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.

utilizar serviço de mensageria na troca de mensagens

Integração com o Broker [1/3]

struct **Connection**	
Message	chan [] byte
MessageHandler	func ([]byte)
Publish(queue string, msg[]byte)	void
Subscribe(queue string)	void
CreateQueue(queue string)	void
Disconnect()	void
NewConnection(url string, id string)	*Connection

```go
func (c *Connection) Subscribe(queue string) {
    msgs := newConsumer(c.ch, queue)
    fmt.Printf("inscrito em <%s>.", queue)
    go (func() {
        for {
            msg := <-msgs
            c.Message <- msg.Body
        }
    })()
}
```

Cliente RABBITMQ cria uma subrotina para cada tópico ouvido.

```go
func NewConnection(url string, id string) *Connection {
    conn := Connection{...}
    opts.SetDefaultPublishHandler(func(...) {
        conn.Message <- msg.Payload()
    })
    return &conn
}
```

Consumo do canal é gerenciado pelo MQTT.

Integração com o Broker [3/3]

```
import (
    "os"
    connection "server/connection_mqqt"
    //connection "server/connection_rabbitmq"
    util "server/util"
    "strings"
)
```

Server.go[1/2]

```go
func main() {
    conn := connection.NewConnection("servidor")
    defer conn.Disconnect()
    if len(os.Args) > 1 && os.Args[1] == "create" {
        conn.CreateQueue(util.Queue)
    }
    conn.Subscribe(util.Queue)
    for {
        msg := <-conn.Message
        go handleConnection(conn, msg)
    }
}
```

Server.go[2/2]

```go
func countLines(str string) int {
    return 1 + int(strings.Count(str, "\n"))
}

func handleConnection(conn *connection.Connection, msg []byte) {
    request := util.JsonToRequest(msg)
    response := util.ResponseToJson(
        util.Response{
            Lines: countLines(request.Content),
        })
    conn.Publish(request.ResponseTo, response)
}
```

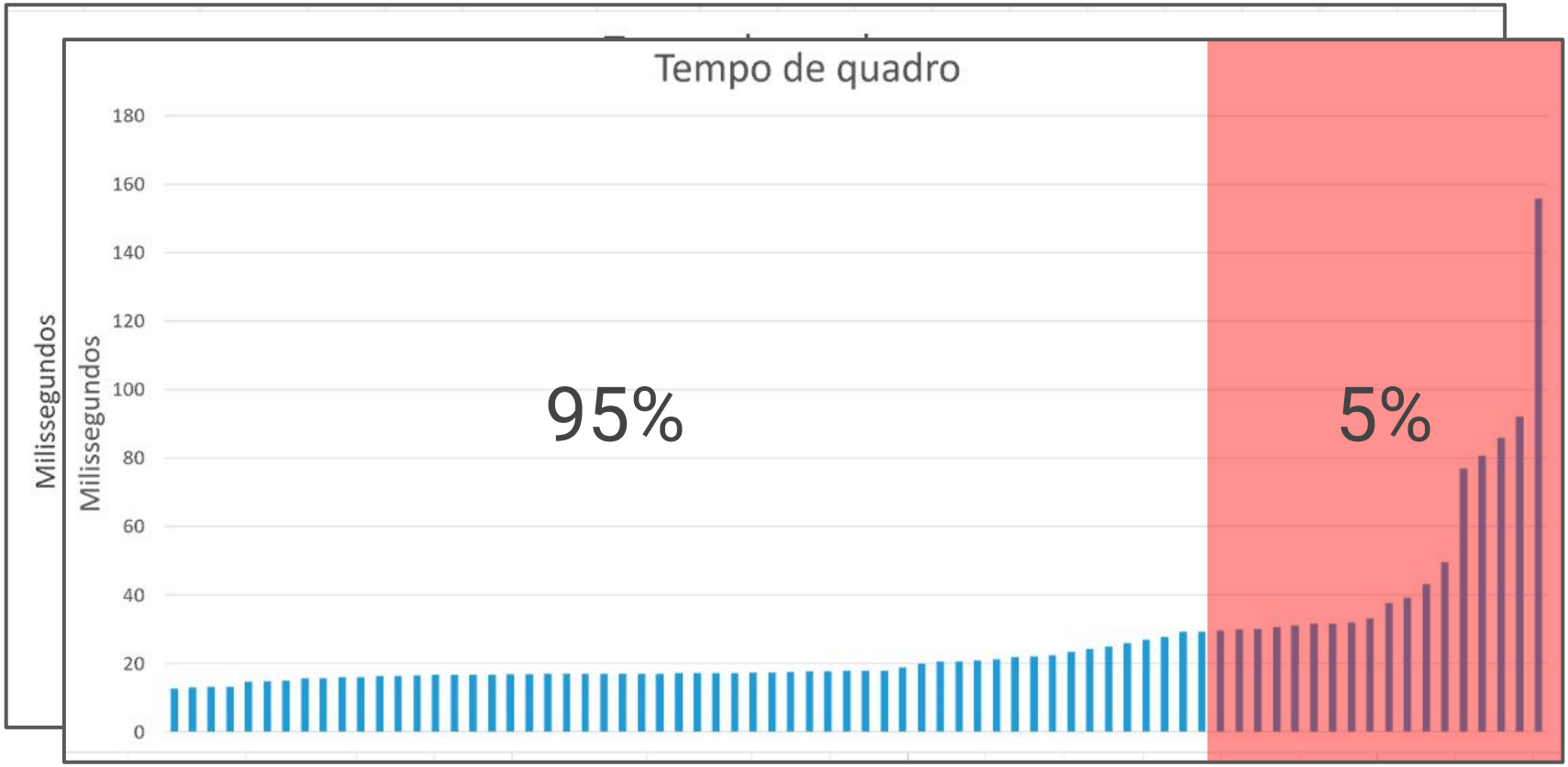
Client.go[1/2]

```go
func main() {
    handleArgs()
    var wg sync.WaitGroup
    start := make(chan struct{})
    for i := 0; i < clients; i++ {
        wg.Add(1)
        id := fmt.Sprintf("client_%d", i+1)
        conn := connection.NewConnection(id)
        topic_name := fmt.Sprintf("fila_%d", i+1)
        conn.CreateQueue(topic_name)
        conn.Subscribe(topic_name)
        go clientGO(*conn, topic_name, &wg, start)
    }
    close(start)
    wg.Wait()
    fmt.Println()
}
```

Client.go[2/2]

```go
func clientGO(conn connection.Connection, responseTo string, wg
    defer conn.Disconnect(); defer wg.Done();
    msgBytes := util.RequestToJson(
        util.Request {
            Content:    message,
            ResponseTo: responseTo,
        },
    )
    <-start
    for i := 0; i < count; i++ {
        start := time.Now()
        ////////////
        conn.Publish(util.Queue, msgBytes)
        <-conn.Message
        ////////////
        delta := time.Since(start) / time.Nanosecond
        fmt.Println(strconv.FormatInt(delta.Nanoseconds(), 10))
    }
}
```

