

# Programação concorrente em Python

---

# Modelos de concorrência em Python

- Processos: módulos multiprocessing, futures
- Threads: módulos threading, futures
- Corrotinas: módulo asyncio
  - frameworks modernos como FastAPI

Vamos explorar os  
três modelos na  
teoria e na prática

Hoje veremos  
threads e  
processos

# Demonstração: imagens da Wikipédia

localhost:8888/notebooks/wikipics/wikipics-demo.ipynb

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)


```
[11]: %%time
      from concurrent import futures

      num_images = 10
      gallery = Gallery(num_images)
      gallery.display()

      urls = get_sample_urls(5_000_000, num_images)
      with futures.ThreadPoolExecutor() as pool:
          img_records = pool.map(fetch, urls)
          for i, img_rec in enumerate(img_records):
              gallery.update(i, img_rec.pixels)

      print(f'TOTAL BYTES: {gallery.size:_}')
```

https://bit.ly/wikipics2024



|       |                 |  |
|-------|-----------------|--|
| ( 1 ) | 5_086_950 bytes | European Parliament Strasbourg Hemicycle -- Diliff.jpg       |
| ( 2 ) | 5_073_179 bytes | William Cranch.jpg   |
| ( 3 ) | 4_943_441 bytes | Lee Bollinger -- Daniella Zalcmann_less_noise.jpg            |
| ( 4 ) | 5_037_414 bytes | Sidney Hall -- Urania%27s Mirror -- Draco and Ursa Minor.jpg |
| ( 5 ) | 5_007_245 bytes | The Day the Earth Smiled -- PIA17172.jpg                     |
| ( 6 ) | 5_063_007 bytes | Elizabeth I Steven Van Der Meulen.jpg                        |

# Análise dos resultados

## Download sequencial

```
TOTAL BYTES: 19_955_565  
CPU times: user 732 ms, sys: 262 ms, total: 994 ms  
Wall time: 16.4 s
```

## Download com threads

```
TOTAL BYTES: 19_950_522  
CPU times: user 9.27 s, sys: 251 ms, total: 9.52 s  
Wall time: 2.67 s
```

# Entendendo as medidas de CPU time

```
CPU times: user 732 ms, sys: 262 ms, total: 994 ms  
Wall time: 16.4 s
```

- **user**: tempo executando código user-mode (interpretador, seu código, bibliotecas)
- **sys**: tempo executando código kernel-mode (tudo que envolve o hardware: armazenagem, rede, tela, etc.)
- **total**: user + sys
- **wall time**: tempo transcorrido no "relógio da parede", também conhecido como "real time"

# Análise: download sequencial

```
TOTAL BYTES: 19_955_565  
CPU times: user 732 ms, sys: 262 ms, total: 994 ms  
Wall time: 16.4 s
```

- Baixou  $\approx 20$  MB em 16.4 s:  $\approx 1.2$  MB/s
- A CPU trabalhou por 0.994 s (6% do total)
- 94% do tempo o SO executou outros processos enquanto aguardava respostas da rede
- Isso é uma tarefa I/O bound (limitada por E/S)

# Análise: download com threads

```
TOTAL BYTES: 19_950_522  
CPU times: user 9.27 s, sys: 251 ms, total: 9.52 s  
Wall time: 2.67 s
```

- Baixou  $\approx 20$  MB em 2.67 s:  $\approx 7.5$  MB/s
  - 6.1 vezes mais rápido que sequencial (16.4 s)
- A CPU trabalhou por 9.52 s (356% do total)
  - bibliotecas usaram vários núcleos da CPU
- Melhor uso da CPU em tarefa I/O bound

