TESTE DE PERFORMANCE 9

Engenharia da Computação
Projeto em Arquitetura de Computadores, Sistemas Operacionais e Redes

Kaio Henrique Silva da Cunha
Prof.: Alcione Dolavale

Fortaleza, CE
24/11/2021

Teste de Performance 9

A proposta deste projeto foi criar um software cliente-servidor em Python, juntando elementos dos testes de performance anteriores para exibir para o usuário informações de arquitetura de redes, arquitetura de computadores e/ou de sistemas operacionais. Para sintetizar todo esse processo escolhi apenas 6 funções, criadas entre o TP 4 e 7, para integrar o software cliente-servidor. Essas funções são:

- Do TP4, escolhi uma função que exibe informações dos diretórios, como nome, tamanho, localização, data de criação, data de modificação, tipo, etc. Também do TP4, uma função de captura das informações dos processos do sistema, como PID, nome do executável, consumo de processamento, consumo de memória.
- Do TP5, apenas a função de escalonamento de chamadas, utilizando módulo sched.
- Do TP6, uma função que exibe os hosts ativos numa subnet específica, e outra função que mostra as portas abertas nessa subnet. No entanto, após diversas tentativas - e mesmo com suporte dos colegas -, não tive sucesso em exibir as informações dessas funções de forma satisfatória no cliente. Decidi incluí-las, apesar disso, pensando em realizar esse ajuste posteriormente. Para isso, conto também com alguma sugestão ou orientação do professor.
- Do TP7, extraí apenas uma função que retorna informações de interfaces de rede como IPv4, IPv6, máscara e MAC Address.

Optei também por fazer o TP8 juntamente com o TP9, integrando todas as funções escolhidas no software cliente-servidor, no lugar de uma ou duas.

Os módulos utilizados no servidor e no cliente foram:

- **psutil**: esse módulo é utilizado para recuperar informações sobre processos em execução e utilização do sistema (CPU, memória, discos, rede, sensores) em Python. Ele me foi particularmente útil nas funções pid(), onde precisei iterar sobre todos os processos da máquina, e na função network(), onde utilizei o psutil para buscar informações de interface de rede.
- **time**: esse módulo retorna o número de segundos passados desde uma epoch específica. Utilizei esse módulo com frequência nos inícios e nos fins das funções para saber quanto tempo elas demoraram para fazer seu trabalho.
- **os**: o módulo os em Python fornece funções para interagir com o sistema operacional. Com ele foi possível listar diretórios, obter informações de arquivos, e etc. O seu uso mais interessante foi para executar comandos na máquina, como nmap e clear, para capturar informações sobre hosts e limpar o terminal depois de um processamento, respectivamente.

- **pickle:** o módulo pickle pode transformar um objeto complexo em um fluxo de bytes e pode transformar o fluxo de bytes em um objeto com a mesma estrutura interna. Eu utilizei esse módulo no projeto para manusear as informações trocadas entre o servidor e o cliente.

- **sched:** o sched serviu para executar tarefas em tempos e ordens específicas. A classe do sched usa uma função de time para aprender a hora atual e uma função de atraso para esperar por um período de tempo específico. As unidades reais de tempo não são importantes, o que torna a interface flexível o suficiente para ser usada para muitos propósitos. Utilizei esse módulo na função sched() para escalonar duas outras funções do mesmo projeto, e o cliente apenas exibe o tempo gasto para executar ambas as funções.

- **socket:** o módulo socket da Biblioteca Padrão Python fornece o equivalente à interface de socket BSD, e fornece vários objetos, constantes, funções e exceções relacionadas para a construção de aplicativos de redes, incluindo programas de cliente e servidor. Foi o módulo mais importante desse projeto final, pois permitiu que eu capturasse, entre outras coisas, informações de host e portas a partir de suas funções.

