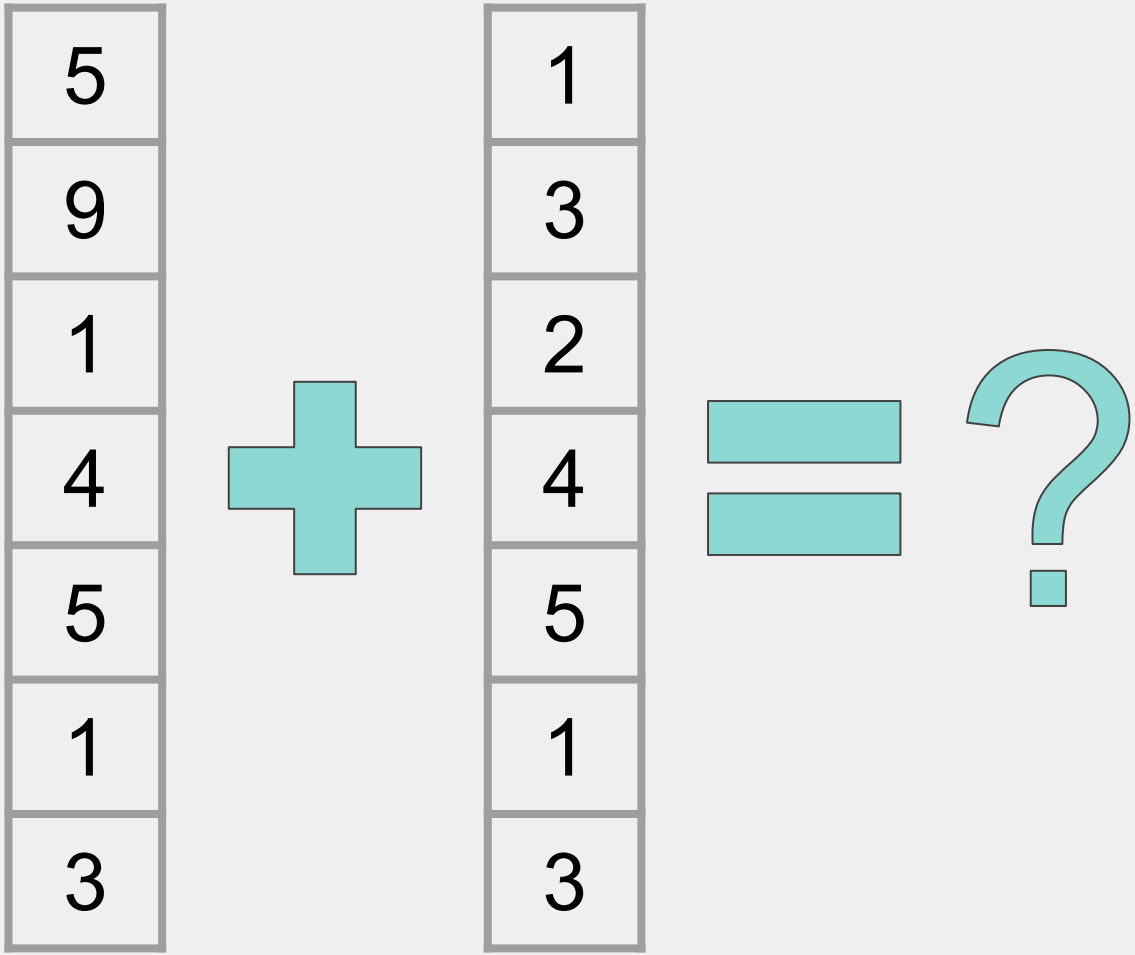
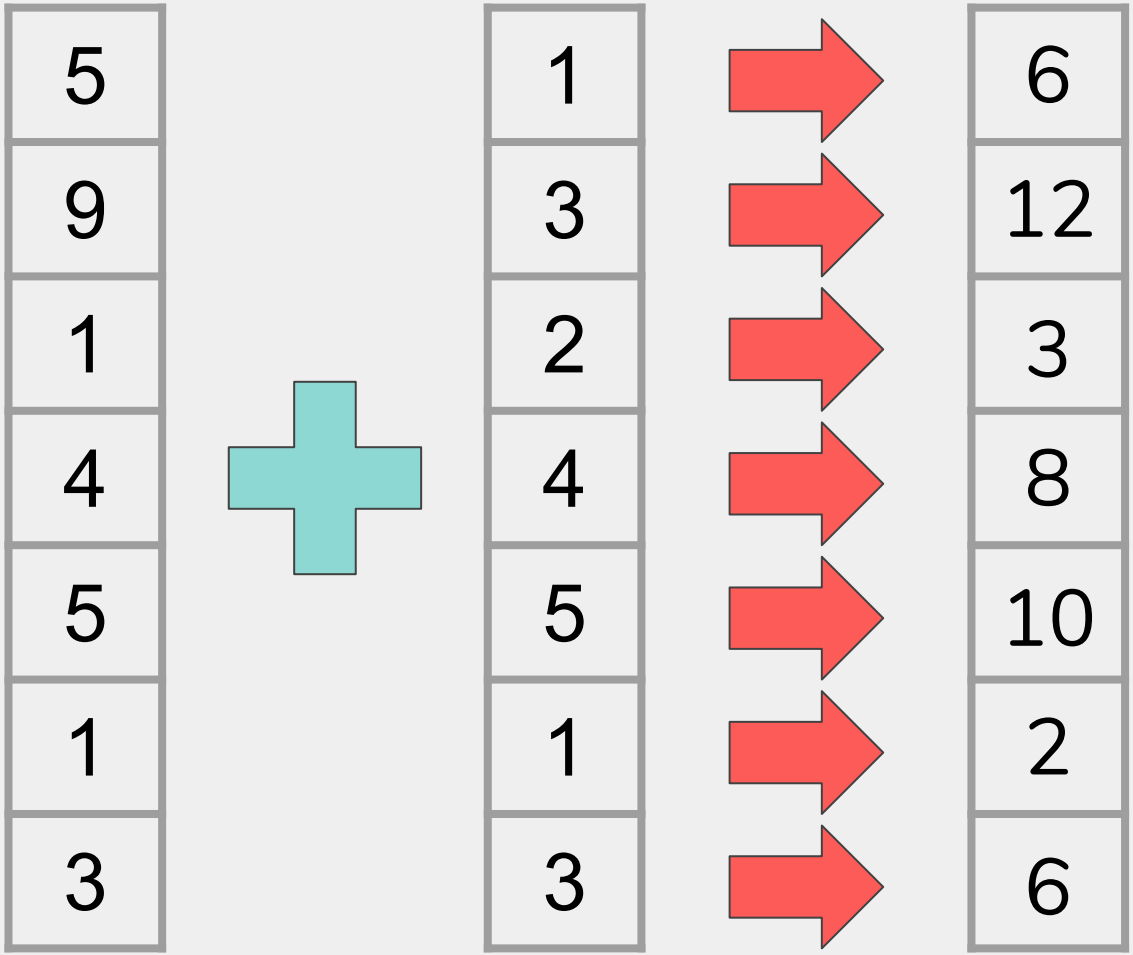
Vettorizzazione: a hands-on approach in Python

