ThoughtWorks®

com exemplos em Python

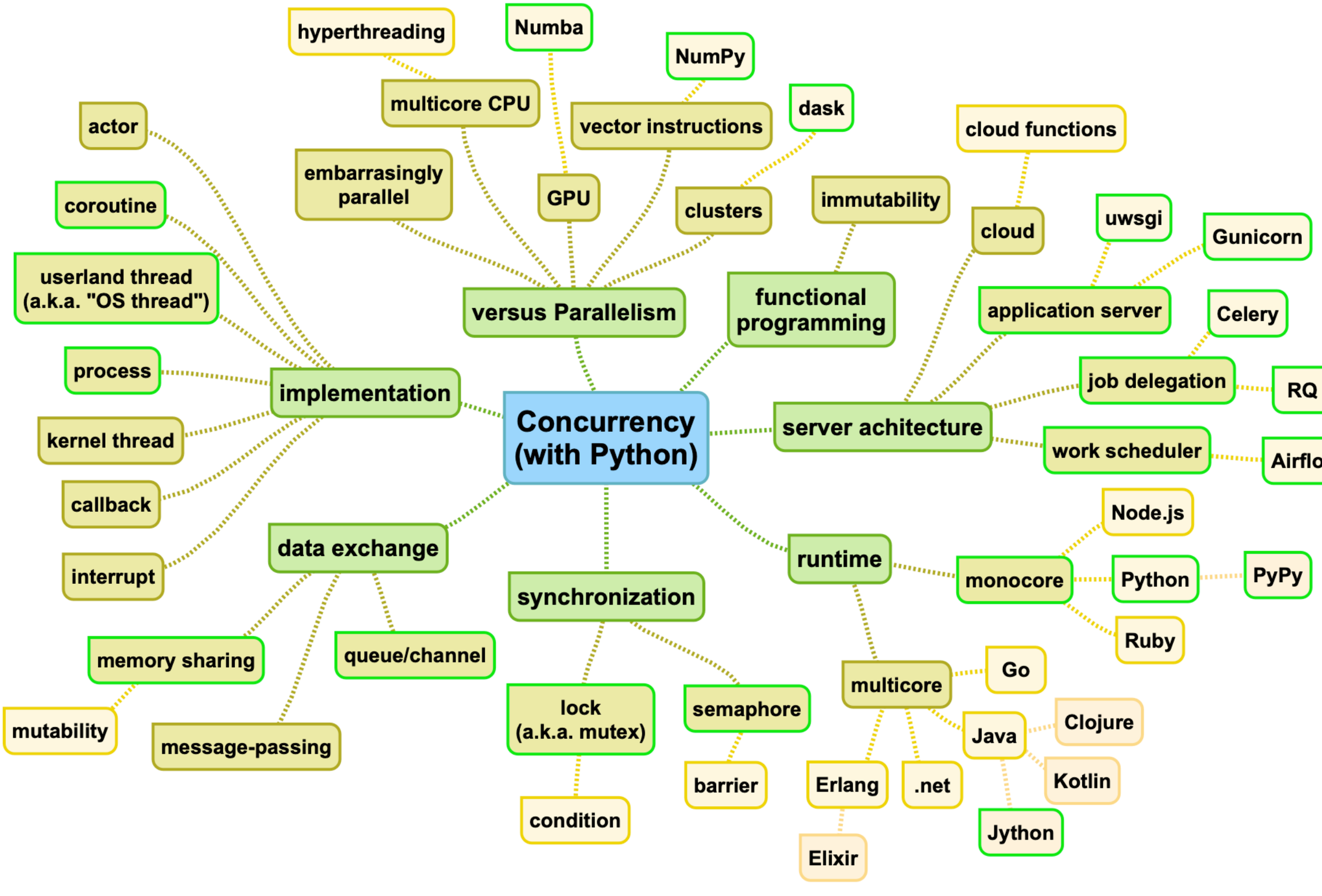
MODELOS DE CONCORRÊNCIA

Diferentes modos de construir sistemas que
fazem várias coisas "ao mesmo tempo"

Março de 2020

Como foi possível que YouTube e Instagram atingissem escala planetária usando principalmente Python, que só avança uma linha de execução (thread) de cada vez?

MAPA MENTAL: GUIA PARA PESQUISAR MAIS



ThoughtWorks®

CONCORRÊNCIA

✕

PARALELISMO

Conceitos relacionados porém distintos

GIRANDO PRATOS

A idéia essencial de concorrência: não é preciso 18 braços para girar 18 pratos.

