Parallel Programming in Python – a very, very short introduction

Agenda

1. Computer architectures, data structures, and opportunities
2. Case 1 – computing a distance matrix
3. Case 2 – computing the average of a large sample

The scope of the technique that will be presented [1]:

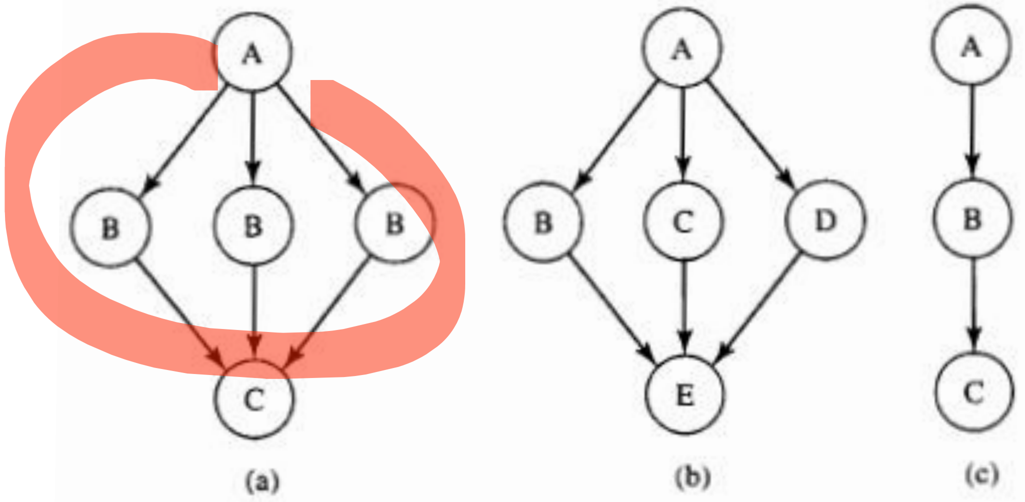
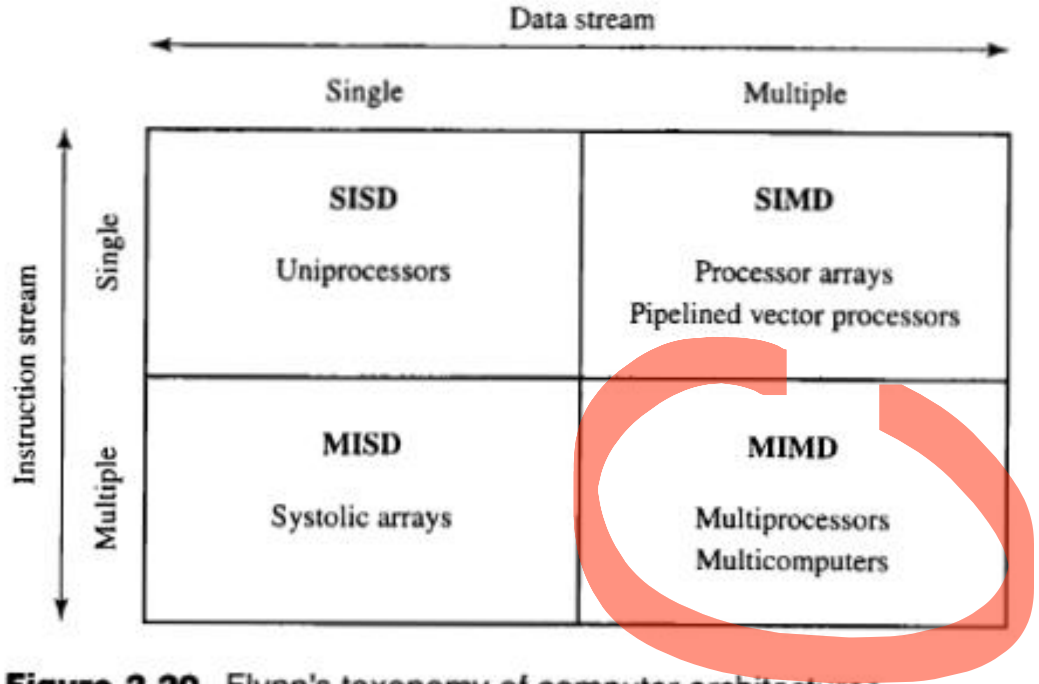


Figure 1.4 Parallelism in data dependence graphs. Vertices represent tasks. The letter inside a vertex indicates the operation being performed. Edges denote dependences among tasks. (a) A graph exhibiting data parallelism. Three tasks may concurrently apply operation B to different operands. (b) A graph exhibiting functional parallelism. Tasks performing operations B, C, and D may be performed concurrently. (c) A purely sequential dependence graph. However, if all tasks take the same amount of time to execute and multiple problem instances need to be processed, operation C may be performed on instance *i* while operation B is performed on instance *i* + 1 and operation A is performed on instance *i* + 2. This structure is called pipelining.