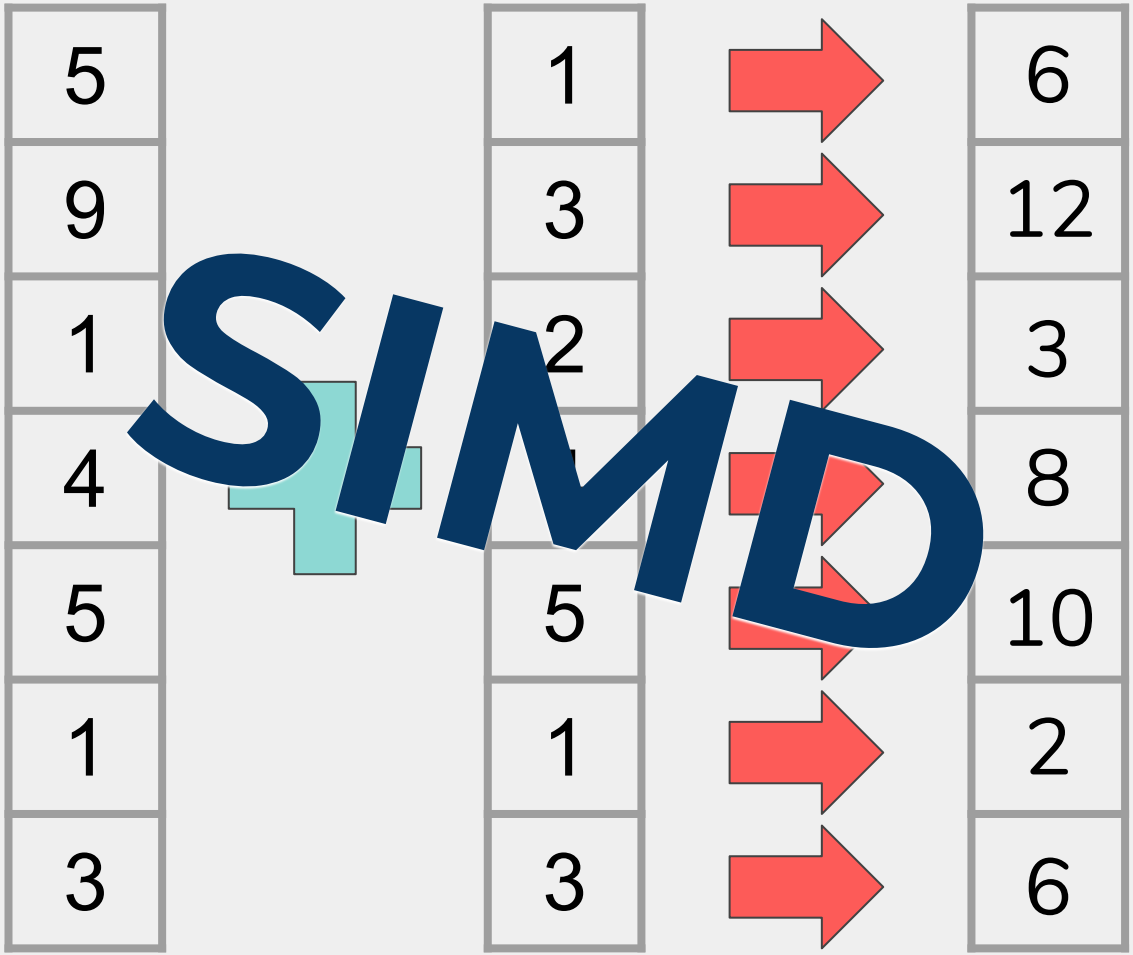
Davide Riva (driva95@protonmail.com)