# Aprendizados da demonstração

---

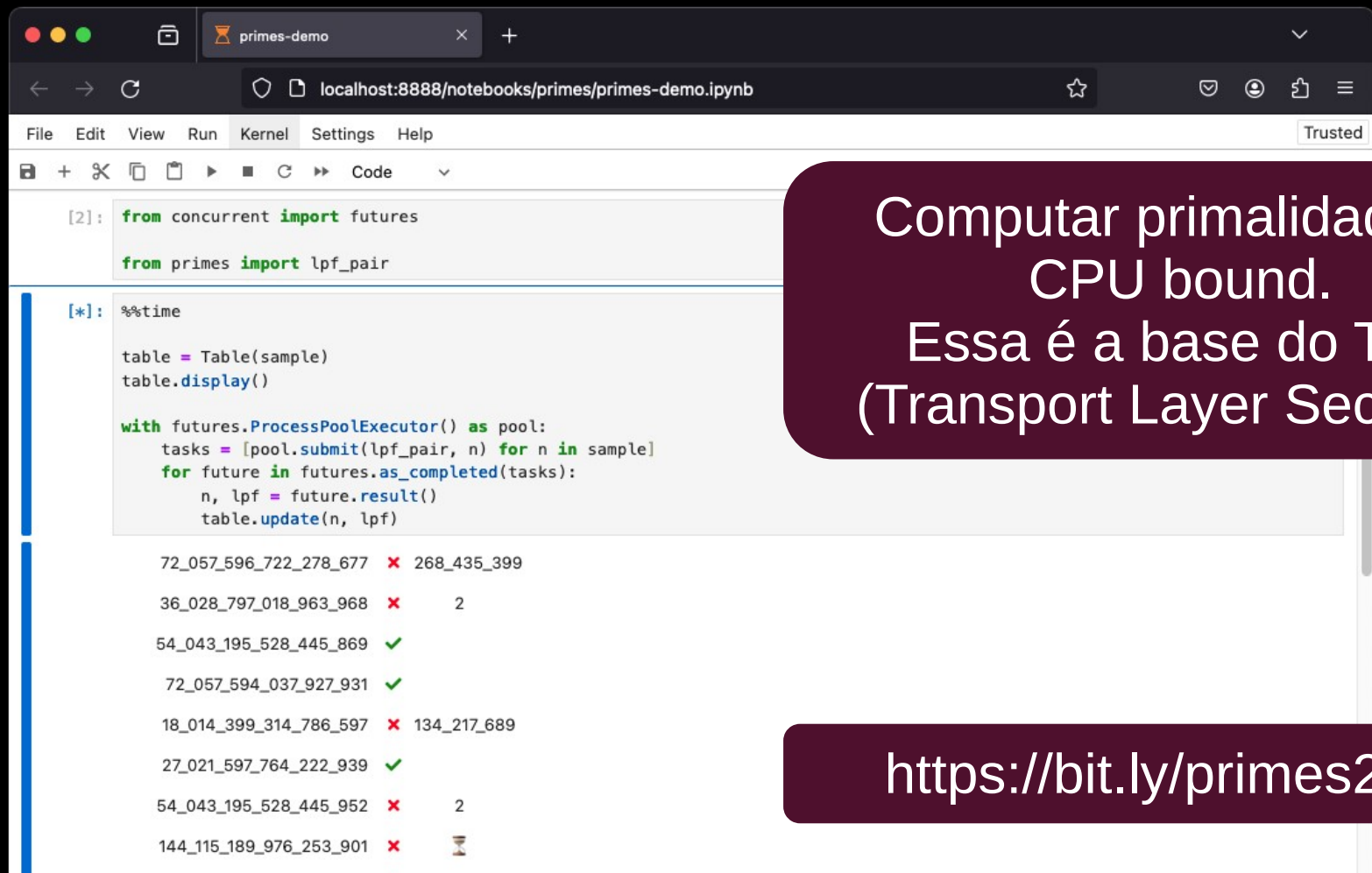
- Baixar imagem é tarefa I/O bound
  - assim como acessar arquivos, bancos de dados, APIs remotas, receber/enviar pacotes TCP/IP etc.
- Threads funcionam bem para I/O concorrente
  - Apesar da famosa GIL (Global Interpreter Lock)
    - A GIL limita todo bytecode Python a uma thread
    - Uma thread só usa um núcleo de CPU
    - O SO e certas bibliotecas escritas em C, C++, Rust, Fortran etc. não são limitadas pela GIL



# Porque não usar threads sempre

- Processos, threads, e corrotinas têm características muito diferentes
  - Utilização de recursos: núcleos de CPU, consumo de memória, custos de inicialização
- Threads e corrotinas são eficientes para processos I/O bound
- O que acontece com CPU bound?

# Demonstração: números primos



The screenshot shows a Jupyter Notebook window titled 'primes-demo' with the URL 'localhost:8888/notebooks/primes/primes-demo.ipynb'. The notebook contains two code cells. The first cell imports 'futures' from 'concurrent' and 'lpf\_pair' from 'primes'. The second cell uses a magic command '%time' to measure the execution time of a function that generates prime numbers. The output of the second cell is a list of prime number ranges, each followed by a status indicator (red 'X' for failure, green checkmark for success) and a count of primes found.

```
[2]: from concurrent import futures
     from primes import lpf_pair

[*]: %%time

     table = Table(sample)
     table.display()

     with futures.ProcessPoolExecutor() as pool:
         tasks = [pool.submit(lpf_pair, n) for n in sample]
         for future in futures.as_completed(tasks):
             n, lpf = future.result()
             table.update(n, lpf)
```

|                         |   |             |
|-------------------------|---|-------------|
| 72_057_596_722_278_677  | ✗ | 268_435_399 |
| 36_028_797_018_963_968  | ✗ | 2           |
| 54_043_195_528_445_869  | ✓ |             |
| 72_057_594_037_927_931  | ✓ |             |
| 18_014_399_314_786_597  | ✗ | 134_217_689 |
| 27_021_597_764_222_939  | ✓ |             |
| 54_043_195_528_445_952  | ✗ | 2           |
| 144_115_189_976_253_901 | ✗ | 🕒           |