Dá para fazer com 2 braços, se você souber quando cada prato precisa de sua intervenção para continuar girando.

EXIBIÇÃO DE XADREZ: PARTIDAS SIMULTÂNEAS

Exemplo de Michael Grinberg em
“Asynchronous Python for the Complete Beginner” (PyCon 2017)

EXIBIÇÃO DE XADREZ: PARTIDAS SIMULTÂNEAS

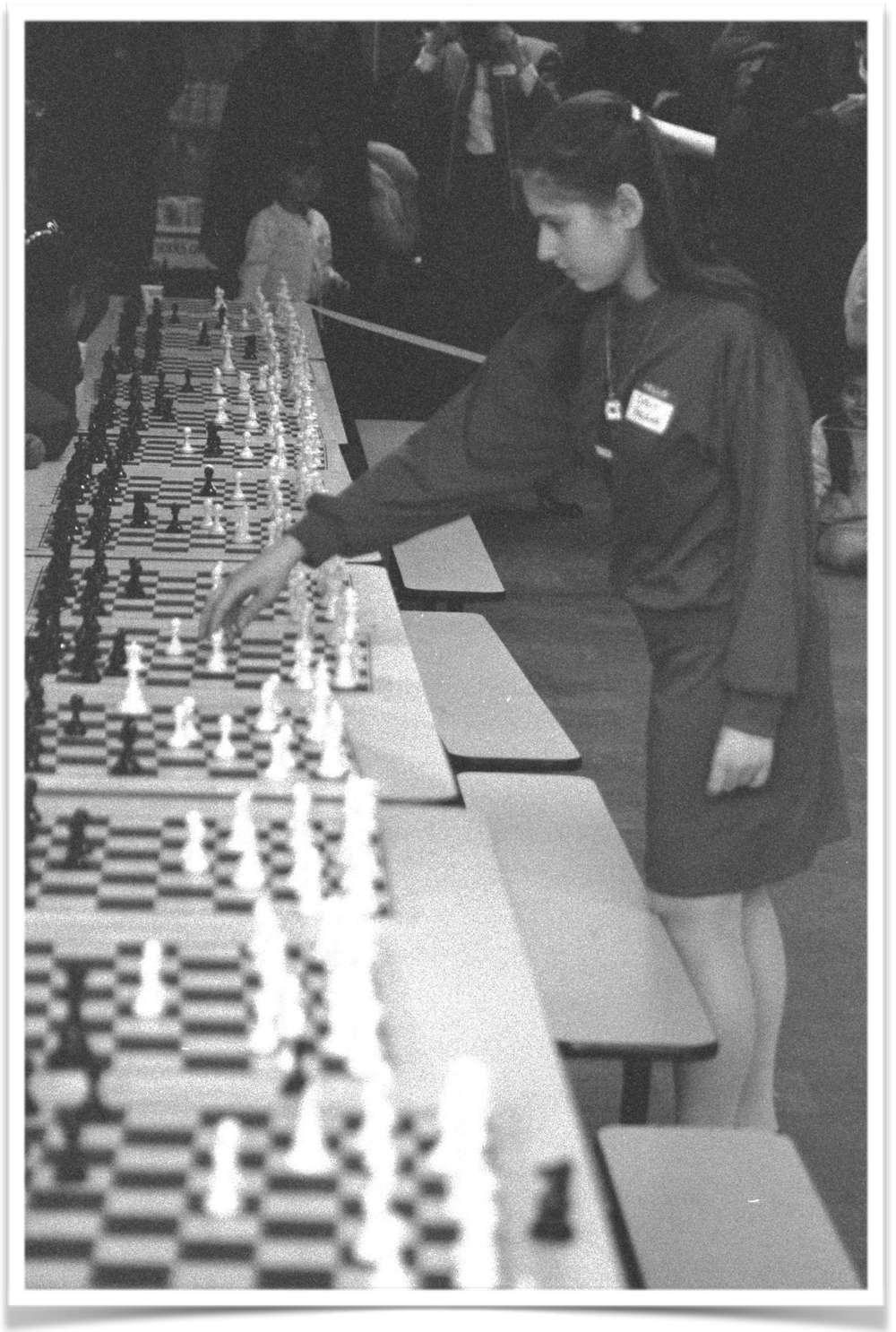
Exemplo de Michael Grinberg em
“Asynchronous Python for the Complete Beginner” (PyCon 2017)