5
9
1
4
5
1
3
+
1
3
2
4
5
1
3
=
?

Senza vettorizzazione

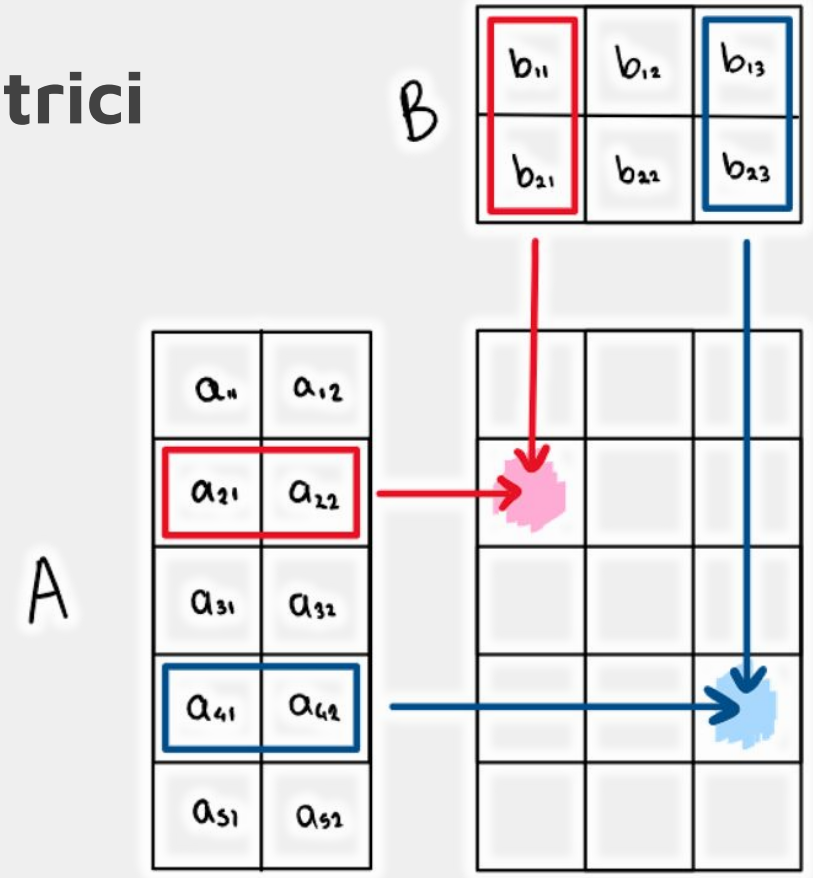
```
1. v1 = [5, 9, 1, 4, 5, 1, 3]
2. v2 = [1, 3, 2, 4, 5, 1, 3]
3.
4. vout = []
5. for i in range(len(v1)):
6.     vout.append(v1[i] + v2[i])
```


