MPI Performance in C and Python

Remington Breeze

Goal:

*"Compare 3 different MPI algorithms' performance in **C**/C++ vs the same algorithm in **Python** or Rust..."*

Chosen Algorithms

- Count primes up to a given number
- Compute average of randomized list of given size
- Calculate pi to a given accuracy

Relevant Articles

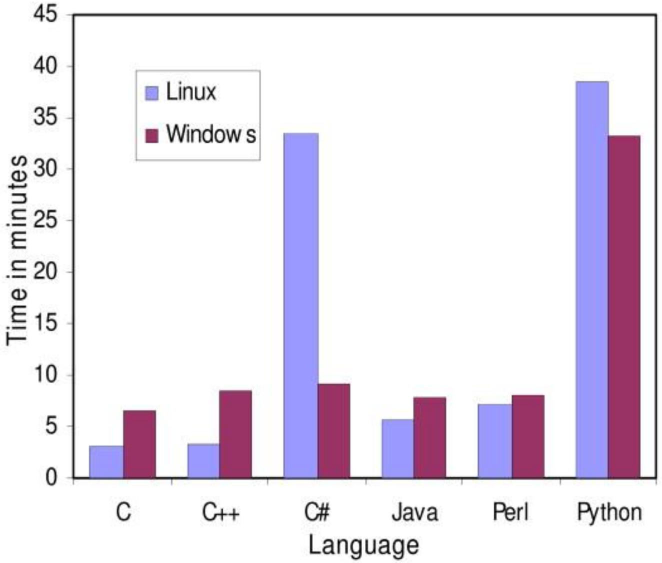
Performance of MPI Codes Written in Python with NumPy and mpi4py (IEEE)

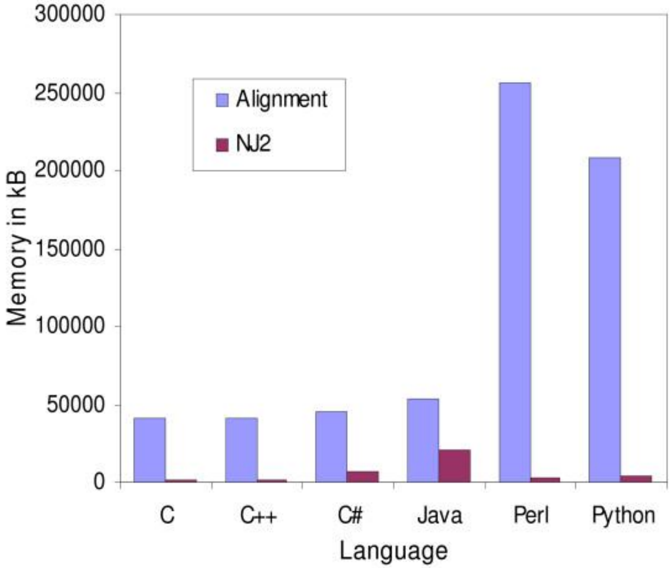
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7836843

A comparison of common programming languages used in bioinformatics

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2267699/

Non MPI performance comparison





On the performance of the Python programming language for serial and parallel scientific computations

http://heim.ifi.uio.no/~xingca/DR/Cai_Ref10.pdf

Performance of MPI Codes Written in Python with NumPy and mpi4py

https://www.dlr.de/sc/Portaldata/15/Resources/dokumente/pyhpc2016/slides/PyHPC_2016_talk_13.pdf