Data stream
Instruction stream

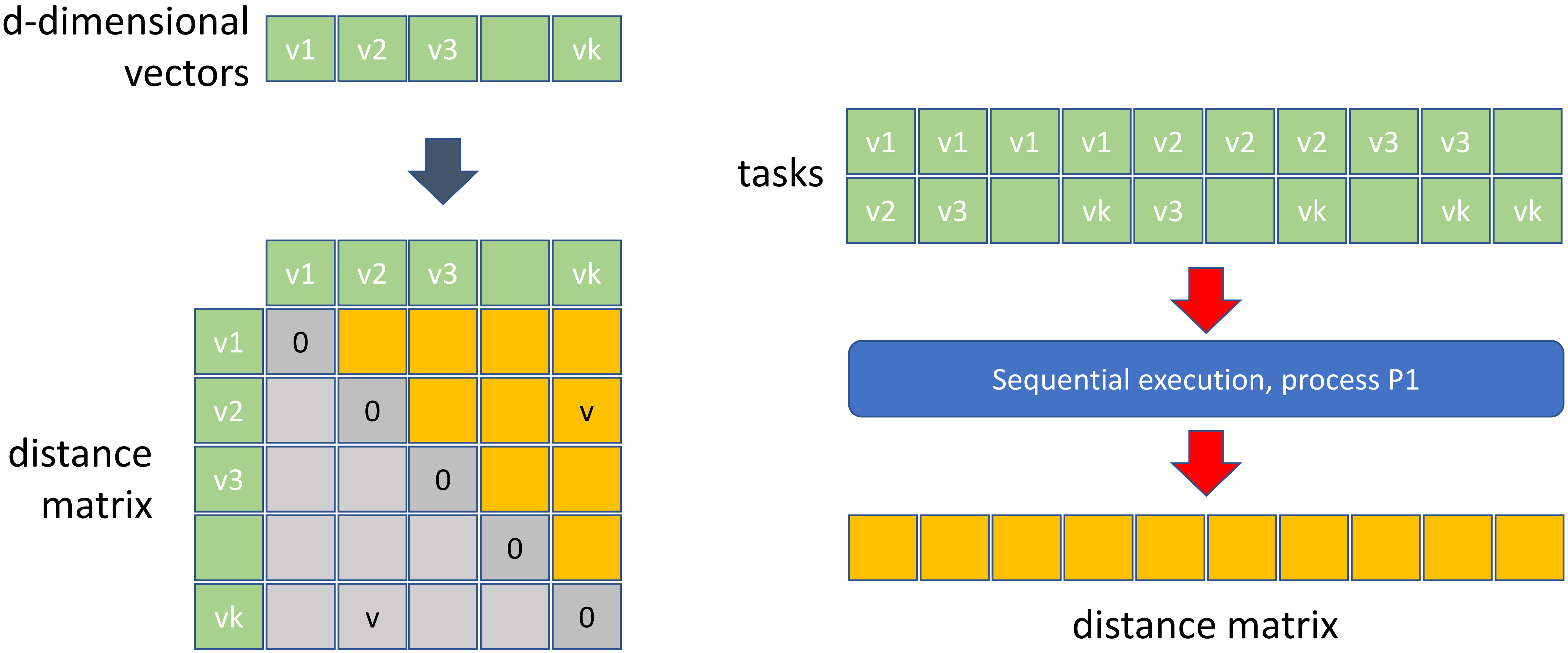
		Data stream	
		Single	Multiple
Instruction stream	Single	SISD Uniprocessors	SIMD Processor arrays Pipelined vector processors
	Multiple	MISD Systolic arrays	MIMD Multiprocessors Multicomputers

Figure 2.20 Flynn's taxonomy of computer architectures.

Resumo. *Aplicações Bag-of-Tasks (BoT) são aplicações paralelas compostas de tarefas independentes (ou seja, embaraçosamente paralelas), que não se comunicam entre si, podem depender de um ou mais arquivos de entrada e podem ser executadas em qualquer orden* [2]