5
9
1
4
5
1
3
1
3
2
4
5
1
3
6
12
3
8
10
2
6

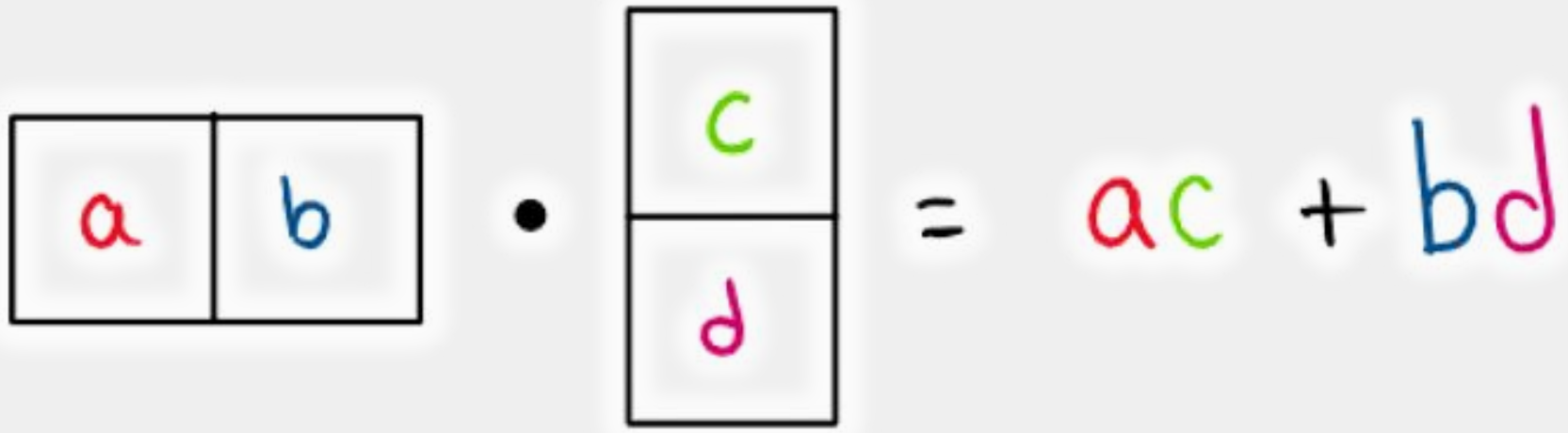

5
9
1
4
5
1
3
1
3
2
5
1
3
6
12
3
8
10
2
6
SIMD

Benchmark

Prodotto tra matrici
C = A B



Prodotto scalare



Senza vettorizzazione

```
1.  def my_matmul(A, B):
2.      assert A.shape[1] == B.shape[0]
3.      C = np.zeros((A.shape[0], B.shape[1]), dtype=np.int64)
4.
5.      for i in range(C.shape[0]):
6.          for j in range(C.shape[1]):
7.              col = B[:, j] # Colonna j-esima
8.              row = A[i, :] # Riga i-esima
9.              inner_out = 0
10.             for inner in range(len(col)):
11.                 inner_out = inner_out + (col[inner] * row[inner])
12.             C[i, j] = inner_out
13.     return C
```