Judit Polgár em 1992 (16 anos) Foto de Ed Yourdon

PARTIDAS SEQUENCIAIS

Dados (hipotéticos):

- 24 oponentes
- Polgár faz um lance em **5s**
- Cada oponente faz um lance em **55s**
- Partida média tem 30 pares de lances



Significa:

- Cada partida tuda **30 minutos**
- 24 partidas em sequência: **12 horas**

```
>>> 30 * 24
720
>>> 720 / 60
12.0
```

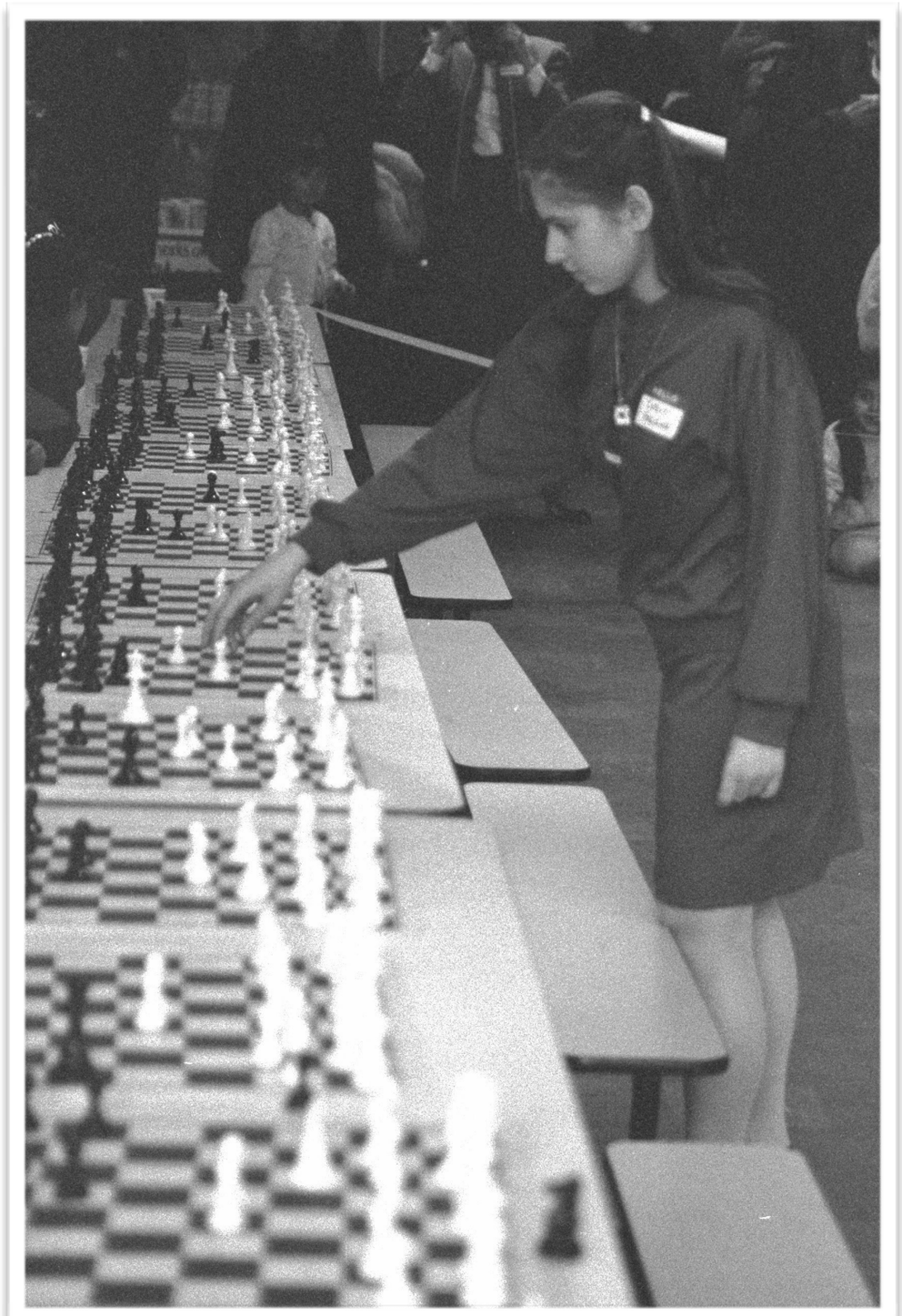
PARTIDAS SIMULTÂNEAS EM MODO ASSÍNCRONO

- Polgár faz 1º lance na 1ª partida
- Enquanto oponente pensa, ela faz lance na 2ª partida, 3ª partida...
- Polgár faz uma volta completa pelas 24 partidas em **2 minutos**

```
>>> 5 * 24
120
```

- Ao completar a primeira volta, o 1º oponente já jogou e ela pode fazer seu 2º lance
- Polgár vence 24 partidas de 30 lances em **1 hora**!

```
>>> 30 * 2
60
```



Polgár não perde tempo esperando!