A organização do software se deu da seguinte forma:

- Importei os módulos necessários para o desenvolvimento da aplicação, e utilizei o módulo socket para buscar o host name, além de estabelecer informações de protocolos, como UDP e TCP. Feito isso, foi possível realizar o bind, o listen, e o accept. Esse cenário representa a sequência de chamadas de função para o cliente e um servidor que participa de uma conexão TCP.
- Organizei as funções na mesma ordem dos seus respectivos TPs, com algumas diferenças de implementação, e também retornando seus resultados para que fosse possível enviá-los ao cliente.
- Criei laços para troca de informações no servidor e no cliente. No cliente é exibido um menu com as opções para o usuário. Ao digitar um número representando uma dessas opções, o client envia essa opção para o server através da função send. Feito isso, o serve recebe e decodifica a opção, e baseado nela, chama a função apropriada, obtém e serializa seu retorno utilizando a função dump do módulo pickle. O cliente recebe essa informação e a encaminha para a sua função equivalente de processamento e formatação. Segue um trecho deste código:

```
143  while True:
144      option = client.recv(buffer)
145      if option.decode('utf-8') == " ":
146          print("Connected.")
147      elif option.decode('utf-8') == "1":
148          bytes_resp = pickle.dumps(get_file())
149          client.send(bytes_resp)
150      elif option.decode('utf-8') == "2":
151          bytes_resp = pickle.dumps(pid())
152          client.send(bytes_resp)
153      elif option.decode('utf-8') == "3":
154          bytes_resp = pickle.dumps(scheduler())
155          client.send(bytes_resp)
156      elif option.decode('utf-8') == "4":
157          bytes_resp = pickle.dumps(up_hosts())
158          client.send(bytes_resp)
159      elif option.decode('utf-8') == "5":
160          bytes_resp = pickle.dumps(port_nmap())
161          client.send(bytes_resp)
162      elif option.decode('utf-8') == "6":
163          bytes_resp = pickle.dumps(network())
164          client.send(bytes_resp)
165
```

```
try:
    client.connect(HP)
    option = " "
    client.send(option.encode('utf-8'))
    options = ("1","2", "3", "4", "5", "6")

    count = 1
    while count == 1:
        menu()
        option = input('Type in your option: ')
        if (option in options):
            client.send(option.encode('utf-8'))
            bytes_resp = client.recv(buffer)
            response = pickle.loads(bytes_resp)
            if option == "1":
                one()
            elif option == "2":
                two()
            elif option == "3":
                three()
            elif option == "4":
                four()
            elif option == "5":
                five()
            elif option == "6":
                six()
            elif option == '7':
                count == 2
            else:
                print('Invalid option. Try again...')
except Exception as erro:
    print(str(erro))
```

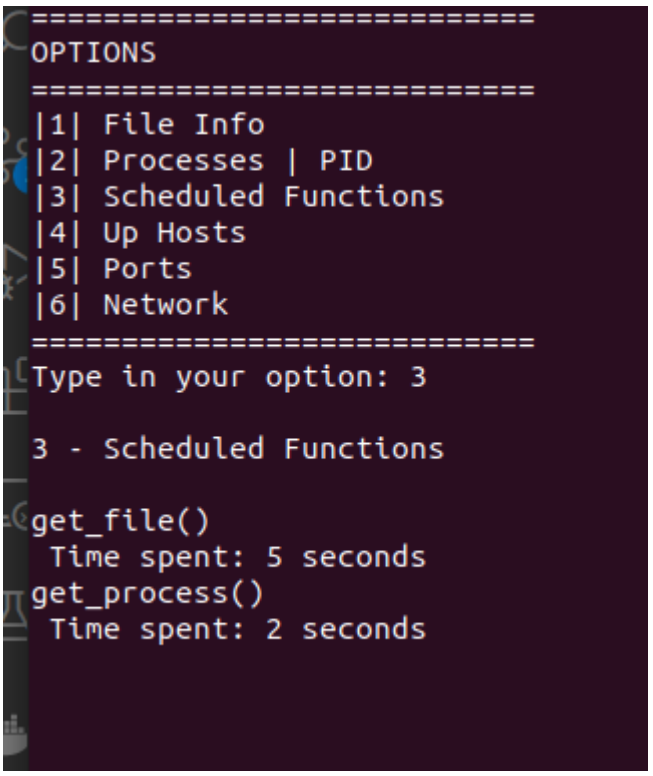
O menu do cliente exibe as seguintes opções:

