

Analise Socket Portas

Kalil Saldanha Kaliffe¹

¹Instituto de Ciências Exatas e Naturais - Faculdade de Computação
Universidade Federal do Pará (UFPA) - Belém, PA - Brasil

{kalil.kaliffe}@icen.ufpa.br

1. Relatório

Foi desenvolvido um código em C / Python [Kalil 2023] para criar e analisar as portas dos sockets, o código em Python serve escrever no disco e compilar o código em C, o qual cria vários sockets e salva a porta atribuída pelo Sistema Operacional em um arquivo CSV, o qual depois é lido pelo restante do script Python para análise.

Portanto, foi desenvolvido duas versões do programa de criação de socket o primeiro cria sockets em sequência sem fechar os sockets anteriores para cada rodada de execução, enquanto o segundo fecha o socket anterior antes de criar um socket novo para cada rodada de execução, sendo que no final de cada rodada de execução os sockets são fechados em ambos,

Dessa forma, as Figuras 1 a 2 as execuções de 200 aberturas para cada computador de cada versão do código, pois uma rodada de execução do código cria 10 em ambas as versões do programa, e foram executadas 10 rodadas para cada.

Isso foi feito duas vezes, em dois computadores diferentes, sendo primeiro gráfico do Computador Ubuntu oferecido pelo Google Colab Figura 3, e o segundo de um computador local rodando Manjaro Figura 4 ,

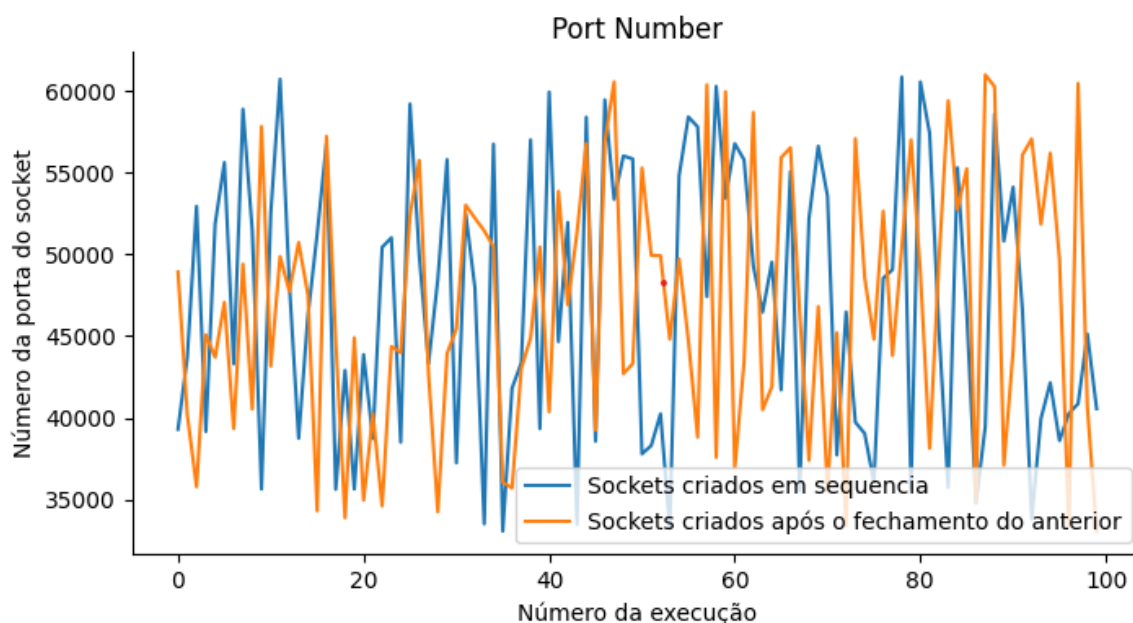
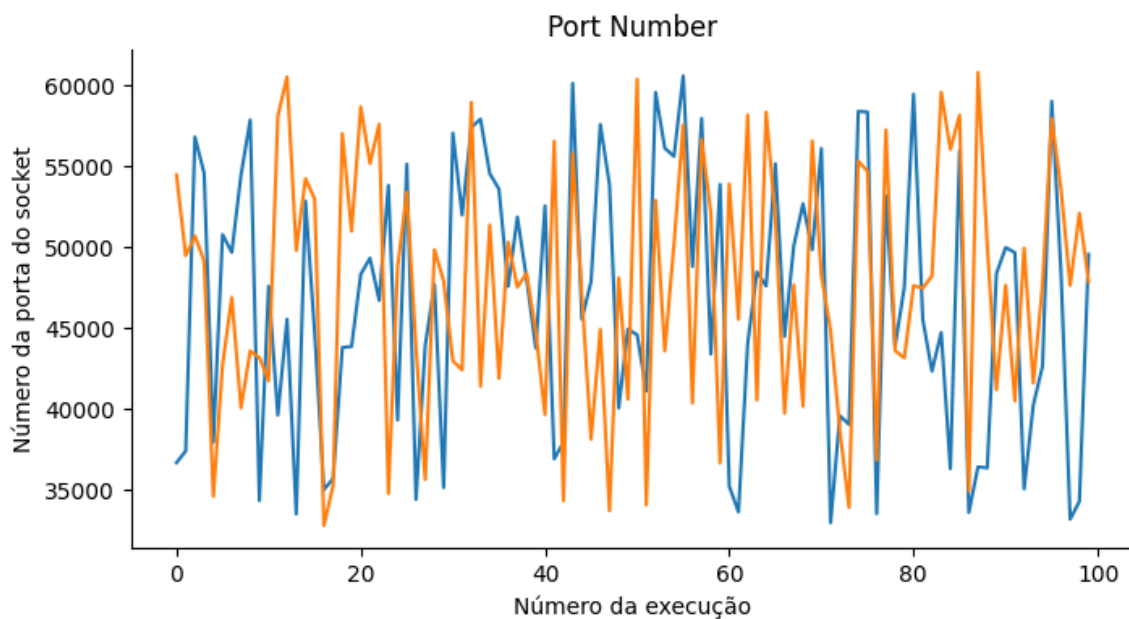