Case 1 – computing a distance matrix

Case 1 - computing a distance matrix, sequential scheme



d-dimensional vectors

v1	v2	v3		vk

distance matrix

	v1	v2	v3		vk
v1	0				
v2		0			v
v3			0		
				0	
vk		v			0

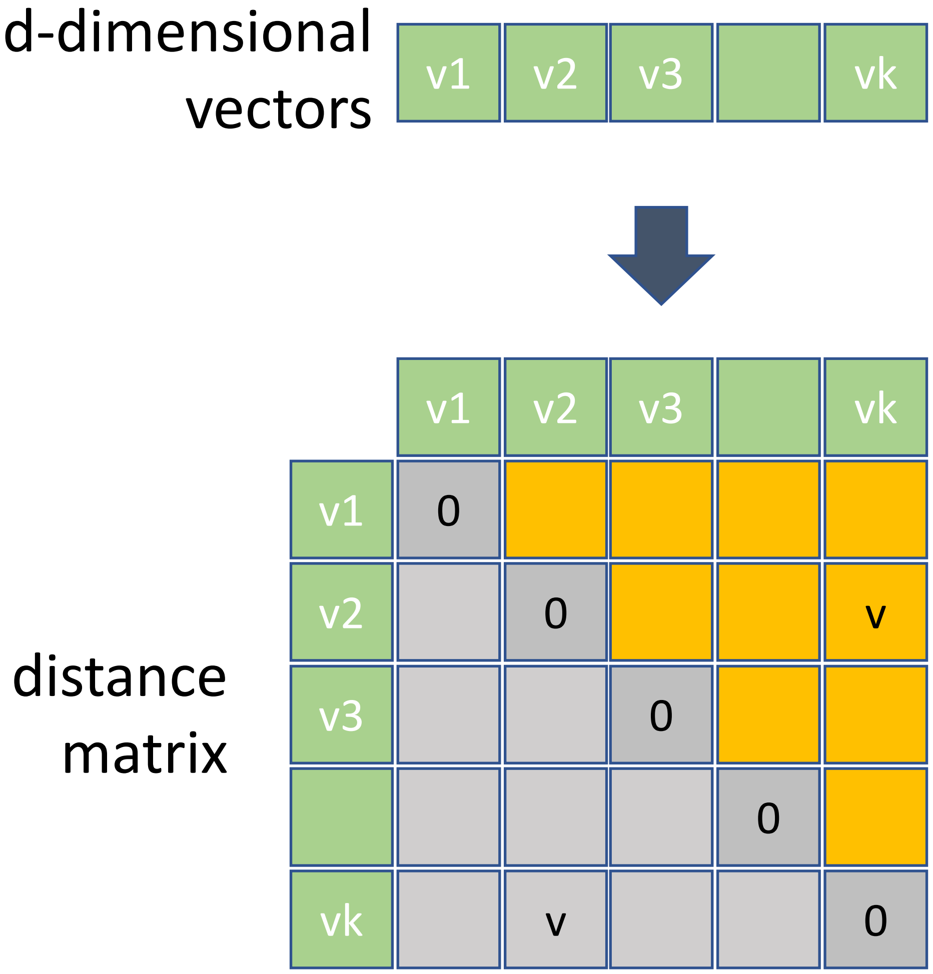
tasks

v1	v1	v1	v1	v2	v2	v2	v3	v3	
v2	v3		vk	v3		vk		vk	vk

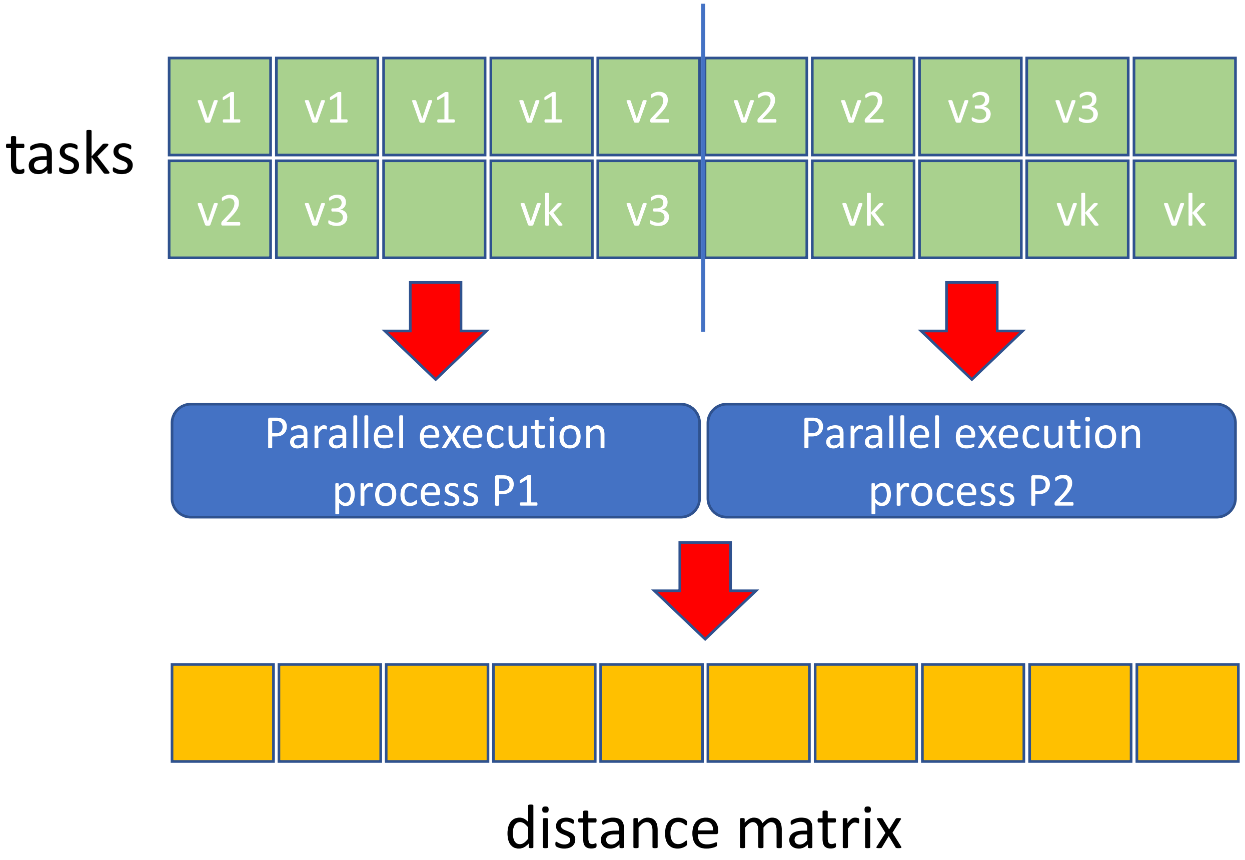
Sequential execution, process P1

distance matrix

Case 1 - computing a distance matrix, parallel scheme

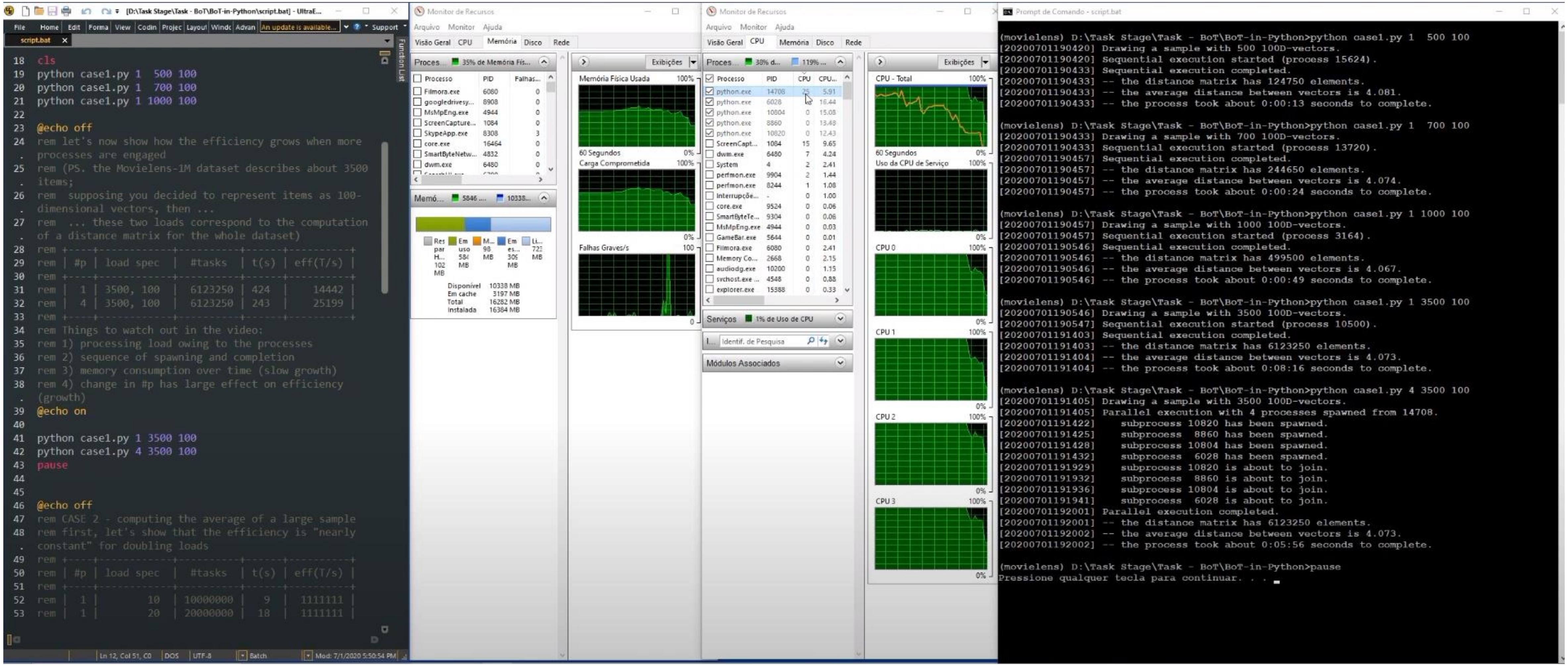


	v1	v2	v3		vk
v1	0				
v2		0			v
v3			0		
				0	
vk		v			0



v1	v1	v1	v1	v2	v2	v2	v3	v3	
v2	v3		vk	v3		vk		vk	vk

Case 1 - computing a distance matrix, sequential scheme



```
cls
python case1.py 1  500 100
python case1.py 1  700 100
python case1.py 1 1000 100

@echo off
rem let's now show how the efficiency grows when more processes are engaged
rem (PS. the Movielens-1M dataset describes about 3500 items;
rem  supposing you decided to represent items as 100-dimensional vectors, then ...
rem  ... these two loads correspond to the computation of a distance matrix for the whole dataset)
rem +----+-------------+-----------+------+-----------+
rem | #p | load spec   |  #tasks   | t(s) | eff(T/s)  |
rem +----+-------------+-----------+------+-----------+
rem |  1 | 3500, 100   |  6123250  | 424  |     14442 |
rem |  4 | 3500, 100   |  6123250  | 243  |     25199 |
rem +----+-------------+-----------+------+-----------+
rem Things to watch out in the video:
rem 1) processing load owing to the processes
rem 2) sequence of spawning and completion