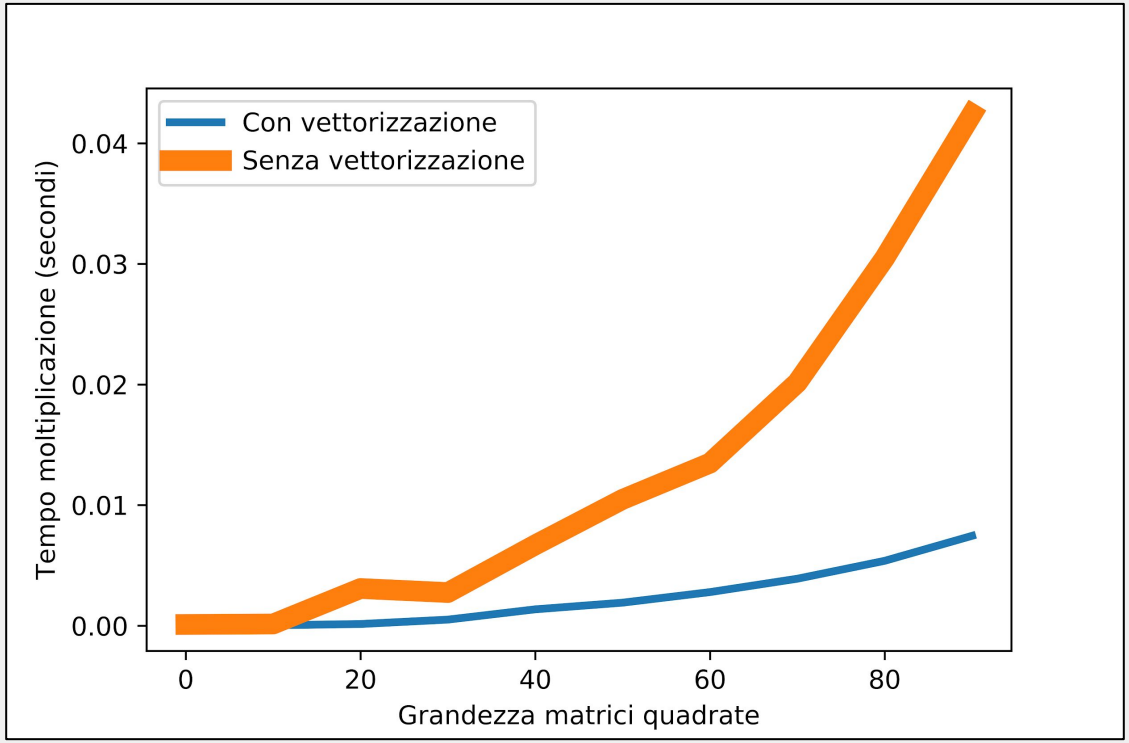
Senza vettorizzazione (Cython)

```
1.  def my_matmul_opt(long[:, ::1] A, long[::1, :] B):
2.      assert A.shape[1] == B.shape[0]
3.      cdef long[:, :] C = np.zeros((A.shape[0], B.shape[1]),
    dtype=np.int64)
4.
5.      cdef long inner_out = 0L
6.      for i in range(C.shape[0]):
7.          for j in range(C.shape[1]):
8.              inner_out = 0L
9.              for inner in range(A.shape[1]):
10.                 inner_out = inner_out + (B[inner, j] * A[i, inner])
11.             C[i, j] = inner_out
12.     return np.asarray(C)
```

Con vettorizzazione

```
C = np.matmul(A, B)
```

Confronto performance



Quali sono le differenze tra vettorizzazione e programmazione multithreading?

Intel Advanced Vector Extensions (AVX)

Insieme di istruzioni per effettuare operazioni SIMD su processori Intel/AMD

add / sub

max / min

average

div / mul

and / or / xor / ...

dot product

bit shift

compare

... e molto altro

Intel Advanced Vector Extensions (AVX)

- Registri fino a 512 bit (AVX-512)
- Operazioni non distruttive: A = B + C
- FMA: a = a + (b * c)
- CPUID per il supporto HW e dell'OS
- Intel rilascia un emulatore per il testing

Intel Xenon Phi VPU

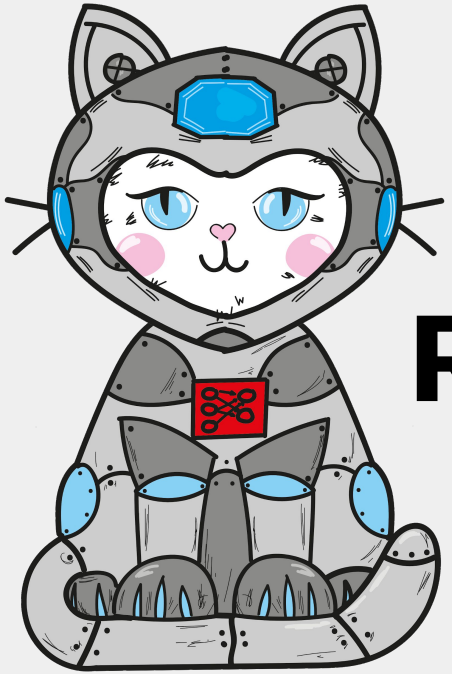
Solitamente:

- latenza: 4 cicli
- throughput: 1 ciclo

What’s next?

- Anaconda: https://www.anaconda.com/distribution/
- NumPy - Quickstart tutorial: https://numpy.org/devdocs/user/quickstart.html

Domande



Rage Against the Data

https://tiny.cc/RageAgainstTheData