Computar primalidade é  
CPU bound.  
Essa é a base do TLS  
(Transport Layer Security)

<https://bit.ly/primes2024>

# Aprendizados da demonstração

---

- Por causa da GIL, threads não servem para fazer processamento intensivo em CPU no código Python
  - Mas bibliotecas como Numpy são escritas em linguagens compiladas não limitadas pela GIL
    - mas depende da implementação da biblioteca

# Resumo: threads x processos

|   | <b>threads</b>  | <b>processos</b>   |
|---|---|--|
| <b>uso de CPU em código Python</b>            | limitado a 1 núcleo da CPU para todas as threads executando bytecode Python | cada processo pode usar um núcleo da CPU                           |
| <b>uso de memória</b>                         | memória alocada para um processo + cerca de 4MB por thread no mínimo        | uso maior de memória: memória isolada para cada processo + threads |
| <b>comunicação entre unidades de execução</b> | todas as threads podem acessar os mesmos objetos na memória                 | comunicação entre processos é mais lenta e complicada              |
| <b>custo de inicialização</b>                 | criar novo processo é caro; criar threads, nem tanto                        | custo maior, pois é multiplicado pelo número de processos          |

# Concorrência x paralelismo

---

# Concorrência x Paralelismo



*Concorrência é lidar com muitas coisas ao mesmo tempo.  
Paralelismo é fazer muitas coisas ao mesmo tempo.  
Não são a mesma coisa, mas estão relacionados.  
Uma é sobre estrutura, outro é sobre execução.  
A concorrência fornece uma maneira de estruturar uma  
solução que pode ser paralelizada  
(mas não necessariamente será).*

—Rob Pike, co-criador da linguagem Go

# Concorrência x Paralelismo

*Concorrência se faz com software.  
Paralelismo se faz com hardware  
e com programação concorrente.*

—LR, autor do Python Fluente 🤗

# Exemplo: girando pratos

**CONCURREN-  
CY WITH  
PYTHON  
3.5 ASYNC  
& AWAIT**





# Exemplo: girando pratos

A idéia essencial da programação concorrente: não precisa ter 18 braços para girar 18 pratos.



# Exemplo: girando pratos



A idéia essencial da concorrência: não precisa ter 18 braços para girar 18 pratos.

# Exemplo: ps ax

```
PID  TT  STAT      TIME COMMAND
  1   ??  Ss       2:45.09 /sbin/launchd
301   ??  Ss       6:09.65 /usr/libexec/logd
302   ??  Ss       0:00.04 /usr/libexec/smd
303   ??  Ss       0:04.16 /usr/libexec/UserEventAgent (System)
305   ??  Ss       0:01.65 /System/Library/PrivateFrameworks/Uninstall.framework/Resources/uninstalld
306   ??  Ss       0:49.87 /System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/FSEvents.framework/Versions/A/Support/fseventsd
307   ??  Ss       0:06.83 /System/Library/PrivateFrameworks/MediaRemote.framework/Support/mediaremoted
310   ??  Ss       0:03.60 /usr/sbin/systemstats --daemon
313   ??  Ss       0:45.56 /usr/libexec/configd
314   ??  Ss       0:00.01 endpointsecurityd
315   ??  Ss       0:20.93 /System/Library/CoreServices/powerd.bundle/powerd
316   ??  Ss       0:00.01 /usr/libexec/IOMFB_bics_daemon
317   ??  Ss       0:01.44 /System/Library/PrivateFrameworks/BiomeStreams.framework/Support/biomed
319   ??  Ss       0:01.14 /usr/libexec/amfid

...

13953  ??  S        0:00.02 /System/Library/PrivateFrameworks/CharacterPicker.framework/Versions/A/XPCServices/CharacterPicker.xpcservice
13956  ??  S        0:00.10 /System/Library/Frameworks/CoreServices.framework/Frameworks/Metadata.framework/Versions/A/Support/metadata
12893 s000  Ss       0:00.01 login -pf luciano
12894 s000  S        0:00.18 -zsh
13963 s000  R+       0:00.00 ps ax
13055 s002  Ss+      0:00.02 /bin/zsh -il
```