Introduction to MPI Python Programming

https://www.nesi.org.nz/sites/default/files/mpi-in-python.pdf

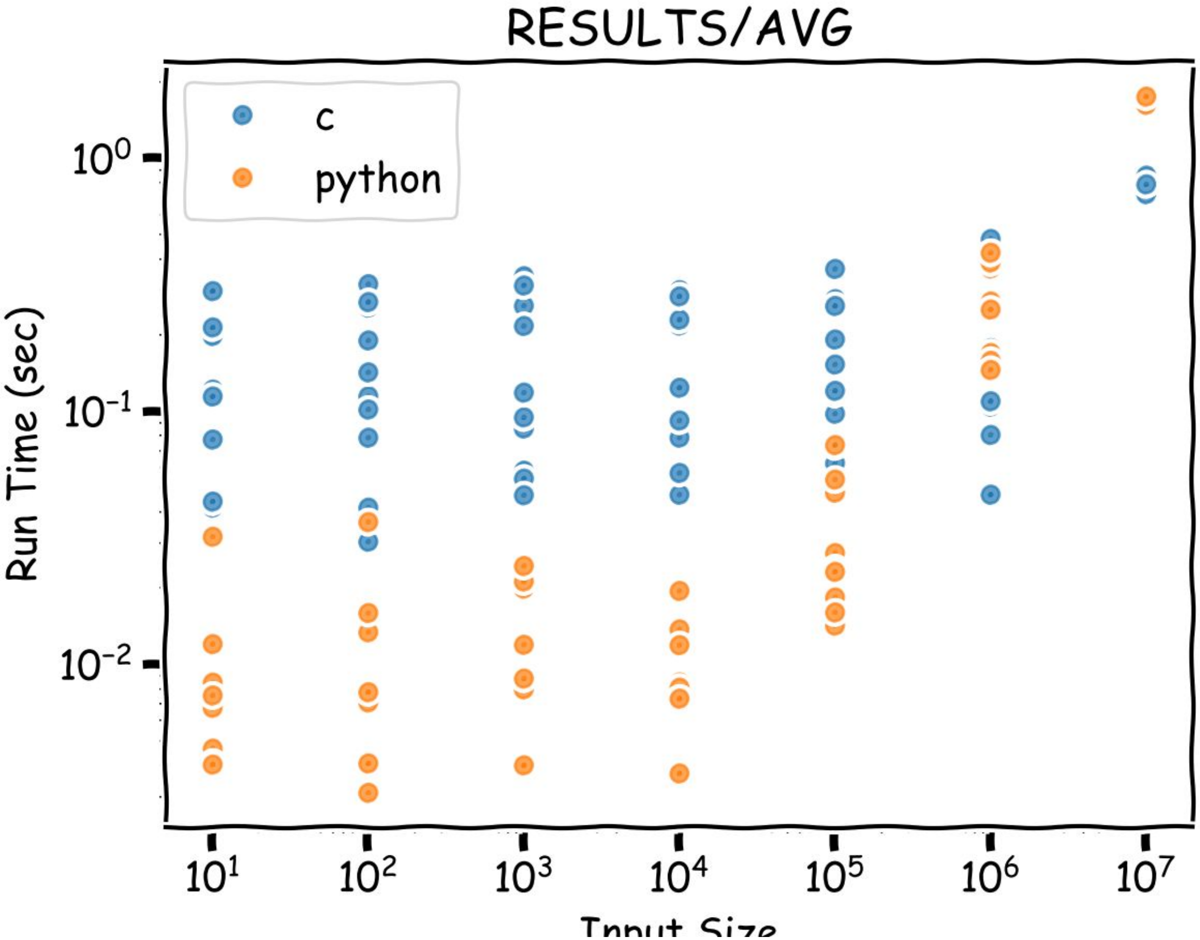
Results

As expected, Python ran significantly slower than C as input size increased.

However, for small input sizes, Python often ran faster than C.

Methodology

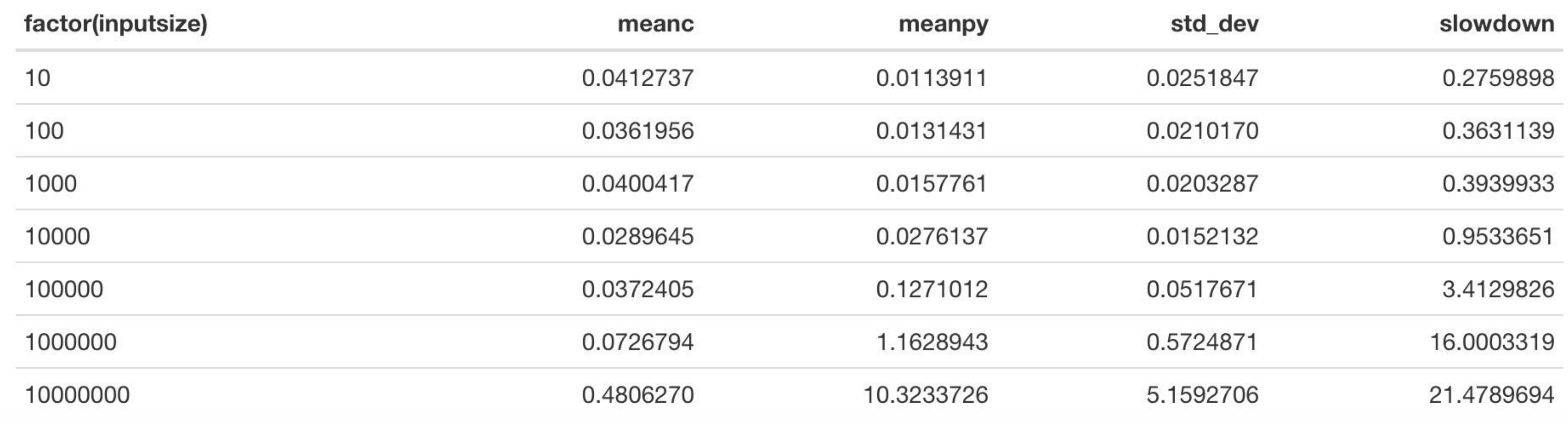
- Tested on OCNL 244 lab machines
- Used 2 machines, utilized 2 cores on each
- Used input sizes from 10 - 1000000
- Ran algorithms 5 times for each input size
- Used CPU timing , only for relevant code segments, for accuracy