Figura 1. Sequencia das portas atribuídas no pc do Colab 3



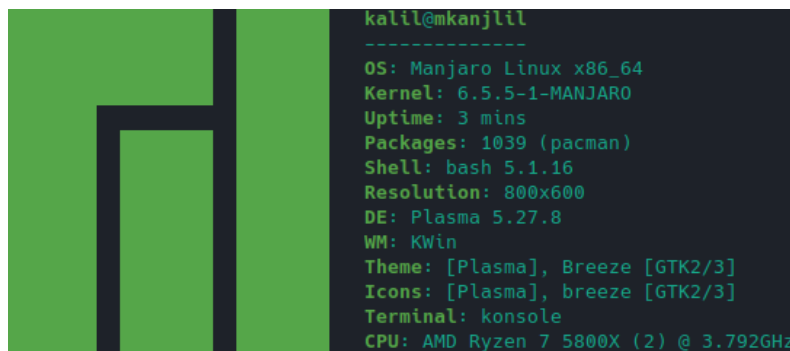
```

.-/+00555500+/-.-
`:+55555555555555555555+:`
-+55555555555555555555yy5555+-
.05555555555555555555dMMMMNy55550.
/555555555555hdmnnNmmYMMMMMh555555/
+5555555555hmydMMMMMMMMddddd55555555+
/5555555555hNMMMyhhyyyYhNMMMMMh55555555/
.5555555555dMMMMNhs555555555555hNMMMMy55555555.
+5555hhhyNMMMy55555555555555yNMMMy55555555+
ossyNMMMNyMMh5555555555555555hmmmh55555550
ossyNMMMNyMMh555555555555555555hmmmh55555550
+5555hhhyNMMNy555555555555555555hNMMMMy55555555+
.55555555dMMMMNhs555555555555hNMMMd55555555.
/5555555555hNMMMyhhyyyYhNMMMMMh55555555/
+5555555555dmydMMMMMMMMddddd55555555+-
/555555555555hdmNNNNmyNMMMNy55555555/
.05555555555555555555dMMMMNy55550.
-+55555555555555555555yy5555+-
`:+55555555555555555555+:`
.-/+00555500+/-.-

root@7962ff1837d6
-----
OS: Ubuntu 22.04.2 LTS x86_64
Host: Google Compute Engine
Kernel: 5.15.120+
Uptime: 5 hours, 45 mins
Packages: 1296 (dpkg)
Shell: bash 5.1.16
Terminal: jupyter-notebook
CPU: Intel Xeon (2) @ 2.199GHz
Memory: 936MiB / 12982MiB