# Exemplo: **ps** **ax**

```
% ps ax | wc -l
      632
% sysctl -n hw.ncpu
12
```

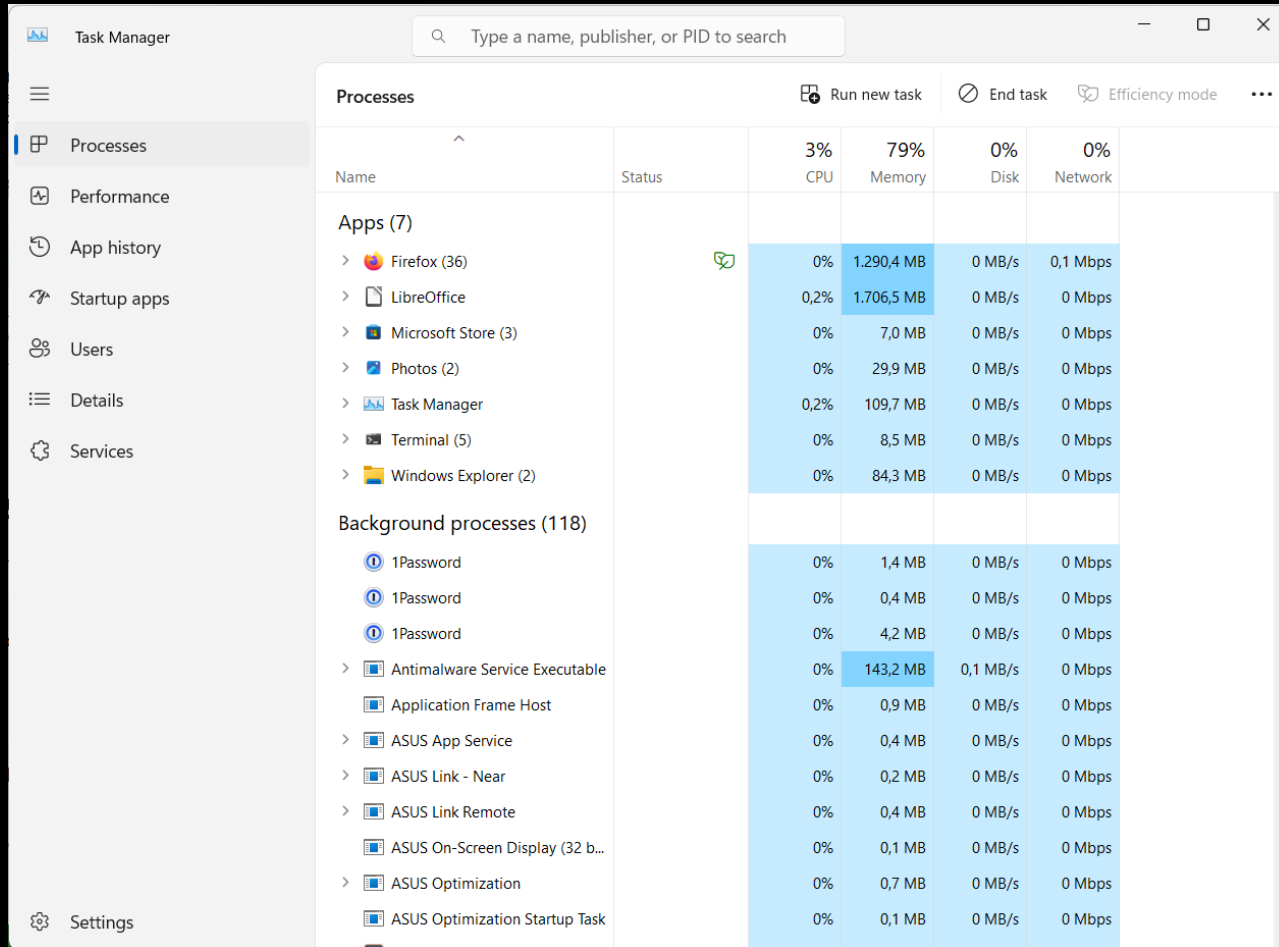
- 632 processos rodando no macOS Sonoma 14.1.1
- Mas a máquina tem 12 CPUs

# Exemplo: **ps ax**

```
$ ps ax | wc -l
307
$ lscpu | grep "CPU(s) : "
CPU(s) : 4
NUMA node0 CPU(s) : 0-3
```

- 307 processos rodando no Ubuntu Linux 22.04
- Mas a máquina tem 4 "CPUs"
  - na real, 2 CPUs com hyperthreading