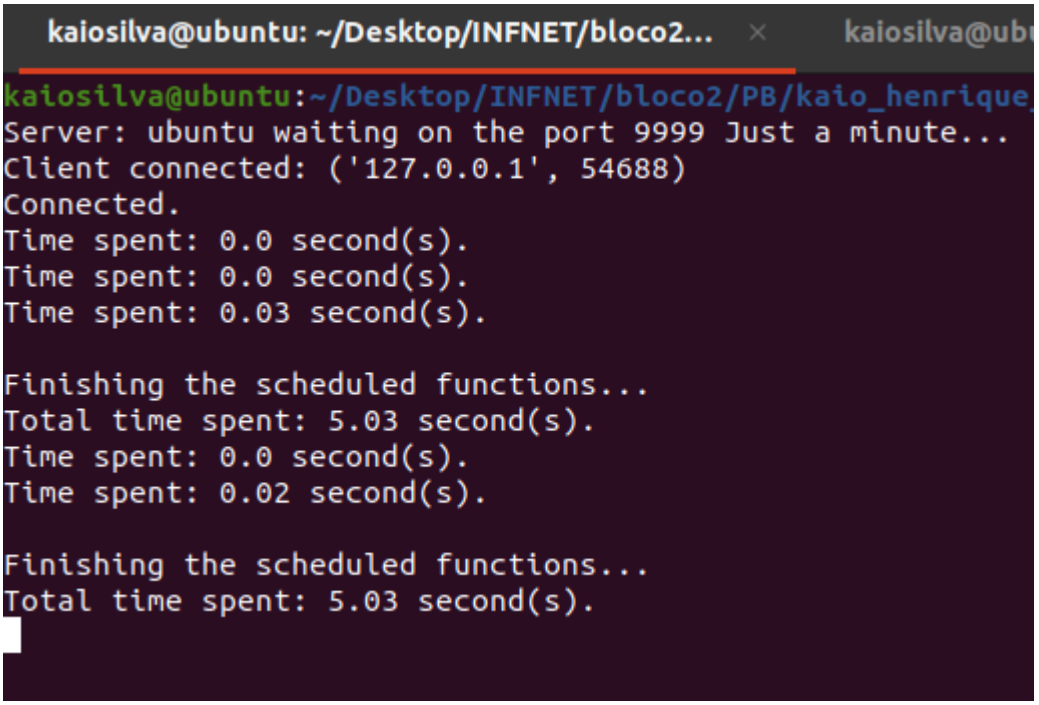
```
124  def menu():
125      print('============================')
126      print('OPTIONS')
127      print('============================')
128      print('|1| File Info')
129      print('|2| Processes | PID')
130      print('|3| Scheduled Functions')
131      print('|4| Up Hosts')
132      print('|5| Ports')
133      print('|6| Network')
134      print('============================')
```

Ao digitar uma dessas opções, o programa executa a função equivalente, por exemplo:

```
43  def three():
44      list = []
45
46      print("\n3 - Scheduled Functions\n")
47      d1 = response[1]
48      d2 = response[0]
49
50      print('get_file()')
51      print(d1)
52      print('get_process()')
53      print(d2)
54
55      time.sleep(8)
56      os.system('clear')
57      list.append(d1)
58      list.append(d2)
59      return list
```

No terminal, é possível visualizar o resultado dessas funções:


============================
OPTIONS
============================
|1| File Info
|2| Processes | PID
|3| Scheduled Functions
|4| Up Hosts
|5| Ports
|6| Network
============================
Type in your option: 3
3 - Scheduled Functions
get_file()
Time spent: 5 seconds
get_process()
Time spent: 2 seconds

