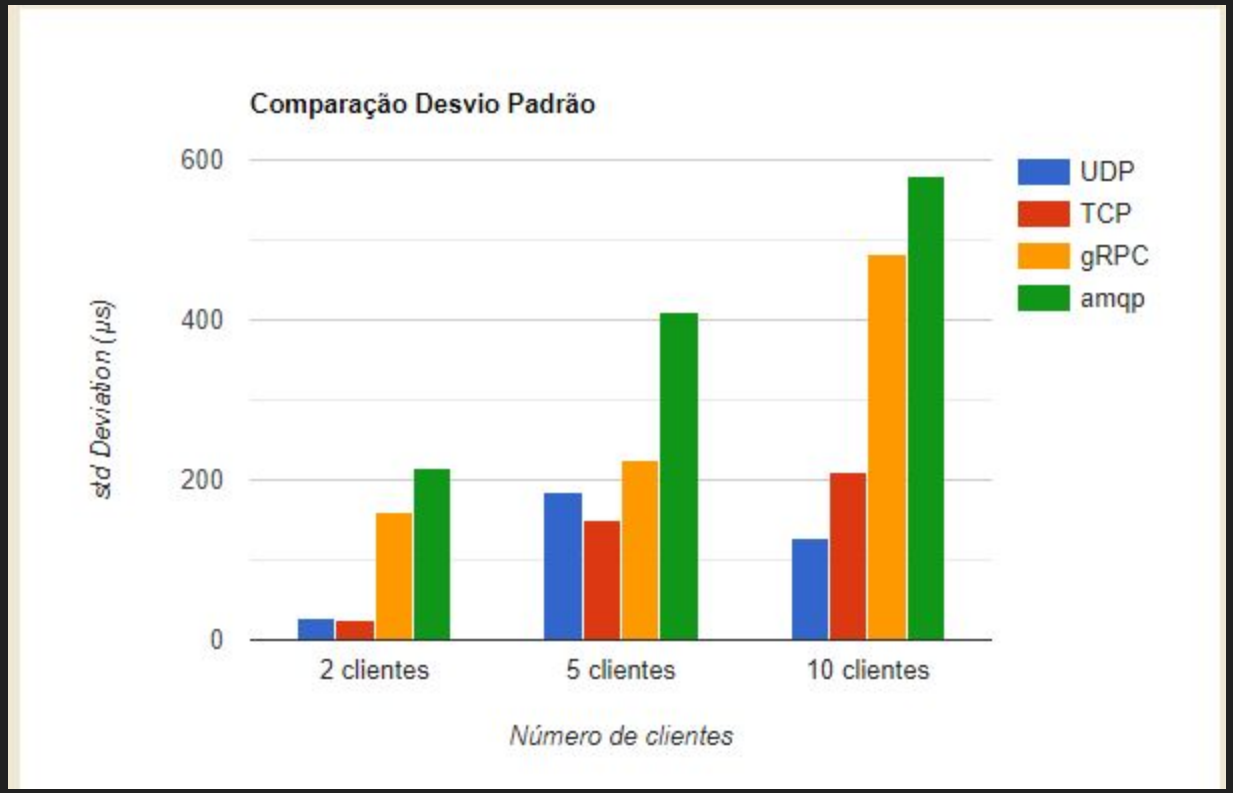
```
conn, err := amqp.Dial("amqp://guest:guest@localhost:5672/")

ch, err := conn.Channel()

requestQueue, err := ch.QueueDeclare("request", false, false, false,
    false, nil)
replyQueue, err := ch.QueueDeclare("response", false, false, false,
    false, nil)

msgsFromClient, err := ch.Consume(requestQueue.Name, "", true, false,
    false, false, nil)
```

Servidor

```
msgRequest := shared.Request{}

err := json.Unmarshal(d.Body, &msgRequest)
r := impl.Calculadora{}.InvocaCalculadora(msgRequest)

replyMsg := shared.Reply{Result: []interface{}{r}}
replyMsgBytes, err := json.Marshal(replyMsg)

err = ch.Publish("", replyQueue.Name, false, false,
amqp.Publishing{ContentType: "text/plain", Body: []byte(replyMsgBytes)})
```

Comparações




Comparação Desvio Padrão
600
400
200
0
std Deviation (µs)
UDP
TCP
gRPC
amqp
2 clientes
5 clientes
10 clientes
Número de clientes