rem 3) memory consumption over time (slow growth)
rem 4) change in #p has large effect on efficiency (growth)
@echo on

python case1.py 1 3500 100
python case1.py 4 3500 100
pause


@echo off
rem CASE 2 - computing the average of a large sample
rem first, let's show that the efficiency is "nearly constant" for doubling loads
rem +----+-------------+-----------+------+-----------+
rem | #p | load spec   |  #tasks   | t(s) | eff(T/s)  |
rem +----+-------------+-----------+------+-----------+
rem |  1 |          10 |  10000000 |    9 |   1111111 |
rem |  1 |          20 |  20000000 |   18 |   1111111 |
```

□ Processo	PID	Falhas...
□ Filmora.exe	6080	0
□ googledrivesy...	8908	0
□ MsMpEng.exe	4944	0
□ ScreenCapture...	1084	0
□ SkypeApp.exe	8308	3
□ core.exe	16464	0
□ SmartByteNetw...	4832	0
□ dwm.exe	6480	0

Memó... 5846 .... 10338...

🟫 Res par H... 102 MB | 🟩 Em uso 584 MB | 🟧 M... 98 MB | 🟦 Em es... 309 MB | ▢ Li... 723 MB

Disponível	10338 MB
Em cache	3197 MB
Total	16282 MB
Instalada	16384 MB

☑ Processo	PID	CPU	CPU...
☑ python.exe	14708	25	5.91
☑ python.exe	6028	[illegible]	16.44
☑ python.exe	10804	0	15.08
☑ python.exe	8860	0	13.48
☑ python.exe	10820	0	12.43
□ ScreenCapt...	1084	15	9.65
□ dwm.exe	6480	7	4.24
□ System	4	2	2.41
□ perfmon.exe	9904	2	1.44
□ perfmon.exe	8244	1	1.08
□ Interrupçõe...	-	0	1.00
□ core.exe	9524	0	0.06
□ SmartByteTe...	9304	0	0.06
□ MsMpEng.exe	4944	0	0.03
□ GameBar.exe	5644	0	0.01
□ Filmora.exe	6080	0	2.41
□ Memory Co...	2668	0	2.15
□ audiodg.exe	10200	0	1.15
□ svchost.exe ...	4548	0	0.88
□ explorer.exe	15388	0	0.33