```

Os valores das portas obtidos variaram de execução para execução, mas seguiram uma tendência crescente. Isso indica que o sistema operacional pode estar tentando atribuir as portas de forma sequencial inicialmente, no entanto, pode haver lacunas ou saltos devido a outras aplicações que usam as portas e o algoritmo aleatório com base no tempo atual do sistema na execução. Além disso, fechar explicitamente cada socket antes de criar o próximo não afeta os valores das portas atribuídas pelo sistema operacional, já que

A terminal window with a dark background and green text. On the left, there is a green square logo with a black stylized 'H' shape. The terminal text shows the user 'kalil@mkanjll' and various system details.

```
kalil@mkanjll
-----
OS: Manjaro Linux x86_64
Kernel: 6.5.5-1-MANJARO
Uptime: 3 mins
Packages: 1039 (pacman)
Shell: bash 5.1.16
Resolution: 800x600
DE: Plasma 5.27.8
WM: KWin
Theme: [Plasma], Breeze [GTK2/3]
Icons: [Plasma], breeze [GTK2/3]
Terminal: konsole
CPU: AMD Ryzen 7 5800X (2) @ 3.792GHz
```

Figura 4. Pc local

o fator aleatório se mantêm, assim como a tendência crescente, entre as execuções que fecham ou não os sockets, pois o sistema operacional libera as portas automaticamente após o término de um round de 10 execuções do programa.

Dessa forma, não há uma relação direta entre os valores das portas obtidas, mas sim uma dependência do estado das portas no momento da execução do programa, a qual não é previsível. Entretanto, o range desses números disponíveis é definido pelo OS, isso pode ser observado nas métricas das Figuras 5 e 6 a qual aparenta estar entre 32000 e 61000

Assim, fechar explicitamente cada socket antes de criar o próximo não afeta os valores das portas atribuídas pelo sistema operacional em geral, pois a média, mediana e moda não tem muita variação, sendo de aproximadamente 1000 números de porta.

Entretanto, como a tendência crescente reinicia a cada vez que um round de 10 execuções acaba, pois são liberadas as portas, na Figura 2 a tendência crescente não é tão evidente para os rounds de 10 como como no cenário da Figura 1 sem liberar as portas constantemente (só no final do round de 10). Justamente porque, os números aleatórios anteriores foram liberados não limitando novas portas a portas crescentes.

No código em C [Kalil 2023], a função `socket.bind()` é usada para associar o socket a um endereço e porta específicos. O endereço localhost é usado para indicar que o socket deve ser associado à interface de loopback do sistema. O número da porta é definido como 0, o que indica que o sistema operacional deve escolher uma porta disponível.

Assim, em loop o socket principal cria sockets filho e os filhos podem ser fechados quando o principal é fechado ou fechar eles após sua criação individualmente.

Inicialmente, o programa se conectava com um cliente, no entanto, isso não se mostrou necessário para essa análise, assim como o uso correto do bind para atribuir um socket específico e não deixar o OS escolher, já que o ponto desse experimento é analisar essa atribuição automática.

Referências

Kalil (2023). Código Fonte.

```
Minimum Port: 33063
Maximum Port: 60865
Average Port: 47084.7
Median Port: 47033.0
Mode Port: 39299
Variance Port: 69671943.62626262
Standard Deviation Port: 8346.972123247006
Range Port: 27802

Minimum Port: 33049
Maximum Port: 60995
Average Port: 47084.7
Median Port: 46596.0
Mode Port: 48931
Variance Port: 62141264.63676768
Standard Deviation Port: 7882.973083600355
Range Port: 27946
```

Figura 5. Métricas do CSV do gráfico 1

```
Minimum Port: 32949
Maximum Port: 60587
Average Port: 46735.92
Median Port: 47587.0
Mode Port: 36663
Variance Port: 65820287.87232323
Standard Deviation Port: 8112.970348295576
Range Port: 27638

Minimum Port: 32779
Maximum Port: 60799
Average Port: 46735.92
Median Port: 47882.0
Mode Port: 43575
Variance Port: 58240230.71474747
Standard Deviation Port: 7631.528727243806
Range Port: 28020
```

Figura 6. Métricas do CSV do gráfico 1