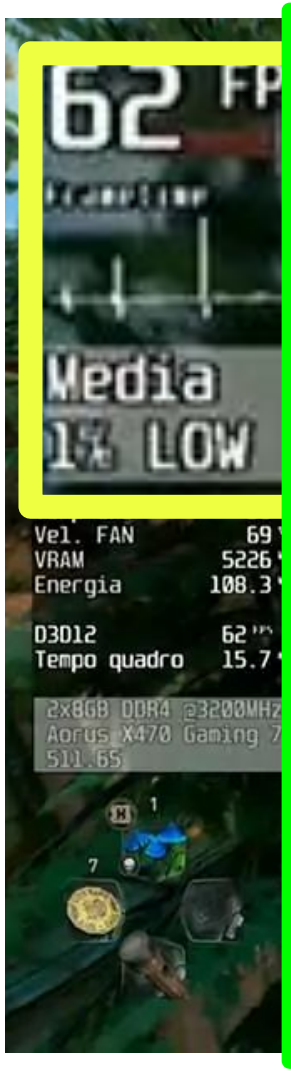
Percentil

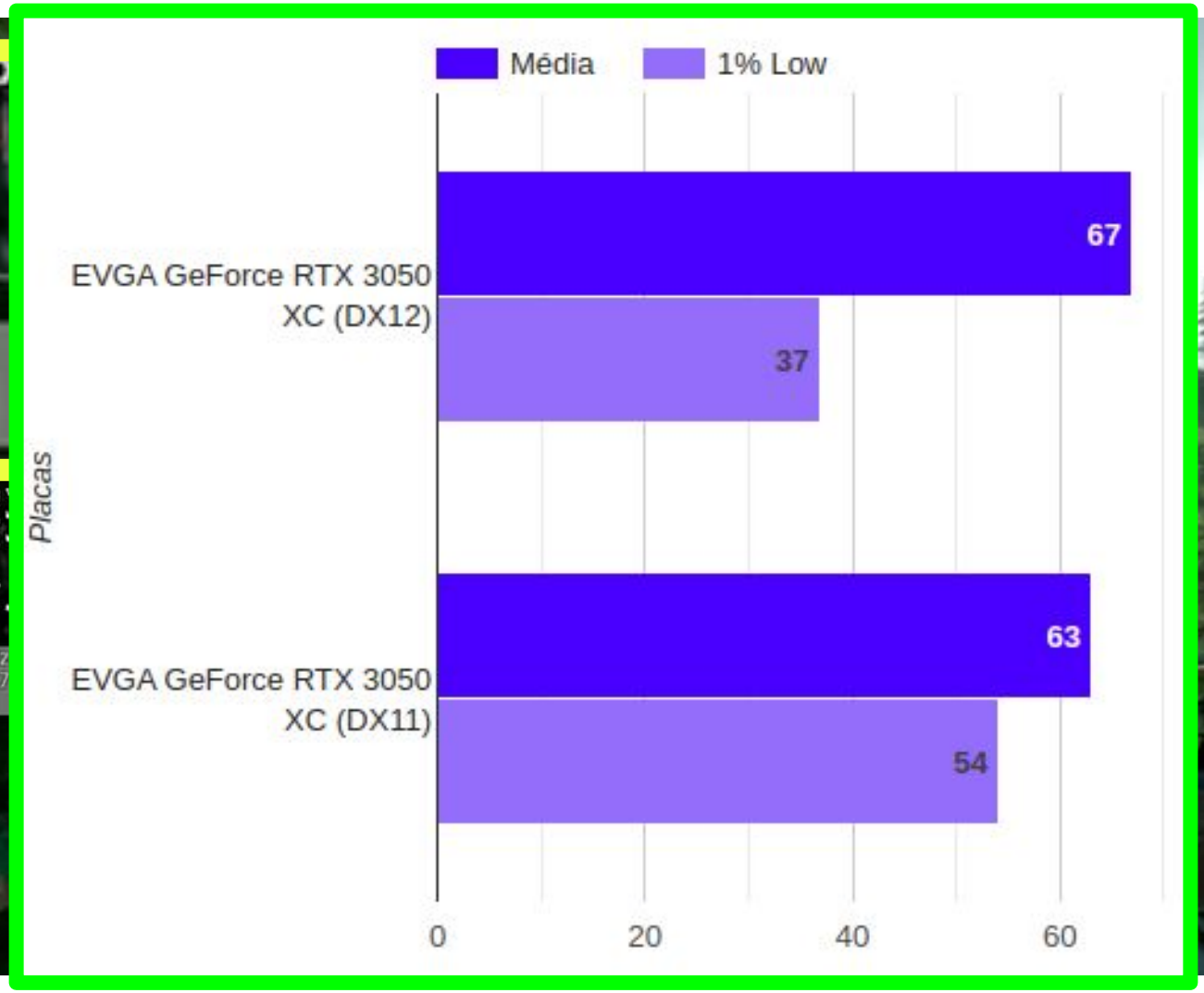


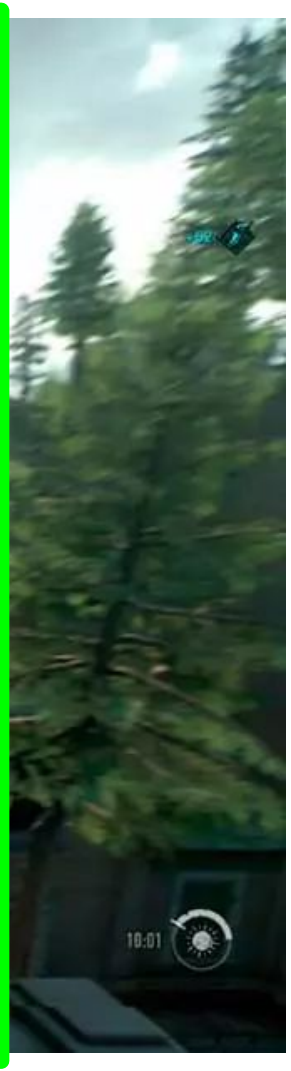
p99 | p95 | p90 ...