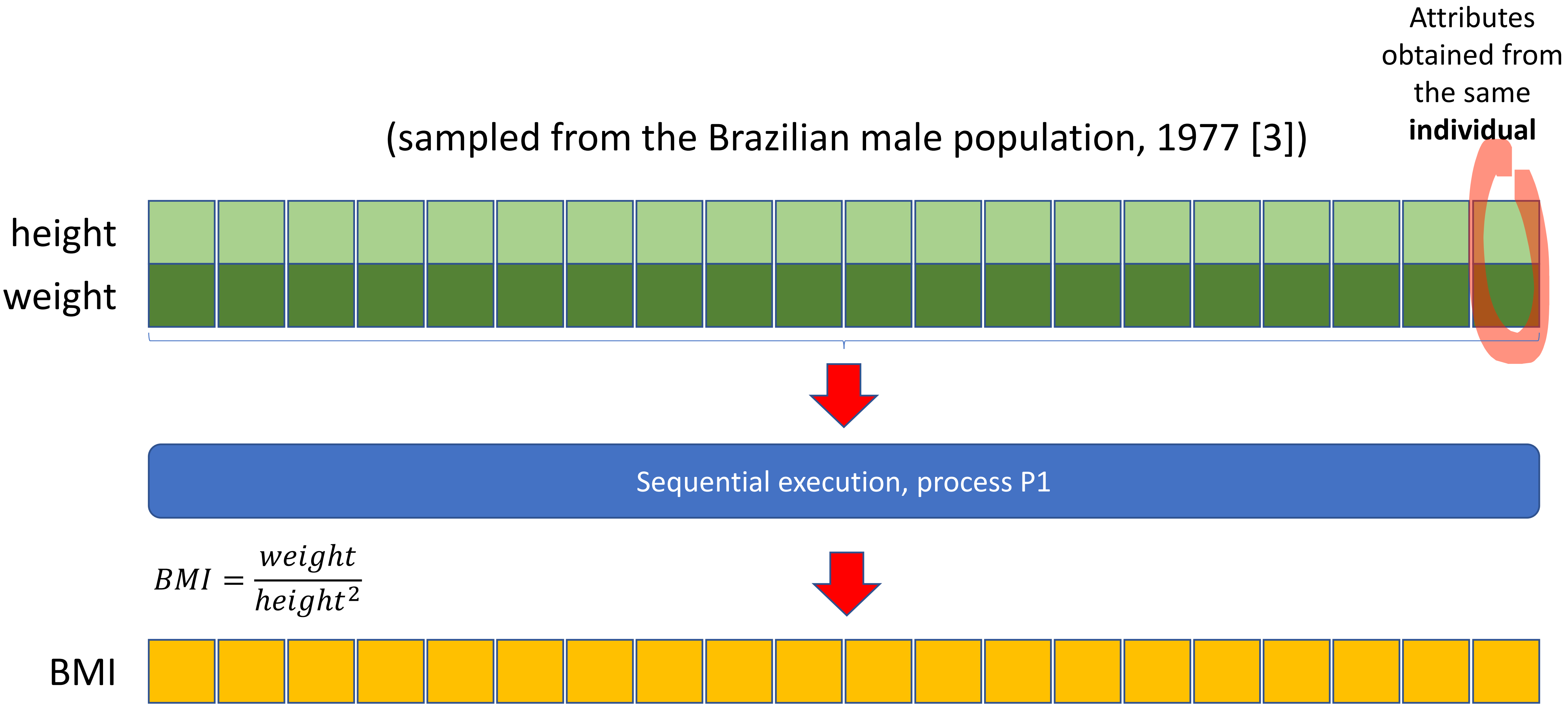
```
(movielens) D:\Task Stage\Task - BoT\BoT-in-Python>python case1.py 1  500 100
[20200701190420] Drawing a sample with 500 100D-vectors.
[20200701190420] Sequential execution started (process 15624).
[20200701190433] Sequential execution completed.
[20200701190433] -- the distance matrix has 124750 elements.
[20200701190433] -- the average distance between vectors is 4.081.
[20200701190433] -- the process took about 0:00:13 seconds to complete.

(movielens) D:\Task Stage\Task - BoT\BoT-in-Python>python case1.py 1  700 100
[20200701190433] Drawing a sample with 700 100D-vectors.
[20200701190433] Sequential execution started (process 13720).
[20200701190457] Sequential execution completed.
[20200701190457] -- the distance matrix has 244650 elements.
[20200701190457] -- the average distance between vectors is 4.074.
[20200701190457] -- the process took about 0:00:24 seconds to complete.

(movielens) D:\Task Stage\Task - BoT\BoT-in-Python>python case1.py 1 1000 100
[20200701190457] Drawing a sample with 1000 100D-vectors.
[20200701190457] Sequential execution started (process 3164).
[20200701190546] Sequential execution completed.
[20200701190546] -- the distance matrix has 499500 elements.
[20200701190546] -- the average distance between vectors is 4.067.
[20200701190546] -- the process took about 0:00:49 seconds to complete.

(movielens) D:\Task Stage\Task - BoT\BoT-in-Python>python case1.py 1 3500 100
[20200701190546] Drawing a sample with 3500 100D-vectors.
[20200701190547] Sequential execution started (process 10500).
[20200701191403] Sequential execution completed.
[20200701191403] -- the distance matrix has 6123250 elements.
[20200701191404] -- the average distance between vectors is 4.073.
[20200701191404] -- the process took about 0:08:16 seconds to complete.

(movielens) D:\Task Stage\Task - BoT\BoT-in-Python>python case1.py 4 3500 100
[20200701191405] Drawing a sample with 3500 100D-vectors.
[20200701191405] Parallel execution with 4 processes spawned from 14708.
[20200701191422]    subprocess 10820 has been spawned.
[20200701191425]    subprocess  8860 has been spawned.
[20200701191428]    subprocess 10804 has been spawned.
[20200701191432]    subprocess  6028 has been spawned.
[20200701191929]    subprocess 10820 is about to join.
[20200701191932]    subprocess  8860 is about to join.
[20200701191936]    subprocess 10804 is about to join.
[20200701191941]    subprocess  6028 is about to join.
[20200701192001] Parallel execution completed.
[20200701192001] -- the distance matrix has 6123250 elements.
[20200701192002] -- the average distance between vectors is 4.073.
[20200701192002] -- the process took about 0:05:56 seconds to complete.

(movielens) D:\Task Stage\Task - BoT\BoT-in-Python>pause
Pressione qualquer tecla para continuar. . .
```