# Exemplo: processos no Windows 11

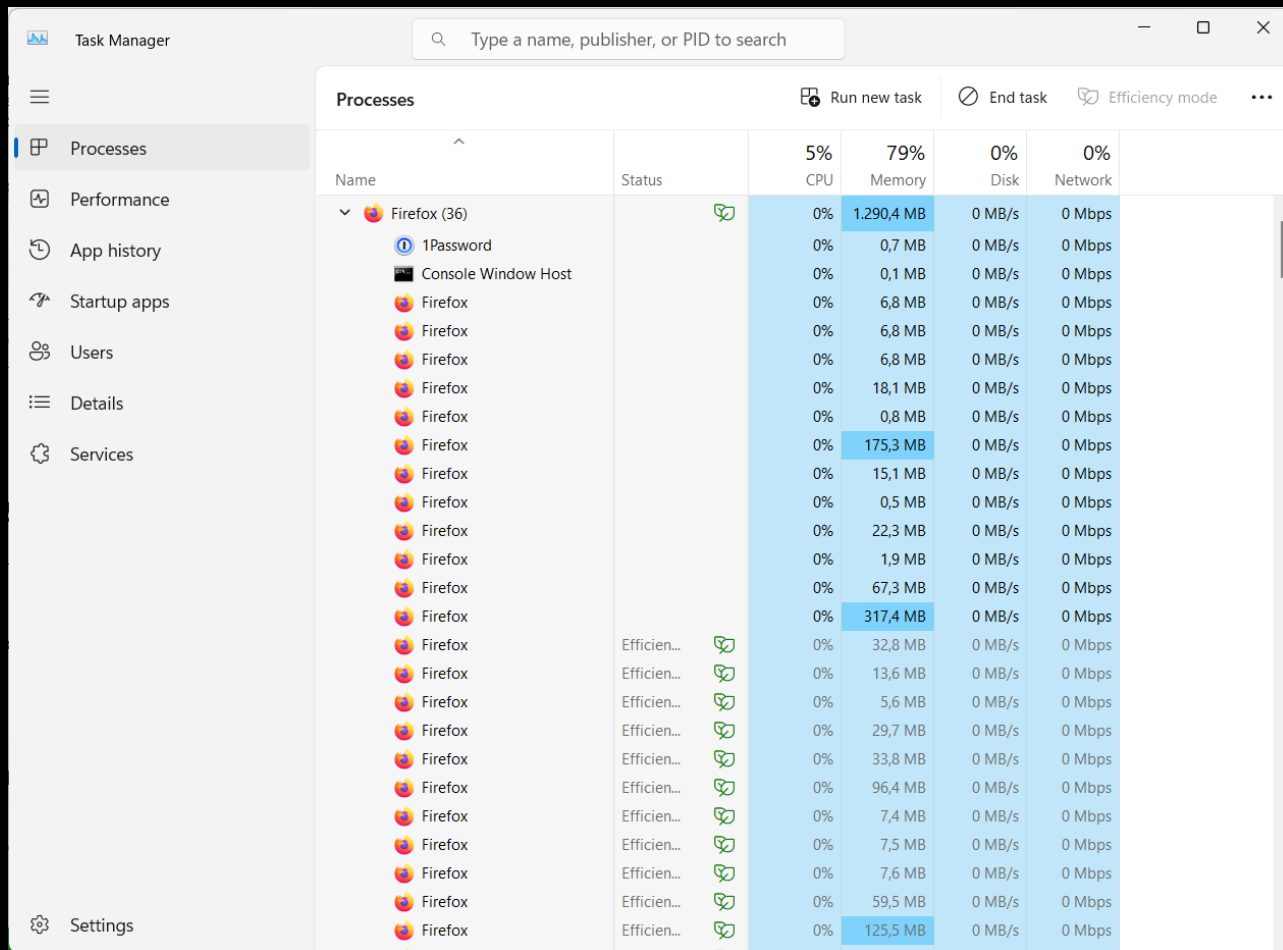


The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. The window title is 'Task Manager' and it has a search bar at the top. The left sidebar shows navigation options: Processes, Performance, App history, Startup apps, Users, Details, and Services. The main area displays a list of processes categorized into 'Apps (7)' and 'Background processes (118)'. The columns are: Name, Status, CPU, Memory, Disk, and Network. The 'Apps (7)' section lists Firefox (36), LibreOffice, Microsoft Store (3), Photos (2), Task Manager, Terminal (5), and Windows Explorer (2). The 'Background processes (118)' section lists various system services and applications, including 1Password, Antimalware Service Executable, Application Frame Host, ASUS App Service, ASUS Link - Near, ASUS Link Remote, ASUS On-Screen Display (32 b...), ASUS Optimization, and ASUS Optimization Startup Task.