https://www.adrenaline.com.br/

1% Low







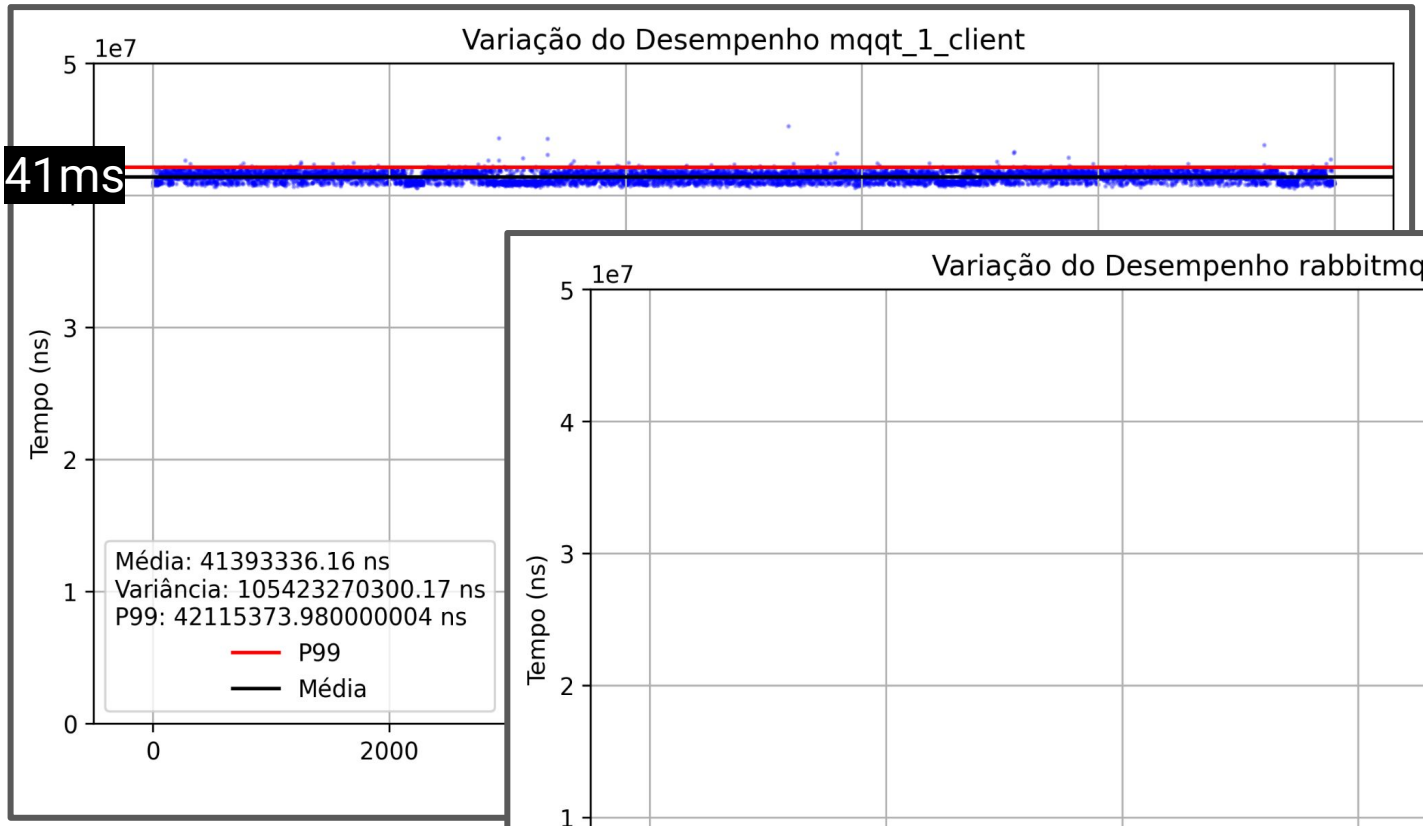
https://www.adrenaline.com.br/

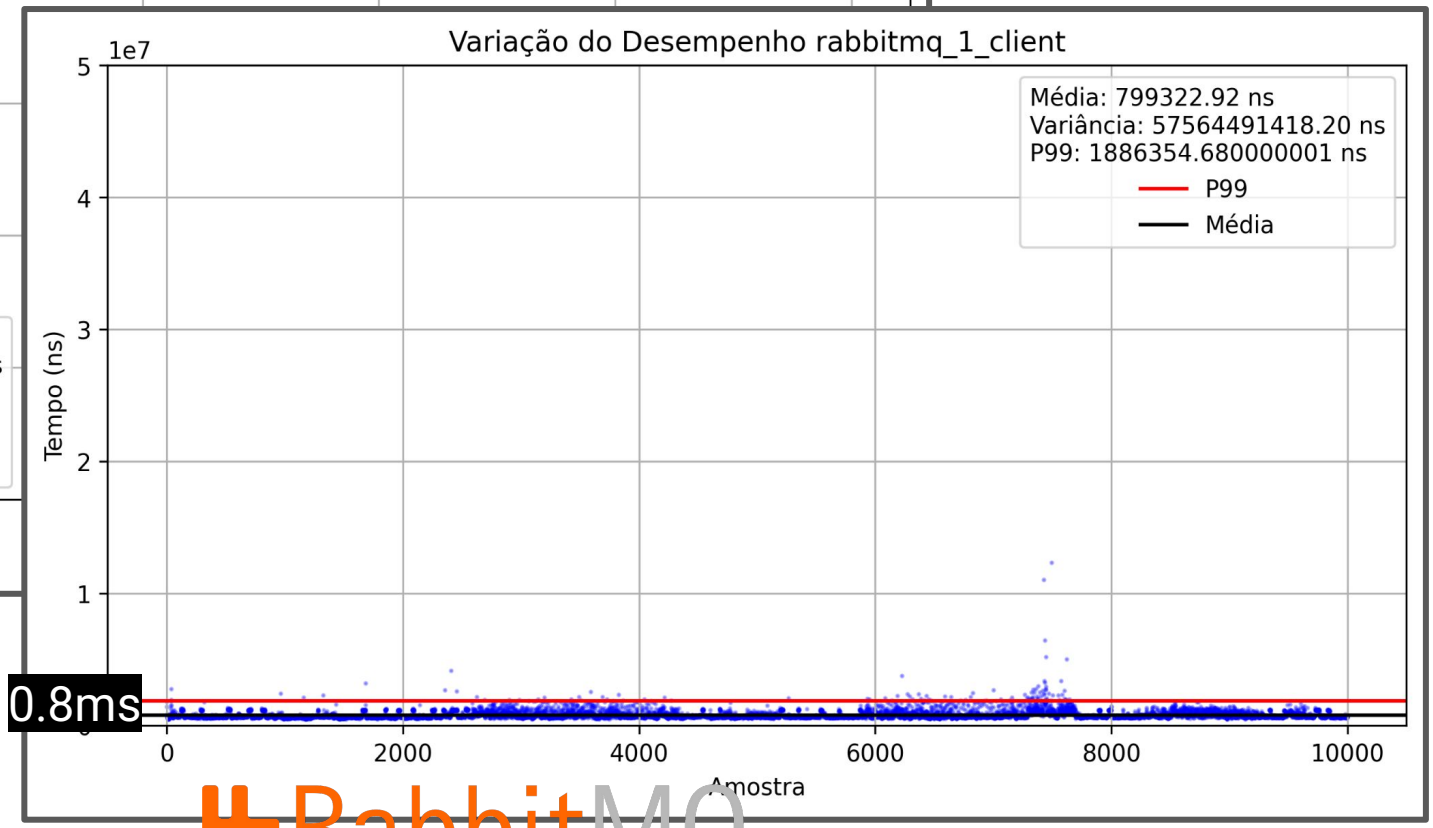
Configuração