CONCORRÊNCIA × PARALELISMO

Concurrency vs. parallelism

Concurrency is about dealing with lots of things at once.

Parallelism is about doing lots of things at once.

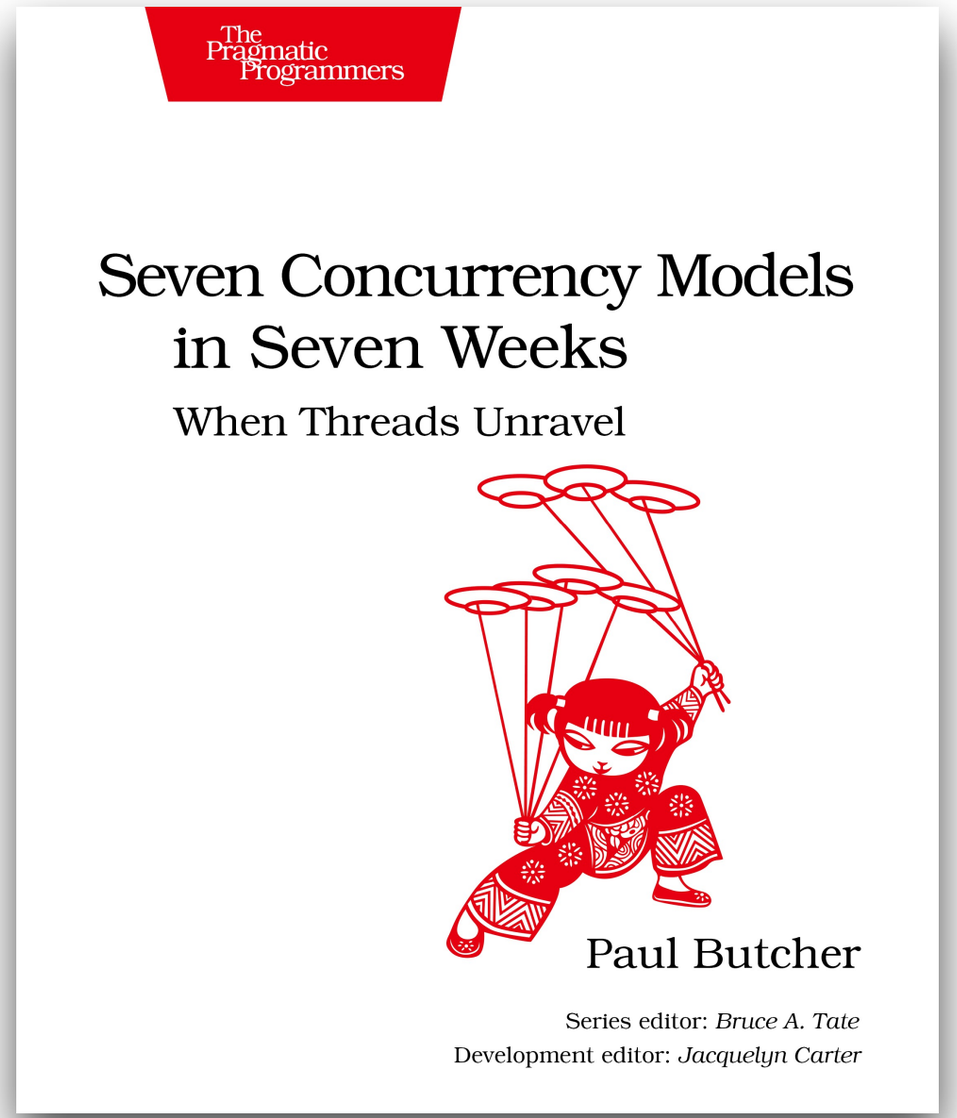
Not the same, but related.

Concurrency is about structure, parallelism is about execution.

Concurrency provides a way to structure a solution to solve a problem that may (but not necessarily) be parallelizable.