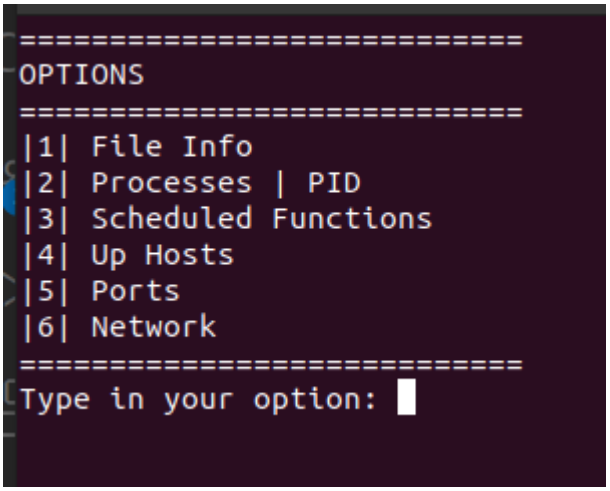

kaiosilva@ubuntu: ~/Desktop/INFNET/bloco2...
kaiosilva@ubuntu:~/Desktop/INFNET/bloco2/PB/kaio_henrique_
Server: ubuntu waiting on the port 9999 Just a minute...
Client connected: ('127.0.0.1', 54688)
Connected.
Time spent: 0.0 second(s).
Time spent: 0.0 second(s).
Time spent: 0.03 second(s).
Finishing the scheduled functions...
Total time spent: 5.03 second(s).
Time spent: 0.0 second(s).
Time spent: 0.02 second(s).
Finishing the scheduled functions...
Total time spent: 5.03 second(s).

```
============================
OPTIONS
============================
|1| File Info
|2| Processes | PID
|3| Scheduled Functions
|4| Up Hosts
|5| Ports
|6| Network
============================
Type in your option: 2

2- Processes | PID

Name:                           python3
PID:                            4789
Executable:                     /usr/bin/python3.8
CPU(s):                         0.59
Memory(MB):                     44.69

Name:                           python3
PID:                            4790
Executable:                     /usr/bin/python3.8
CPU(s):                         0.54
Memory(MB):                     44.75

Name:                           python3
PID:                            4791
Executable:                     /usr/bin/python3.8
CPU(s):                         11.61
Memory(MB):                     66.68
```