Broker - Client - Server	Ubuntu 24.04.2	Ryzen 7 3700U	8gb	100Mbps

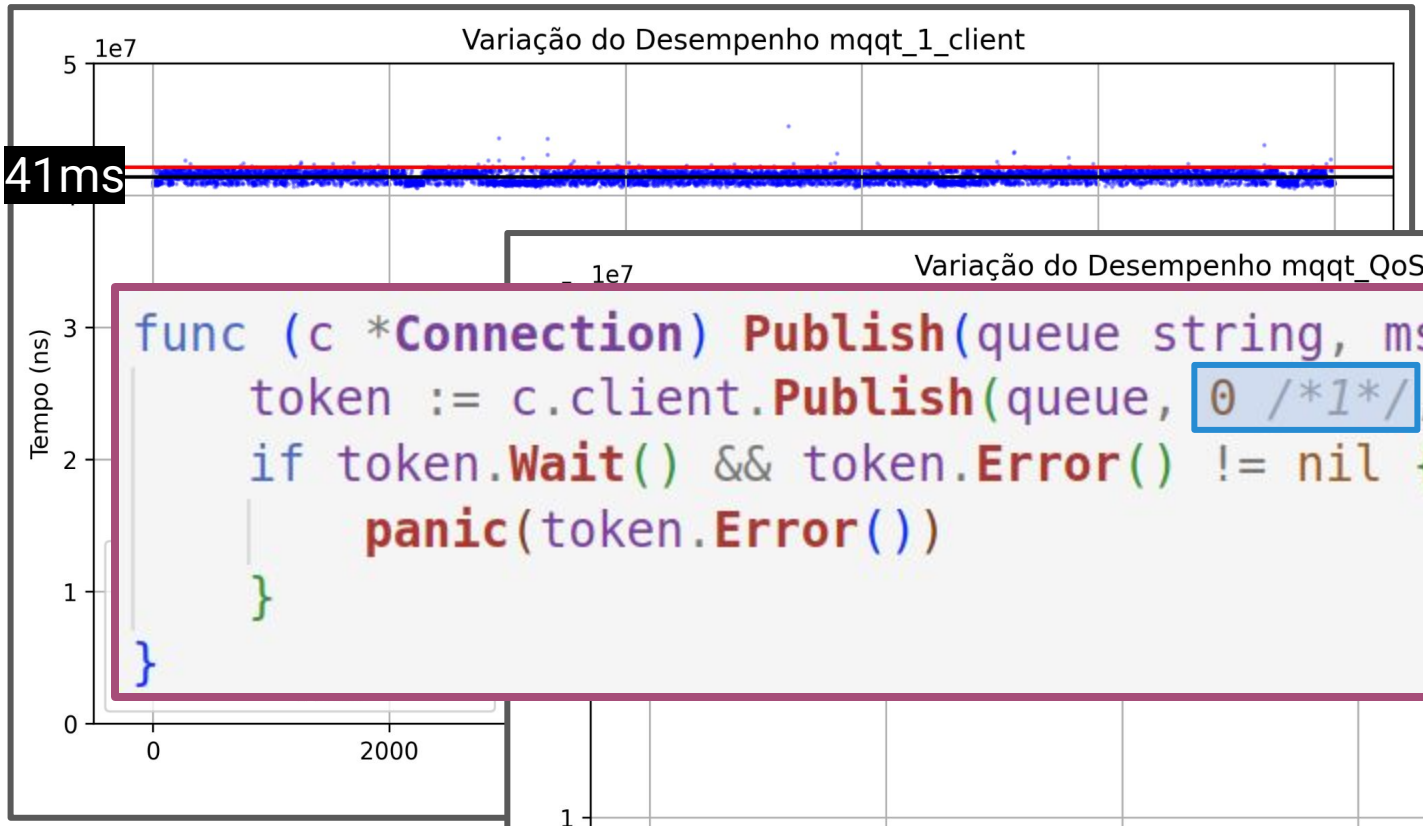
RabbitMQ	rabbitmq:4.0-management	
MQTT	eclipse-mosquitto	persistence true allow_anonymous true