Link para o vídeo

Case 2 – computing the average of a large sample

Case 2 - computing the average of a large sample, sequential scheme

(sampled from the Brazilian male population, 1977 [3])



Attributes obtained from the same **individual**

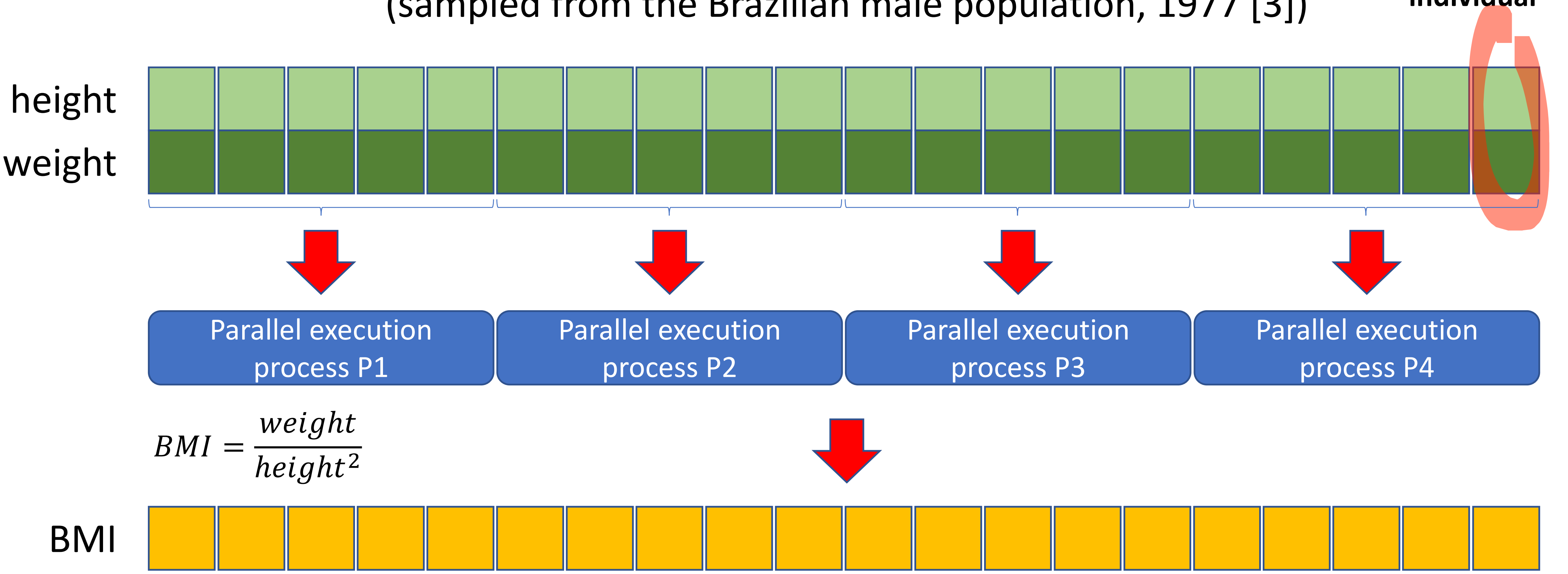
Sequential execution, process P1

$$BMI = \frac{weight}{height^2}$$

Case 2 - computing the average of a large sample, parallel scheme

(sampled from the Brazilian male population, 1977 [3])