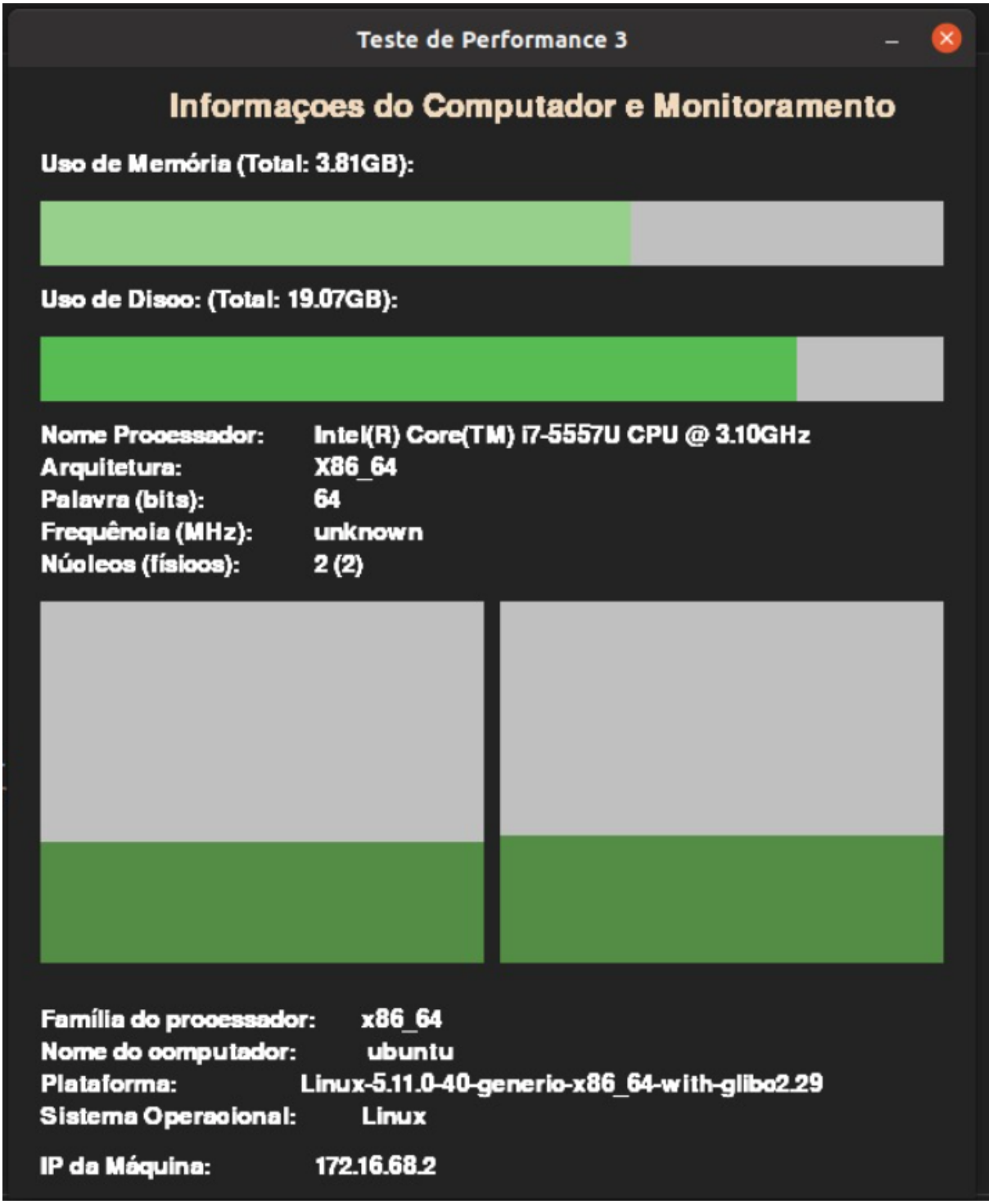
```
OPTIONS
============================
|1| File Info
|2| Processes | PID
|3| Scheduled Functions
|4| Up Hosts
|5| Ports
|6| Network
============================
Type in your option: 1
\Files

kaio_henrique_PB_TP8.ipynb
Size (bytes): 25
Path: /home/kaiosilva/Desktop/INFNET/bloco2/PB/kaio_henrique_PB_TP9/kaio_henrique_PB_TP8.ipynb
Criação: Thu Nov 25 00:24:54 2021
Last modification: Thu Nov 25 00:24:54 2021
Last access: Wed Nov 24 23:28:50 2021

client.py
Size (bytes): 4
Path: /home/kaiosilva/Desktop/INFNET/bloco2/PB/kaio_henrique_PB_TP9/client.py
Criação: Wed Nov 24 23:56:13 2021
Last modification: Wed Nov 24 23:56:13 2021
Last access: Wed Nov 24 23:56:18 2021

server.py
Size (bytes): 5
Path: /home/kaiosilva/Desktop/INFNET/bloco2/PB/kaio_henrique_PB_TP9/server.py
Criação: Wed Nov 24 23:35:46 2021
Last modification: Wed Nov 24 23:35:46 2021
Last access: Wed Nov 24 23:35:51 2021
```

```
============================
OPTIONS
============================
|1| File Info
|2| Processes | PID
|3| Scheduled Functions
|4| Up Hosts
|5| Ports
|6| Network
============================
Type in your option: 6

6- Network

IPv4:                           ::1
IPv6:                           00:00:00:00:00:00
Mask:                           ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
MAC:                            127.0.0.1
```

Outra dificuldade encontrada, além daquelas com a exibição dos resultados do nmap, foi fazer com que a opção 2 retornasse, depois de executada, para esse ponto:



Todo o projeto foi feito numa máquina virtual Linux Ubuntu hospedada num host macOS Monterey. Essa escolha dificultou o meu avanço em alguns momentos, mas foi necessária porque tive um problema com incompatibilidade de versões do Python e do pip na minha máquina host que me impossibilitaram de instalar alguns módulos. Esse problema ainda não foi resolvido.



Para desenvolver e organizar a aplicação utilizei o VS Code e o versionamento de código com git. O TP8 e o TP9 estão contidos nos arquivos server.py e client.py. Todos os outros TPs estão em suas respectivas pastas, assim como num único jupyter notebook chamado kaio_henrique_PB_TP8.ipynb.

O desenvolvimento gradual dos TPs, culminando nesse projeto final, foi como ir juntando as peças de um quebra cabeça sem usar a imagem do resultado final como referência. Digo

que não havia imagem final como referência pois, por mais que tenhamos acesso a todos os TPs desde o início do curso, não temos o entendimento nem a capacidade técnica de saber exatamente o que eles propõem desde o início. Esse conhecimento vai sendo esculpido ao longo da disciplina até que chegamos ao inesperado resultado final.

Referências

Material da disciplina no Moodle e gravações das aulas. Acesso em nov. 2021.