Mqtt QoS=1 - Rabbitmq

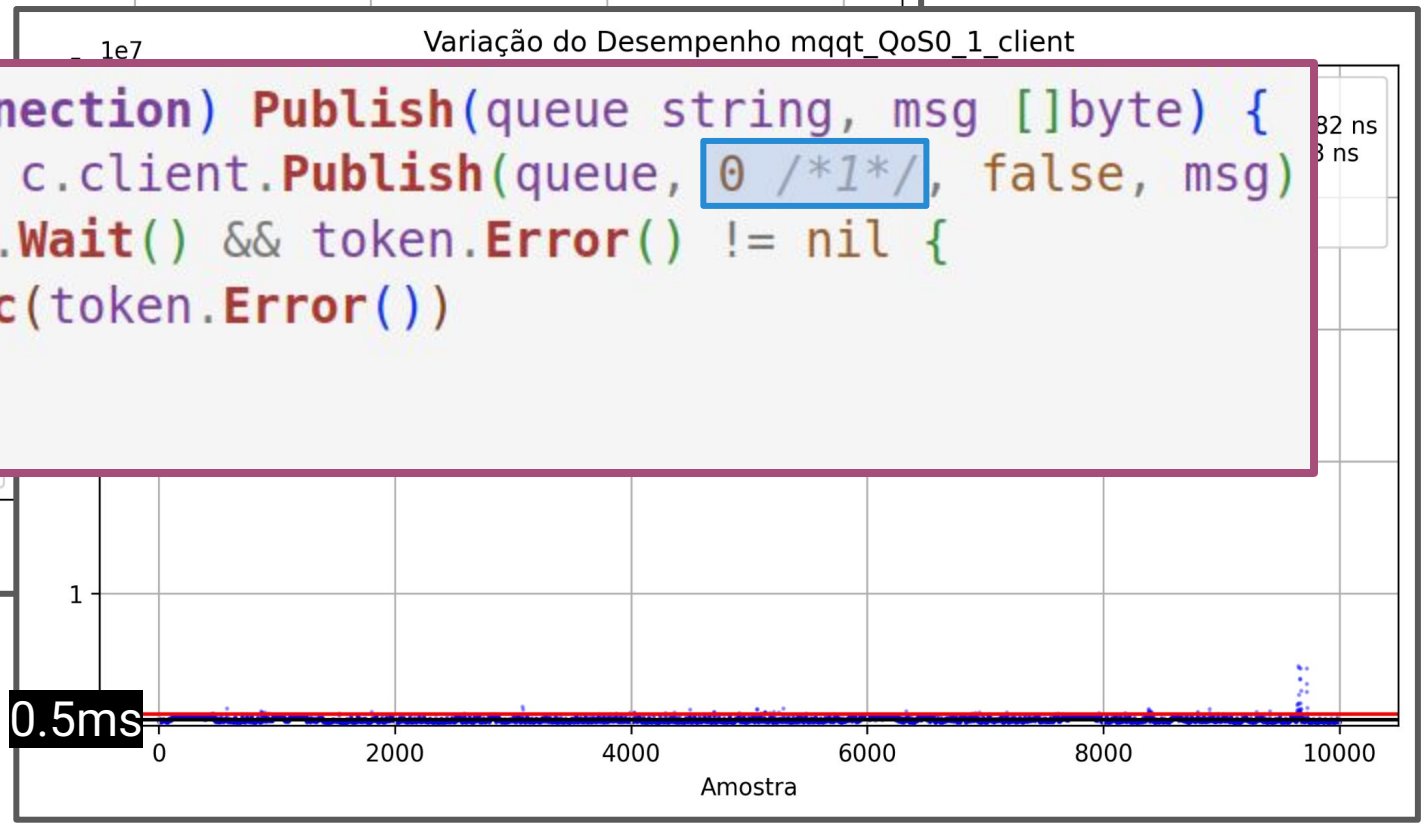




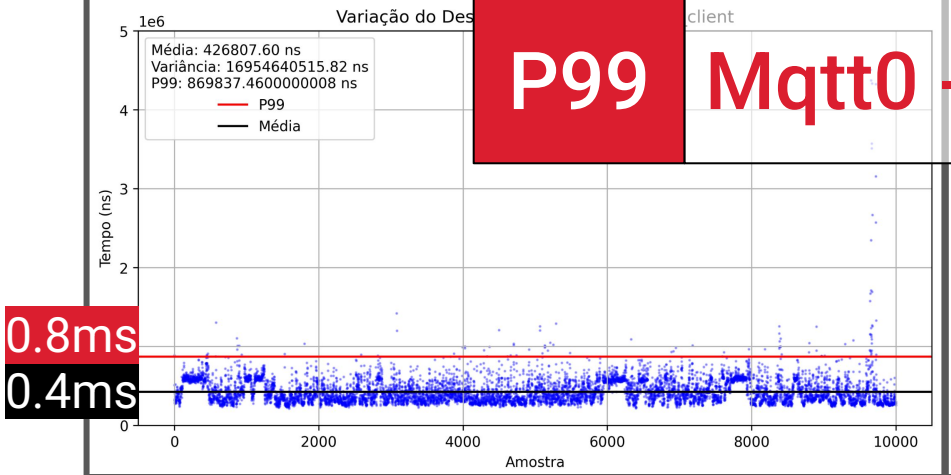
Mqtt QoS=0 - Mqtt QoS=1

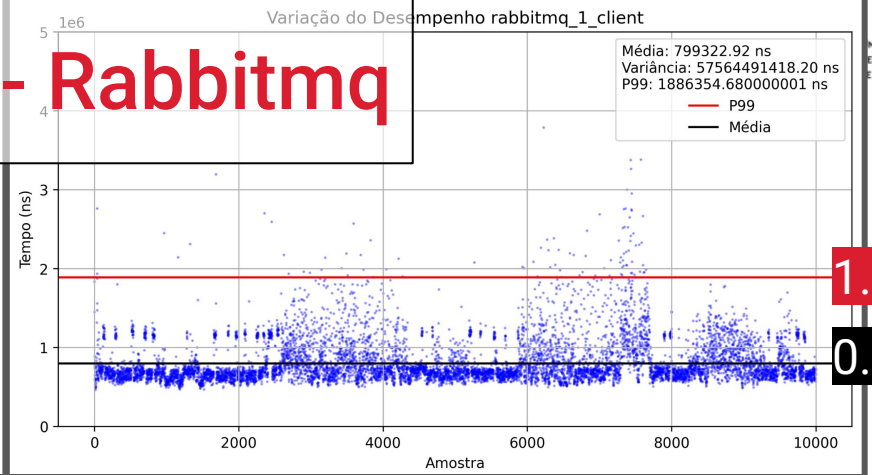


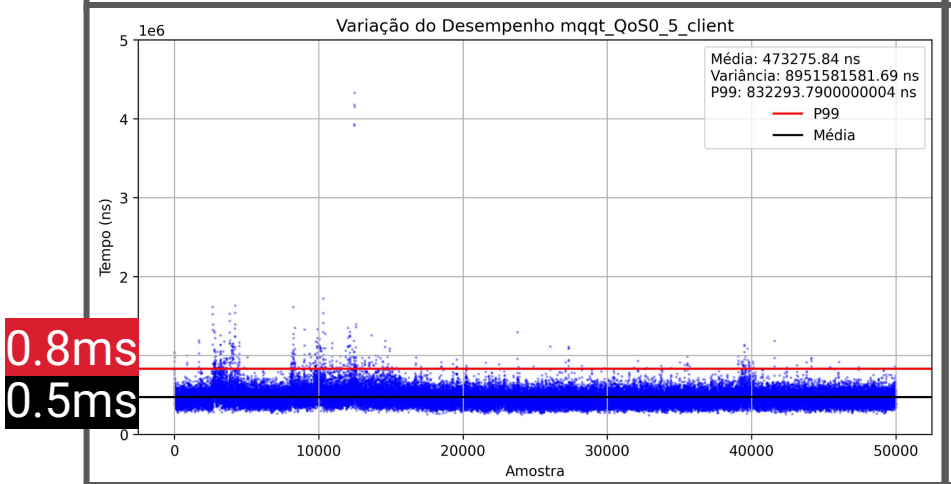

```go
func (c *Connection) Publish(queue string, msg []byte) {
    token := c.client.Publish(queue, 0 /*1*/, false, msg)
    if token.Wait() && token.Error() != nil {
        panic(token.Error())
    }
}
```