| Name  | Status | 3% CPU | 79% Memory | 0% Disk  | 0% Network |
|---|--------|--------|------------|----------|------------|
| <strong>Apps (7)</strong>                   |        |        |            |          |            |
| > Firefox (36)                              |        | 0%     | 1.290,4 MB | 0 MB/s   | 0,1 Mbps   |
| > LibreOffice                               |        | 0,2%   | 1.706,5 MB | 0 MB/s   | 0 Mbps     |
| > Microsoft Store (3)                       |        | 0%     | 7,0 MB     | 0 MB/s   | 0 Mbps     |
| > Photos (2)                                |        | 0%     | 29,9 MB    | 0 MB/s   | 0 Mbps     |
| > Task Manager                              |        | 0,2%   | 109,7 MB   | 0 MB/s   | 0 Mbps     |
| > Terminal (5)                              |        | 0%     | 8,5 MB     | 0 MB/s   | 0 Mbps     |
| > Windows Explorer (2)                      |        | 0%     | 84,3 MB    | 0 MB/s   | 0 Mbps     |
| <strong>Background processes (118)</strong> |        |        |            |          |            |
| 1Password                                   |        | 0%     | 1,4 MB     | 0 MB/s   | 0 Mbps     |
| 1Password                                   |        | 0%     | 0,4 MB     | 0 MB/s   | 0 Mbps     |
| 1Password                                   |        | 0%     | 4,2 MB     | 0 MB/s   | 0 Mbps     |
| > Antimalware Service Executable            |        | 0%     | 143,2 MB   | 0,1 MB/s | 0 Mbps     |
| Application Frame Host                      |        | 0%     | 0,9 MB     | 0 MB/s   | 0 Mbps     |
| > ASUS App Service                          |        | 0%     | 0,4 MB     | 0 MB/s   | 0 Mbps     |
| > ASUS Link - Near                          |        | 0%     | 0,2 MB     | 0 MB/s   | 0 Mbps     |
| > ASUS Link Remote                          |        | 0%     | 0,4 MB     | 0 MB/s   | 0 Mbps     |
| ASUS On-Screen Display (32 b...             |        | 0%     | 0,1 MB     | 0 MB/s   | 0 Mbps     |
| > ASUS Optimization                         |        | 0%     | 0,7 MB     | 0 MB/s   | 0 Mbps     |
| ASUS Optimization Startup Task              |        | 0%     | 0,1 MB     | 0 MB/s   | 0 Mbps     |

- 7 aplicativos
  - Firefox: 36 processos
- 118 processos em background
- 107 processos do Windows