Attributes obtained from the same **individual**



Parallel execution process P1

Parallel execution process P2

Parallel execution process P3

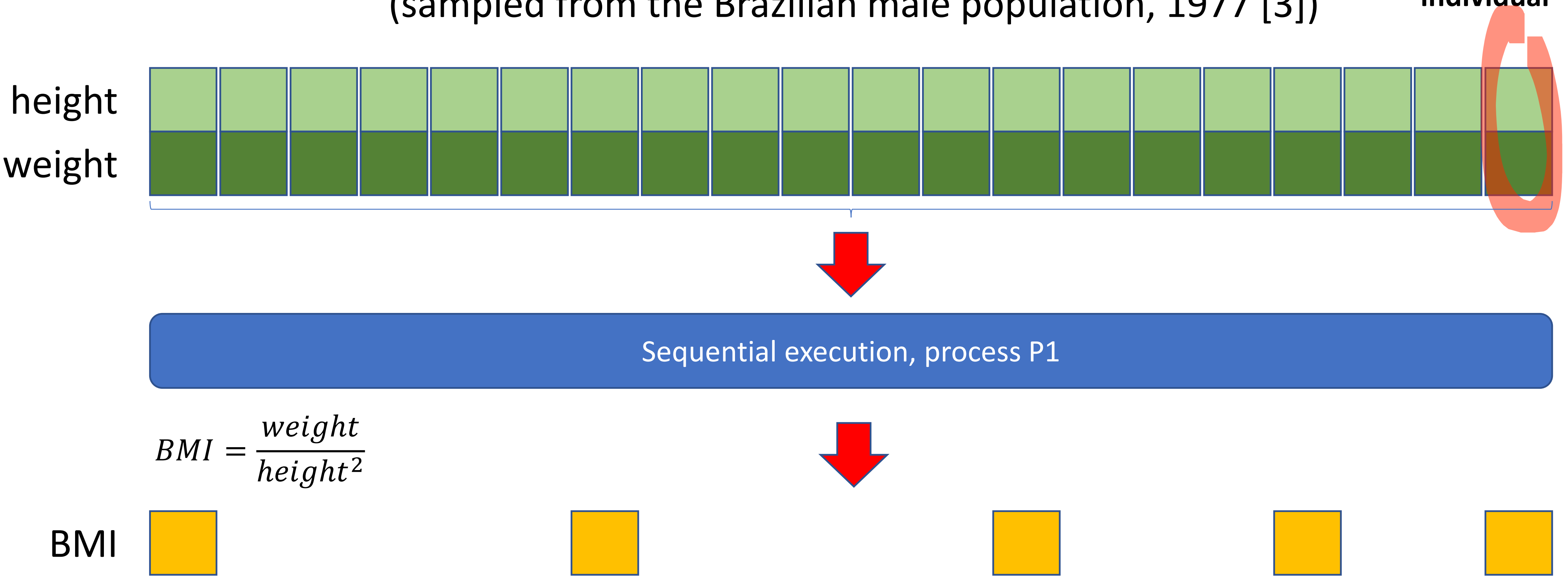
Parallel execution process P4

$$BMI = \frac{weight}{height^2}$$

Case 2 - computing the average of a large sample, statistical scheme

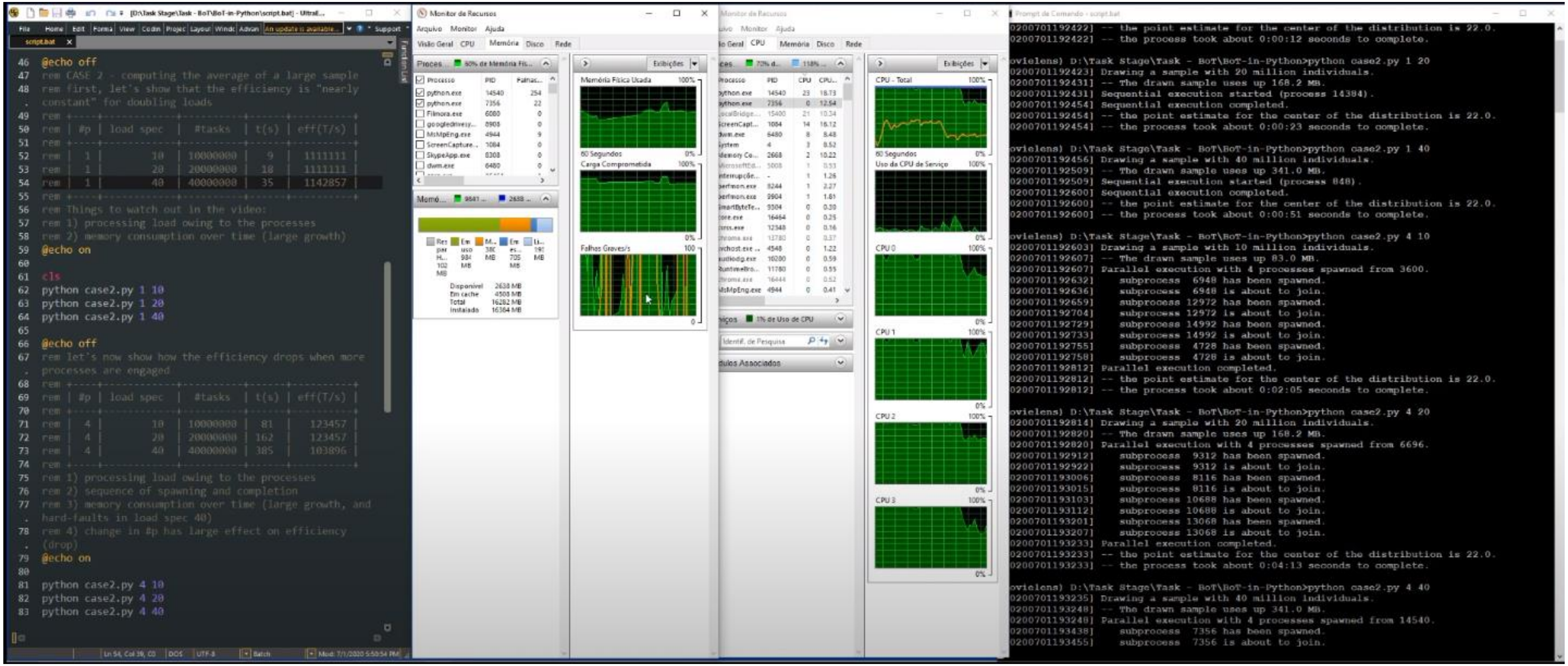
(sampled from the Brazilian male population, 1977 [3])

Attributes obtained from the same **individual**



$$BMI = \frac{weight}{height^2}$$

Case 2 - computing the average of a large sample



```
@echo off
rem CASE 2 - computing the average of a large sample
rem first, let's show that the efficiency is "nearly constant" for doubling loads
rem +----+-------------+-----------+------+-----------+
rem | #p | load spec   |  #tasks   | t(s) | eff(T/s)  |
rem +----+-------------+-----------+------+-----------+
rem |  1 |          10 |  10000000 |   9  |  1111111  |
rem |  1 |          20 |  20000000 |  18  |  1111111  |
rem |  1 |          40 |  40000000 |  35  |  1142857  |
rem +----+-------------+-----------+------+-----------+
rem Things to watch out in the video:
rem 1) processing load owing to the processes
rem 2) memory consumption over time (large growth)
@echo on

cls
python case2.py 1 10
python case2.py 1 20
python case2.py 1 40

@echo off
rem let's now show how the efficiency drops when more processes are engaged
rem +----+-------------+-----------+------+-----------+
rem | #p | load spec   |  #tasks   | t(s) | eff(T/s)  |
rem +----+-------------+-----------+------+-----------+
rem |  4 |          10 |  10000000 |  81  |   123457  |
rem |  4 |          20 |  20000000 | 162  |   123457  |
rem |  4 |          40 |  40000000 | 385  |   103896  |
rem +----+-------------+-----------+------+-----------+
rem 1) processing load owing to the processes
rem 2) sequence of spawning and completion
rem 3) memory consumption over time (large growth, and hard-faults in load spec 40)
rem 4) change in #p has large effect on efficiency (drop)
@echo on

python case2.py 4 10
python case2.py 4 20
python case2.py 4 40
```