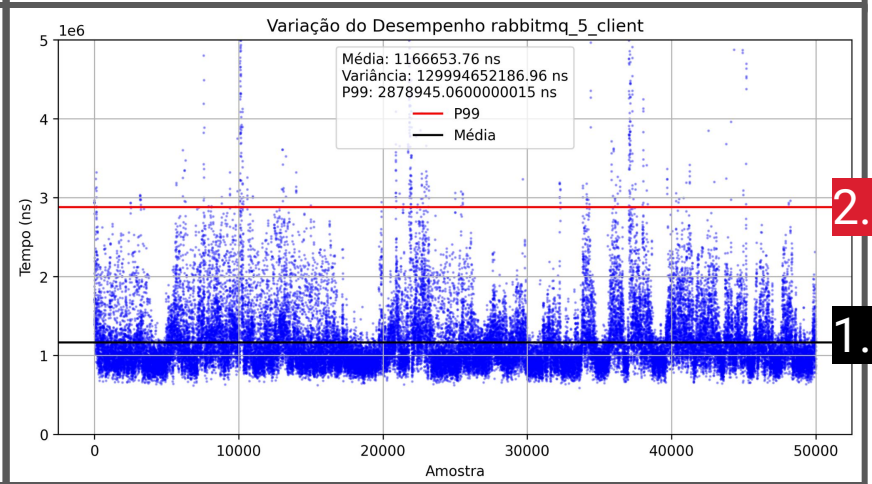


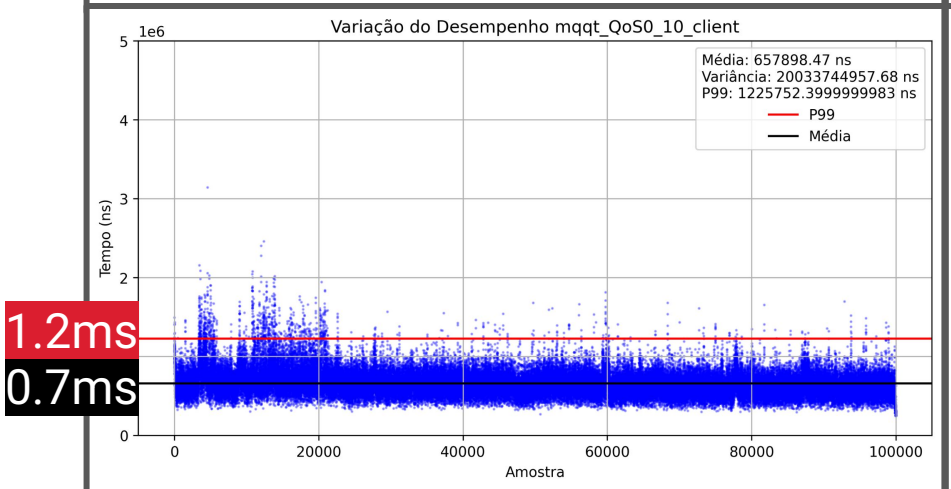
P99 Mqtt0 - Rabbitmq

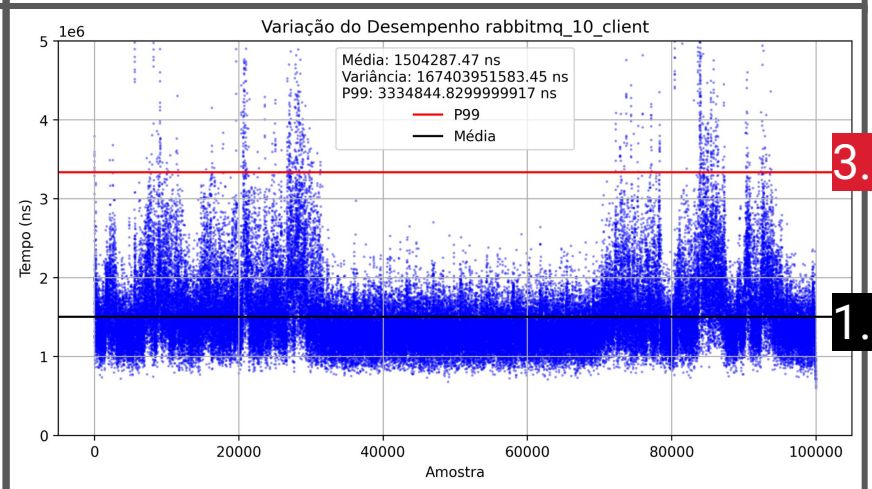




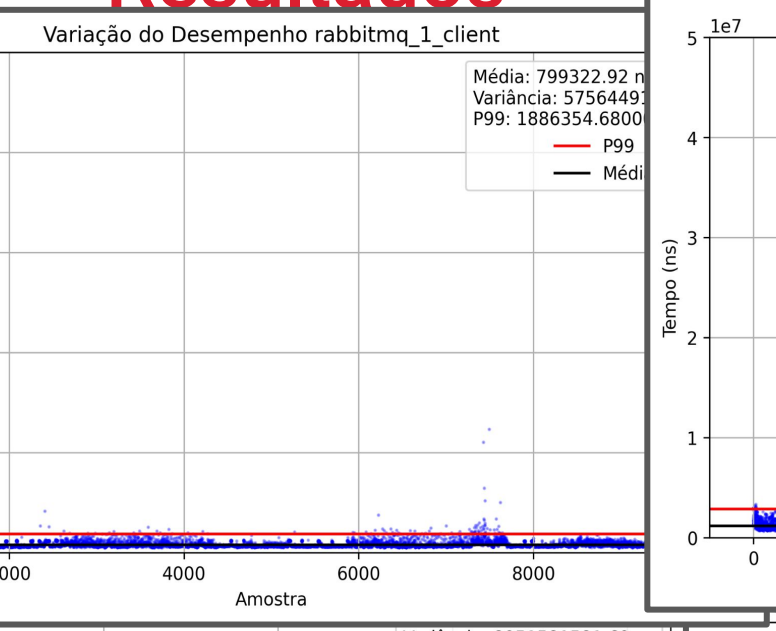


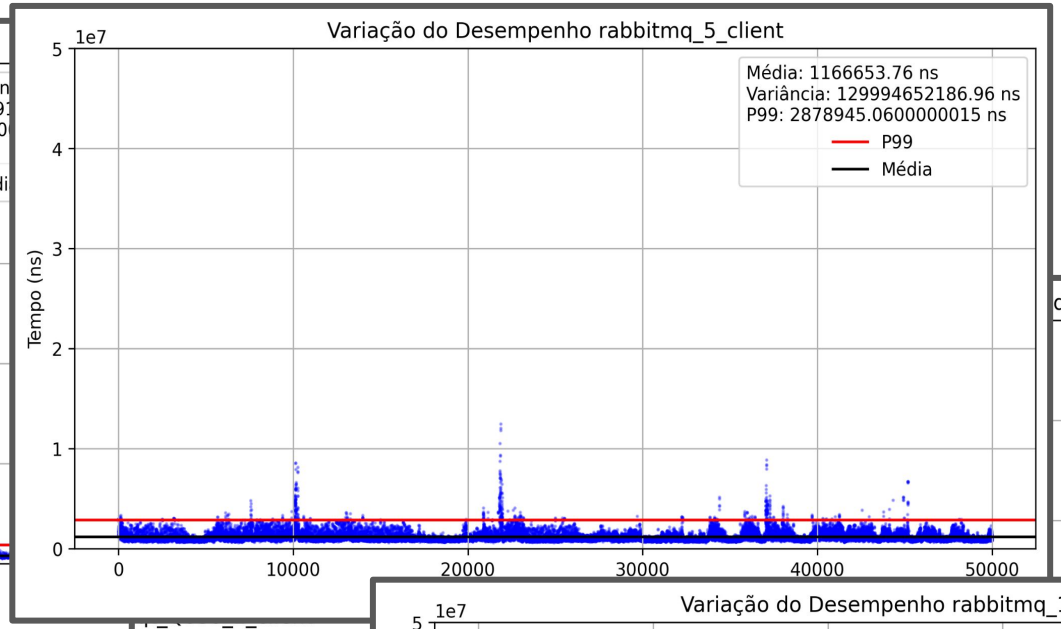