# Exemplo: processos no Windows 11

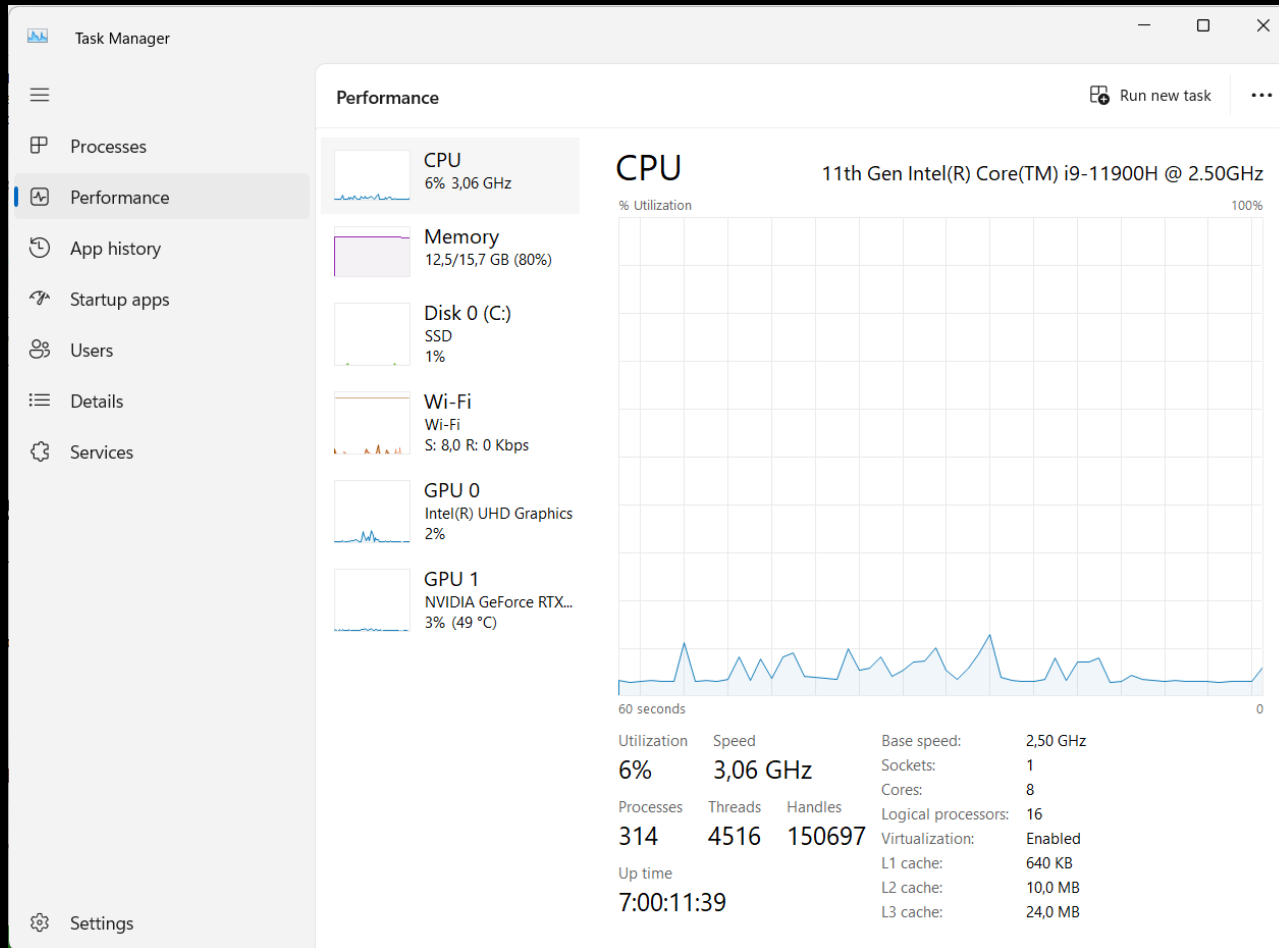


The screenshot shows the Windows Task Manager 'Processes' tab. A search bar at the top contains the text 'Type a name, publisher, or PID to search'. The left sidebar shows 'Processes' selected. The main table lists 36 processes, all named 'Firefox'. The first process is expanded, showing sub-processes like '1Password' and 'Console Window Host'. The table columns are: Name, Status, 5% CPU, 79% Memory, 0% Disk, and 0% Network. The 'Memory' column is highlighted in blue. The 'Status' column shows green checkmarks for most processes and 'Efficien...' for others. The 'Settings' icon is visible in the bottom left corner.

| Name                | Status      | 5% CPU | 79% Memory | 0% Disk | 0% Network |
|---------------------|-------------|--------|------------|---------|------------|
| Firefox (36)        |             | 0%     | 1,290,4 MB | 0 MB/s  | 0 Mbps     |
| 1Password           |             | 0%     | 0,7 MB     | 0 MB/s  | 0 Mbps     |
| Console Window Host |             | 0%     | 0,1 MB     | 0 MB/s  | 0 Mbps     |
| Firefox             |             | 0%     | 6,8 MB     | 0 MB/s  | 0 Mbps     |
| Firefox             |             | 0%     | 6,8 MB     | 0 MB/s  | 0 Mbps     |
| Firefox             |             | 0%     | 6,8 MB     | 0 MB/s  | 0 Mbps     |
| Firefox             |             | 0%     | 18,1 MB    | 0 MB/s  | 0 Mbps     |
| Firefox             |             | 0%     | 0,8 MB     | 0 MB/s  | 0 Mbps     |
| Firefox             |             | 0%     | 175,3 MB   | 0 MB/s  | 0 Mbps     |
| Firefox             |             | 0%     | 15,1 MB    | 0 MB/s  | 0 Mbps     |
| Firefox             |             | 0%     | 0,5 MB     | 0 MB/s  | 0 Mbps     |
| Firefox             |             | 0%     | 22,3 MB    | 0 MB/s  | 0 Mbps     |
| Firefox             |             | 0%     | 1,9 MB     | 0 MB/s  | 0 Mbps     |
| Firefox             |             | 0%     | 67,3 MB    | 0 MB/s  | 0 Mbps     |
| Firefox             |             | 0%     | 317,4 MB   | 0 MB/s  | 0 Mbps     |
| Firefox             | Efficien... | 0%     | 32,8 MB    | 0 MB/s  | 0 Mbps     |
| Firefox             | Efficien... | 0%     | 13,6 MB    | 0 MB/s  | 0 Mbps     |
| Firefox             | Efficien... | 0%     | 5,6 MB     | 0 MB/s  | 0 Mbps     |
| Firefox             | Efficien... | 0%     | 29,7 MB    | 0 MB/s  | 0 Mbps     |
| Firefox             | Efficien... | 0%     | 33,8 MB    | 0 MB/s  | 0 Mbps     |
| Firefox             | Efficien... | 0%     | 96,4 MB    | 0 MB/s  | 0 Mbps     |
| Firefox             | Efficien... | 0%     | 7,4 MB     | 0 MB/s  | 0 Mbps     |
| Firefox             | Efficien... | 0%     | 7,5 MB     | 0 MB/s  | 0 Mbps     |
| Firefox             | Efficien... | 0%     | 7,6 MB     | 0 MB/s  | 0 Mbps     |
| Firefox             | Efficien... | 0%     | 59,5 MB    | 0 MB/s  | 0 Mbps     |
| Firefox             | Efficien... | 0%     | 125,5 MB   | 0 MB/s  | 0 Mbps     |

- 36 processos do Firefox
  - 1Password add-on
  - Console (?)
  - Abas e janelas