Processo	PID	Falhas...
☑ python.exe	14540	254
☑ python.exe	7356	22
□ Filmora.exe	6080	0
□ googledrivesy...	8908	0
□ MsMpEng.exe	4944	9
□ ScreenCapture...	1084	0
□ SkypeApp.exe	8308	0
□ dwm.exe	6480	0

Disponível	2638 MB
Em cache	4508 MB
Total	16282 MB
Instalado	16364 MB

Processo	PID	CPU	CPU...
ython.exe	14540	23	18.73
ython.exe	7356	0	12.54
ocalBridge...	15400	21	10.34
ScreenCapt...	1084	14	16.12
dwm.exe	6480	8	8.48
System	4	3	8.52
Memory Co...	2668	2	10.22
MicrosoftEd...	5008	1	0.53
nterrupçõe...	-	1	1.26
perfmon.exe	8244	1	2.27
perfmon.exe	9904	1	1.81
SmartByteTe...	9304	0	0.30
core.exe	16464	0	0.25
csrss.exe	12348	0	0.16
chrome.exe	13780	0	0.37
svchost.exe ...	4548	0	1.22
audiodg.exe	10200	0	0.59
RuntimeBro...	11780	0	0.55
chrome.exe	16444	0	0.52
MsMpEng.exe	4944	0	0.41

```
0200701192422] -- the point estimate for the center of the distribution is 22.0.
0200701192422] -- the process took about 0:00:12 seconds to complete.

ovielens) D:\Task Stage\Task - BoT\BoT-in-Python>python case2.py 1 20
0200701192423] Drawing a sample with 20 million individuals.
0200701192431] -- The drawn sample uses up 168.2 MB.
0200701192431] Sequential execution started (process 14384).
0200701192454] Sequential execution completed.
0200701192454] -- the point estimate for the center of the distribution is 22.0.
0200701192454] -- the process took about 0:00:23 seconds to complete.

ovielens) D:\Task Stage\Task - BoT\BoT-in-Python>python case2.py 1 40
0200701192456] Drawing a sample with 40 million individuals.
0200701192509] -- The drawn sample uses up 341.0 MB.
0200701192509] Sequential execution started (process 848).
0200701192600] Sequential execution completed.
0200701192600] -- the point estimate for the center of the distribution is 22.0.
0200701192600] -- the process took about 0:00:51 seconds to complete.

ovielens) D:\Task Stage\Task - BoT\BoT-in-Python>python case2.py 4 10
0200701192603] Drawing a sample with 10 million individuals.
0200701192607] -- The drawn sample uses up 83.0 MB.
0200701192607] Parallel execution with 4 processes spawned from 3600.
0200701192632]     subprocess  6948 has been spawned.
0200701192636]     subprocess  6948 is about to join.
0200701192659]     subprocess 12972 has been spawned.
0200701192704]     subprocess 12972 is about to join.
0200701192729]     subprocess 14992 has been spawned.
0200701192733]     subprocess 14992 is about to join.
0200701192755]     subprocess  4728 has been spawned.
0200701192758]     subprocess  4728 is about to join.
0200701192812] Parallel execution completed.
0200701192812] -- the point estimate for the center of the distribution is 22.0.
0200701192812] -- the process took about 0:02:05 seconds to complete.

ovielens) D:\Task Stage\Task - BoT\BoT-in-Python>python case2.py 4 20
0200701192814] Drawing a sample with 20 million individuals.
0200701192820] -- The drawn sample uses up 168.2 MB.
0200701192820] Parallel execution with 4 processes spawned from 6696.
0200701192912]     subprocess  9312 has been spawned.
0200701192922]     subprocess  9312 is about to join.
0200701193006]     subprocess  8116 has been spawned.
0200701193015]     subprocess  8116 is about to join.
0200701193103]     subprocess 10688 has been spawned.
0200701193112]     subprocess 10688 is about to join.
0200701193201]     subprocess 13068 has been spawned.
0200701193207]     subprocess 13068 is about to join.
0200701193233] Parallel execution completed.
0200701193233] -- the point estimate for the center of the distribution is 22.0.
0200701193233] -- the process took about 0:04:13 seconds to complete.

ovielens) D:\Task Stage\Task - BoT\BoT-in-Python>python case2.py 4 40
0200701193235] Drawing a sample with 40 million individuals.
0200701193248] -- The drawn sample uses up 341.0 MB.
0200701193248] Parallel execution with 4 processes spawned from 14540.
0200701193438]     subprocess  7356 has been spawned.
0200701193455]     subprocess  7356 is about to join.
```

Link para o vídeo

Case 3 – TBD

To be resumed in a future meeting ;)

Parallel Programming in Python – a very, very short introduction

Thanks!

Parallel Programming in Python – a very, very short introduction

References:

[1] Quinn, Michael. "Parallel Programming in C with MPI and OpenMP", McGraw-Hill Science, ISBN 9780072822564, 2003.

[2] de Souza, Jaime Freire, Hermes Senger, and Fabricio AB Silva. "Escalabilidade de Aplicações Bag-of-Tasks em Plataformas Heterogêneas." In Anais Principais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, pp. 664-677. SBC, 2019.

[3] Guimaraes, M. I. C. C., & Sordi, G. M. A. A. (1995). *Desenvolvimento do manequim matemático do homem brasileiro para cálculos de dosimetria interna*. Tese de Doutorado. Instituto de Pesquisas Energéticas e Nucleares (IPEN) - Universidade de São Paulo, São Paulo.
# (see page 29; uses data that were collected by IBGE back in 1976/1977)