Rob Pike - 'Concurrency Is Not Parallelism'
https://www.youtube.com/watch?v=cN_DpYBzKso

DESFIANDO THREADS



- *Seven Concurrency Models in Seven Weeks — When Threads Unravel* (Paul Butcher)
- Callbacks: esse livro nem menciona
- Capítulo 1: problemas no uso de threads e locks
- Demais capítulos: abstrações mais poderosas
  - Actors, CSP, STM, data parallelism...
- Suporte nativo em linguagens
  - Erlang, Elixir, Clojure, Go, Cilk, Haskell...

ThoughtWorks®

LABORATÓRIO: FLAGS

Um cliente HTTP simples

EXECUTE OS EXEMPLOS DE DOWNLOAD DE BANDEIRAS

1. Clone:
   https://github.com/pythonfluente/concorrencia2020

2. Encontre os exemplos **flags*.py** em **`flags/`**

3. Execute:
   - **`flags.py`**
   - **`flags_threadpool.py`**
   - **`flags_asyncio.py`***

* Requer a externa biblioteca aiohttp + dependências

SEQUENCIAL × THREAD POOL (1)

Kaleidoscope

flags.py | flags_threadpool.py

A flags flags.py

B flags flags_threadpool.py

```
"""Download flags of top 20 countries by population

Sequential version

Sample run::

    $ python3 flags.py
    BD BR CD CN DE EG ET FR ID IN IR JP MX NG PH PK RU TR US VN
    20 downloads in 5.49s

"""

import os
import time


import urllib.request

POP20_CC = ('CN IN US ID BR PK NG BD RU JP '
            'MX PH VN ET EG DE IR TR CD FR').split()

BASE_URL = 'http://flupy.org/data/flags'

DEST_DIR = 'downloaded/'



def save_flag(img, filename):
    path = os.path.join(DEST_DIR, filename)
    with open(path, 'wb') as fp:
        fp.write(img)
```

```
"""Download flags of top 20 countries by population

ThreadPoolExecutor version

Sample run::

    $ python3 flags_threadpool.py
    DE FR BD CN EG RU IN TR VN ID JP BR NG MX PK ET PH CD US IR
    20 downloads in 0.35s

"""

import os
import time
from concurrent import futures  # <1>

import urllib.request

POP20_CC = ('CN IN US ID BR PK NG BD RU JP '
            'MX PH VN ET EG DE IR TR CD FR').split()

BASE_URL = 'http://flupy.org/data/flags'

DEST_DIR = 'downloaded/'

MAX_WORKERS = 20  # <2>


def save_flag(img, filename):
    path = os.path.join(DEST_DIR, filename)
    with open(path, 'wb') as fp:
        fp.write(img)
```

Blocks Fluid Unified Ignore whitespace Change 3 of 8

SEQUENCIAL × THREAD POOL (2)

Kaleidoscope

flags.py | flags_threadpool.py

A flags flags.py

```
28          fp.write(img)
29
30
31  def get_flag(cc):
32      cc = cc.lower()
33      url = f'{BASE_URL}/{cc}/{cc}.gif'
34      resp = urllib.request.urlopen(url)
35      return resp.read()
36
37
38  def download_many(cc_list):
39      for cc in sorted(cc_list):
40          image = get_flag(cc)
41          print(cc, end=' ', flush=True)
42          save_flag(image, cc.lower() + '.gif')
43
44      return len(cc_list)
45
46
47  def main():
48      t0 = time.perf_counter()
49      count = download_many(POP20_CC)
50      elapsed = time.perf_counter() - t0
51      print(f'\n{count} downloads in {elapsed:.2f}s')
52
53
54  if __name__ == '__main__':
55      main()
```

B flags flags_threadpool.py

```
32          fp.write(img)
33
34
35  def get_flag(cc):
36      cc = cc.lower()
37      url = f'{BASE_URL}/{cc}/{cc}.gif'
38      resp = urllib.request.urlopen(url)
39      return resp.read()
40
41
42  def download_one(cc):  # <3>
43      image = get_flag(cc)
44      print(cc, end=' ', flush=True)
45      save_flag(image, cc.lower() + '.gif')
46      return cc
47
48
49  def download_many(cc_list):
50      workers = min(MAX_WORKERS, len(cc_list))               # <4>
51      with futures.ThreadPoolExecutor(workers) as executor:  # <5>
52          res = executor.map(download_one, sorted(cc_list))  # <6>
53
54      return len(list(res))
55
56
57  def main():
58      t0 = time.perf_counter()
59      count = download_many(POP20_CC)
60      elapsed = time.perf_counter() - t0
61      print(f'\n{count} downloads in {elapsed:.2f}s')
62
63
64  if __name__ == '__main__':
65      main()
```