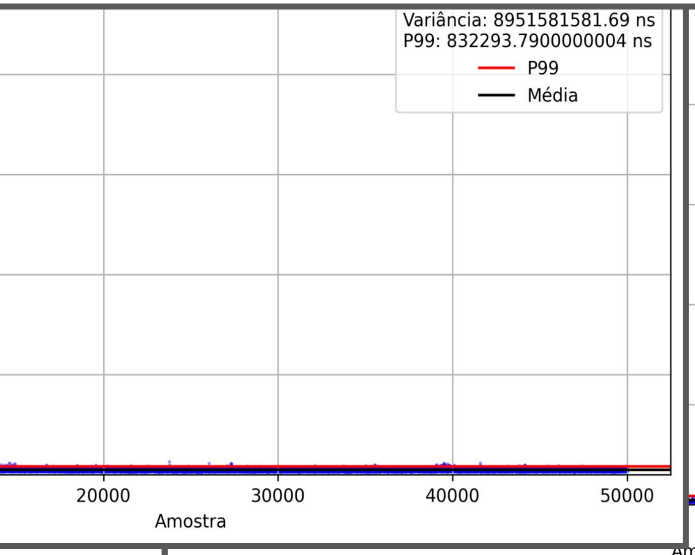


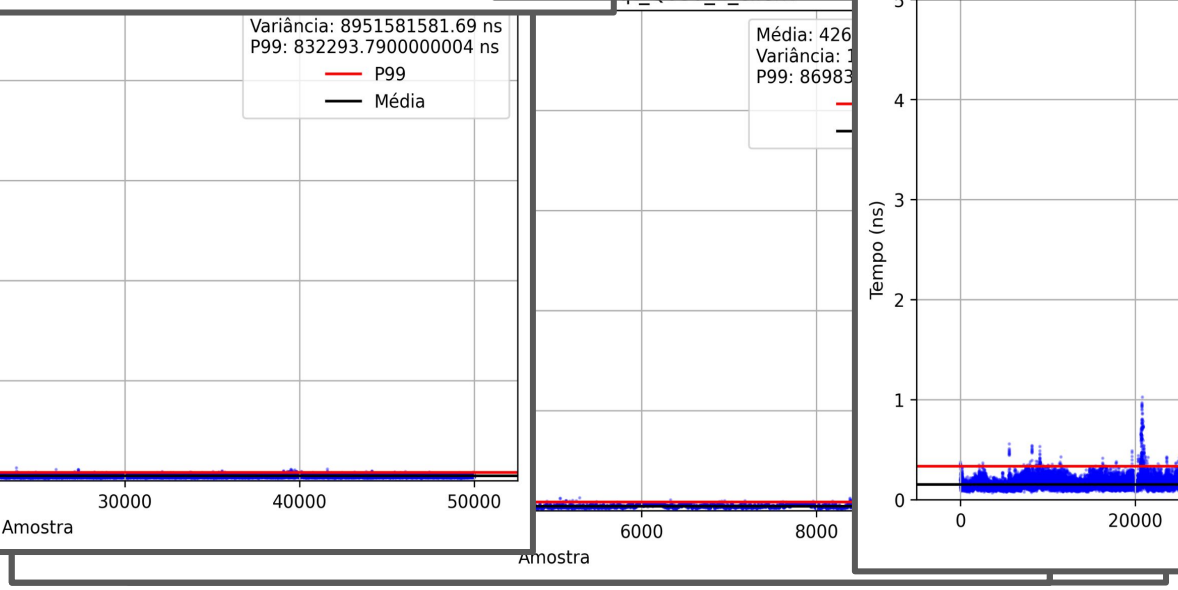


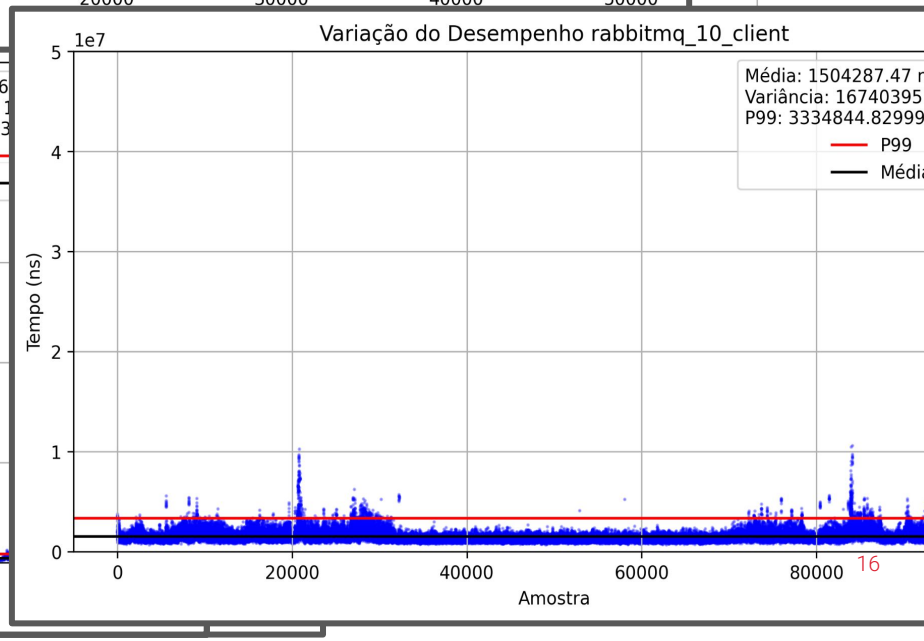
Resultados


Variação do Desempenho rabbitmq_1_client
Média: 799322.92 n
Variância: 5756449
P99: 1886354.6800
P99
Médi
000
4000
6000
8000
Amostra