# Exemplo: processos no Windows 11





# Exemplo: processos no Windows 11

Task Manager

Type a name, publisher, or PID to search

Users Run new task Disconnect Manage user accounts

| User                          | Status    | 5% CPU | 79% Memory | 0% Disk  | 0% Network |
|-------------------------------|-----------|--------|------------|----------|------------|
| luciano@ramalho.org (172)     |           | 3,1%   | 4.007,3 MB | 0,1 MB/s | 0 Mbps     |
| 1Password                     |           | 0%     | 1,6 MB     | 0 MB/s   | 0 Mbps     |
| 1Password                     |           | 0%     | 0,7 MB     | 0 MB/s   | 0 Mbps     |
| 1Password                     |           | 0%     | 1,4 MB     | 0 MB/s   | 0 Mbps     |
| 1Password                     |           | 0%     | 0,4 MB     | 0 MB/s   | 0 Mbps     |
| 1Password                     |           | 0%     | 4,2 MB     | 0 MB/s   | 0 Mbps     |
| Application Frame Host        |           | 0%     | 1,0 MB     | 0 MB/s   | 0 Mbps     |
| ASUS On-Screen Display (...)  |           | 0%     | 0,1 MB     | 0 MB/s   | 0 Mbps     |
| ASUS Optimization Startu...   |           | 0%     | 0,1 MB     | 0 MB/s   | 0 Mbps     |
| ASUS ProArt Host              |           | 0%     | 0,9 MB     | 0 MB/s   | 0 Mbps     |
| ASUS ScreenXpert              |           | 0%     | 0,2 MB     | 0 MB/s   | 0 Mbps     |
| ASUS Software Manager A...    |           | 0%     | 2,7 MB     | 0 MB/s   | 0 Mbps     |
| Background Task Host          | Suspended | 0%     | 0 MB       | 0 MB/s   | 0 Mbps     |
| Background Task Host          | Suspended | 0%     | 0 MB       | 0 MB/s   | 0 Mbps     |
| Background Task Host          | Suspended | 0%     | 0 MB       | 0 MB/s   | 0 Mbps     |
| Client Server Runtime Proc... |           | 0%     | 1,4 MB     | 0 MB/s   | 0 Mbps     |
| COM Surrogate                 |           | 0%     | 1,2 MB     | 0 MB/s   | 0 Mbps     |
| COM Surrogate                 |           | 0%     | 0,1 MB     | 0 MB/s   | 0 Mbps     |
| COM Surrogate                 |           | 0%     | 2,1 MB     | 0 MB/s   | 0 Mbps     |
| COM Surrogate                 |           | 0%     | 0,1 MB     | 0 MB/s   | 0 Mbps     |
| COM Surrogate                 |           | 0%     | 0,1 MB     | 0 MB/s   | 0 Mbps     |
| Console Window Host           |           | 0%     | 0,1 MB     | 0 MB/s   | 0 Mbps     |
| Console Window Host           |           | 0%     | 0,1 MB     | 0 MB/s   | 0 Mbps     |
| Console Window Host           |           | 0%     | 0,1 MB     | 0 MB/s   | 0 Mbps     |
| Console Window Host           |           | 0%     | 0,1 MB     | 0 MB/s   | 0 Mbps     |
| Console Window Host           |           | 0%     | 0,1 MB     | 0 MB/s   | 0 Mbps     |

Processes Performance App history Startup apps Users Details Services Settings

# Processos x threads

---

# Processos

---

- Escalonamento feito pelo sistema operacional
  - SO multi-tarefa preemptivo
    - Suspende processos para dar a vez a outros
- Memória isolada pelo sistema operacional
  - Troca de dados através de pipes, sockets
    - Há mecanismos avançados para compartilhar memória, mas eles não suportam "objetos" apenas bytes

# Threads

---

- Escalonamento feito pelo sistema operacional
  - Threads do SO em *userland*
    - Suspende threads para dar a vez a outras
  - No Cpython, existe a GIL (Global Interpreter Lock)
    - Só uma thread de código Python por vez
    - Extensões binárias podem executar várias threads
- Memória compartilhada
  - Suporte pleno a objetos Python

# Programação assíncrona

---

# Exemplo: Judit Pólgar\*

- 20 adversários
- 1 minuto por jogada
  - Pólgar 5s, adversário 55s
- Média de 30 jogadas por partida
- 20 partidas sequenciais:  
 $20 \times 30\text{min} = 600\text{min} = 10\text{h}$



# Exemplo: Judit Pólgar

- Partidas “simultâneas”
- Pólgar dá o lance inicial e vai pro próximo tabuleiro
- Uma volta completa leva  $20 \times 5s = 40s$
- 20 partidas terminam em  $30 \times 40s = 1200s = 20min$