Blocks Fluid Unified

Ignore whitespace Change 5 of 8

THREAD POOL × ASYNCIO (1)

Kaleidoscope

flags_threadpool.py | flags_asyncio.py

A flags flags_threadpool.py

B flags flags_asyncio.py

```
"""Download flags of top 20 countries by population

ThreadPoolExecutor version

Sample run::

    $ python3 flags_threadpool.py
    DE FR BD CN EG RU IN TR VN ID JP BR NG MX PK ET PH CD US IR
    20 downloads in 0.35s

"""

import os
import time


from concurrent import futures  # <1>

import urllib.request

POP20_CC = ('CN IN US ID BR PK NG BD RU JP '
            'MX PH VN ET EG DE IR TR CD FR').split()

BASE_URL = 'http://flupy.org/data/flags'

DEST_DIR = 'downloaded/'

MAX_WORKERS = 20  # <2>


def save_flag(img, filename):
    path = os.path.join(DEST_DIR, filename)
    with open(path, 'wb') as fp:
        fp.write(img)
```

```
"""Download flags of top 20 countries by population

asyncio + aiottp version

Sample run::

    $ python3 flags_asyncio.py
    CN EG BR IN ID RU NG VN JP DE TR PK FR ET MX PH US IR CD BD
    20 downloads in 0.35s

"""

import os
import time
import asyncio  # <1>

import aiohttp  # <2>


POP20_CC = ('CN IN US ID BR PK NG BD RU JP '
            'MX PH VN ET EG DE IR TR CD FR').split()

BASE_URL = 'http://flupy.org/data/flags'

DEST_DIR = 'downloaded/'


def save_flag(img, filename):
    path = os.path.join(DEST_DIR, filename)
    with open(path, 'wb') as fp:
        fp.write(img)
```

Blocks Fluid Unified Ignore whitespace Change 3 of 16

THREAD POOL × ASYNCIO (2)

Kaleidoscope

flags_threadpool.py | flags_asyncio.py

A flags flags_threadpool.py

```
33
34
35 def get_flag(cc):
36     cc = cc.lower()
37     url = f'{BASE_URL}/{cc}/{cc}.gif'
38     resp = urllib.request.urlopen(url)
39     return resp.read()
40
41
42 def download_one(cc):  # <3>
43     image = get_flag(cc)
44     print(cc, end=' ', flush=True)
45     save_flag(image, cc.lower() + '.gif')
46     return cc
47
48
49 def download_many(cc_list):
50     workers = min(MAX_WORKERS, len(cc_list))               # <4>
51     with futures.ThreadPoolExecutor(workers) as executor:  # <5>
52         res = executor.map(download_one, sorted(cc_list))  # <6>
53
54     return len(list(res))
55
56
57 def main():
58     t0 = time.perf_counter()
59     count = download_many(POP20_CC)
60     elapsed = time.perf_counter() - t0
61     print(f'\n{count} downloads in {elapsed:.2f}s')
62
63
64 if __name__ == '__main__':
65     main()
66
```

B flags flags_asyncio.py

```
32
33
34 async def get_flag(session, cc):          # <3>
35     cc = cc.lower()
36     url = f'{BASE_URL}/{cc}/{cc}.gif'
37     async with session.get(url) as resp:  # <4>
38         return await resp.read()          # <5>
39
40
41 async def download_one(session, cc):      # <6>
42     image = await get_flag(session, cc)   # <7>
43     print(cc, end=' ', flush=True)
44     save_flag(image, cc.lower() + '.gif')
45     return cc
46
47
48 async def download_many(cc_list):
49     async with aiohttp.ClientSession() as session:                # <8>
50         tasks = [asyncio.create_task(download_one(session, cc))   # <9>
51                  for cc in sorted(cc_list)]
52         res = await asyncio.gather(*tasks)                        # <10>
53
54     return len(res)
55
56
57 def main():  # <11>
58     t0 = time.perf_counter()
59     count = asyncio.run(download_many(POP20_CC))
60     elapsed = time.perf_counter() - t0
61     print(f'\n{count} downloads in {elapsed:.2f}s')
62
63
64 if __name__ == '__main__':
65     main()
66
```

Blocks Fluid Unified Ignore whitespace Change 10 of 16