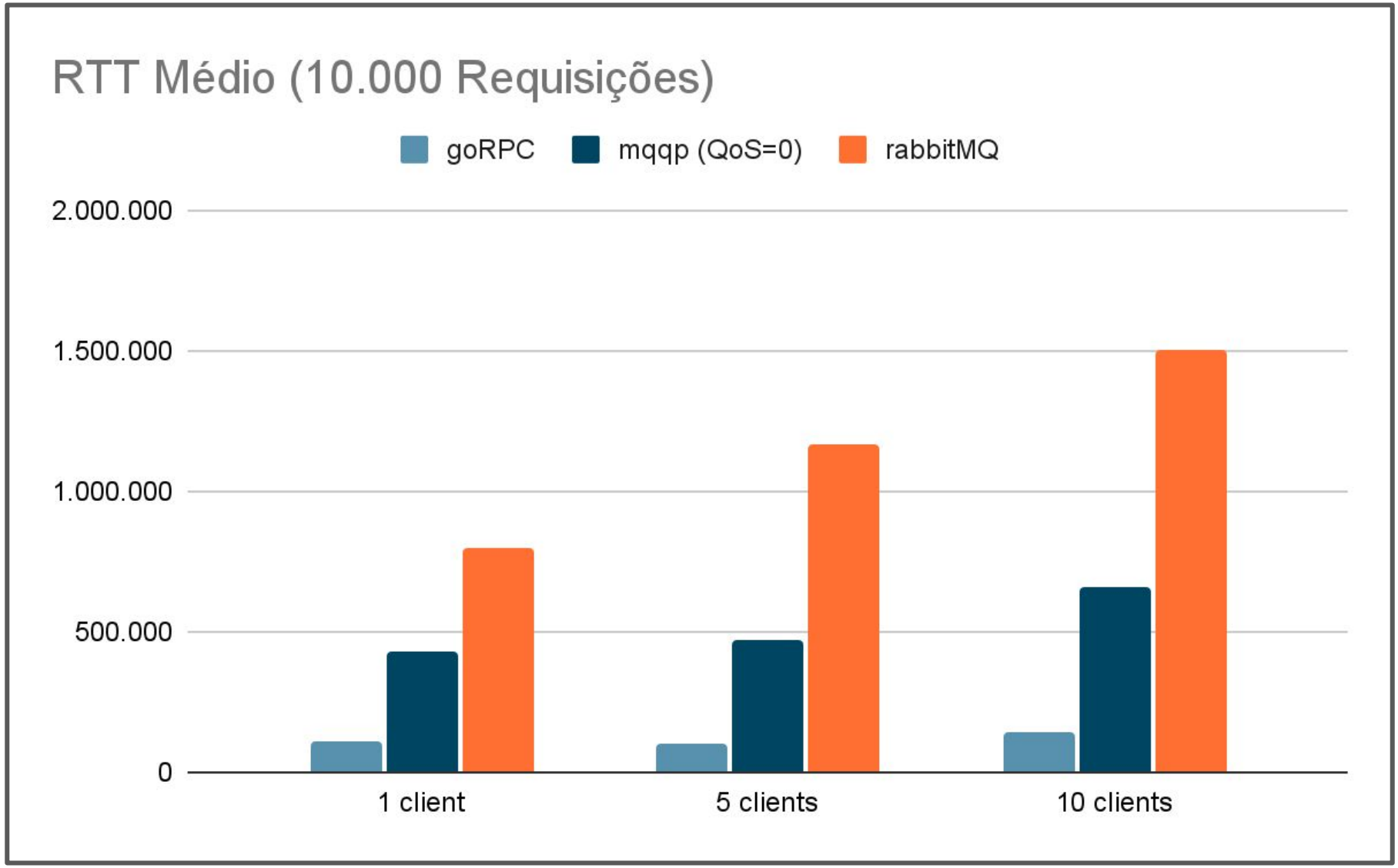

Variação do Desempenho rabbitmq_5_client
1e7
Média: 1166653.76 ns
Variância: 129994652186.96 ns
P99: 2878945.0600000015 ns
P99
Média
Tempo (ns)
0
1
2
3
4
5
0
10000
20000
30000
40000
50000


Variância: 8951581581.69 ns
P99: 832293.7900000004 ns
P99
Média
20000
30000
40000
50000
Amostra

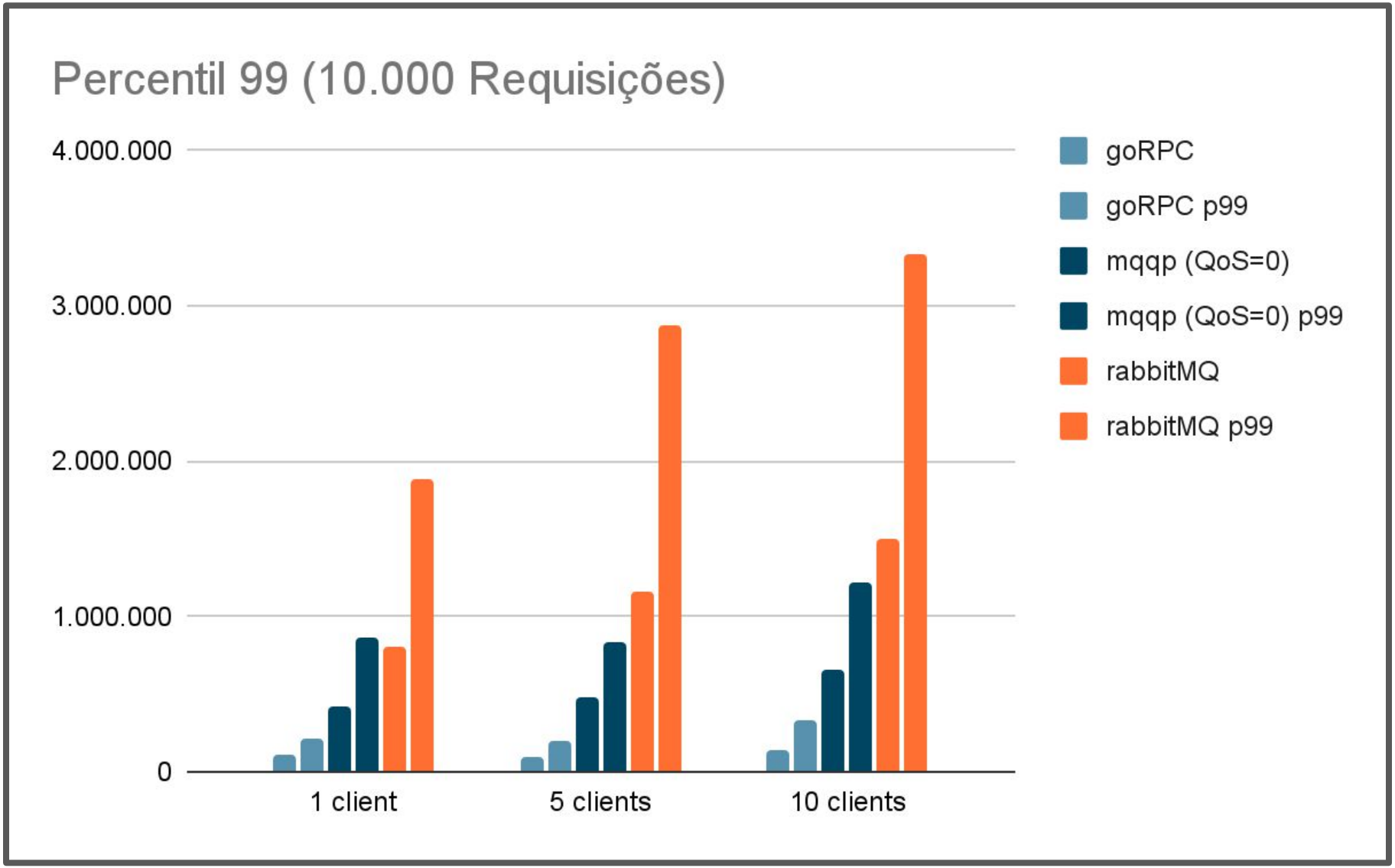

Média: 426
Variância: 1
P99: 86983
6000
8000
Amostra


Variação do Desempenho rabbitmq_10_client
1e7
Média: 1504287.47 n
Variância: 16740395
P99: 3334844.82999
P99
Médi
Tempo (ns)
0
1
2
3
4
5
0
20000
40000
60000
80000
Amostra

RTT Médio


RTT Médio (10.000 Requisições)
goRPC
mqqp (QoS=0)
rabbitMQ
2.000.000
1.500.000
1.000.000
500.000
0
1 client
5 clients
10 clients

RTT Médio


Percentil 99 (10.000 Requisições)
4.000.000
3.000.000
2.000.000
1.000.000
0
1 client
5 clients
10 clients
goRPC
goRPC p99
mqqp (QoS=0)
mqqp (QoS=0) p99
rabbitMQ
rabbitMQ p99

Referências

- **O que é o 1% Low em gráficos de performance?**
  - https://www.adrenaline.com.br/hardware/o-que-e-o-1-low-em-graficos-de-performance/
- **Anton Putra - Performance Benchmark in Kubernetes**
  - https://www.youtube.com/watch?v=PL0c-SvjSVg
- **Códigos Python para Plotagem**
  - https://github.com/erbert-gadelha/matplot-percentil
- **Códigos apresentados**
  - https://github.com/erbert-gadelha/go-files