RESULTS/AVG
c
python
Run Time (sec)
10^{0}
10^{-1}
10^{-2}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}
Input Size

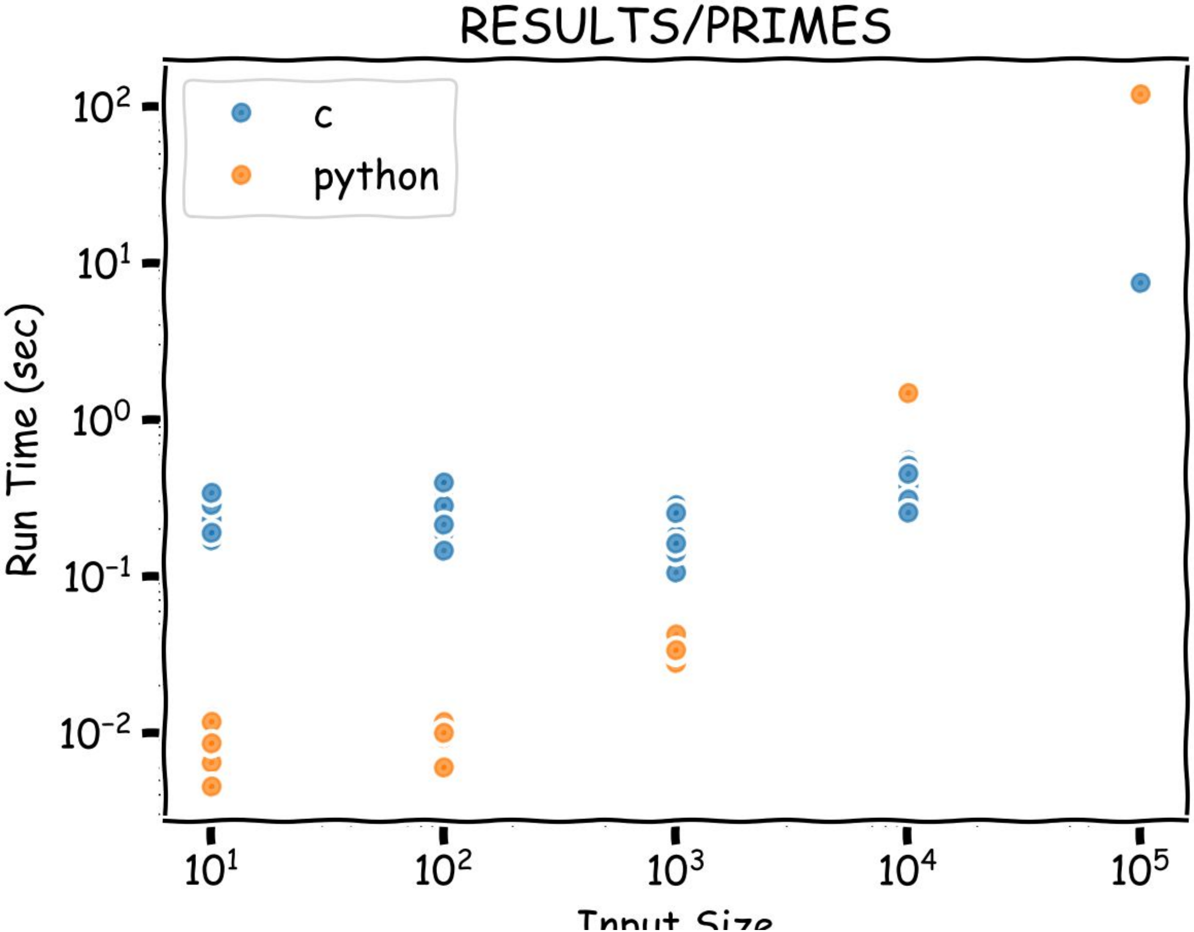
21x slowdown

For Python vs C for 10 million array items

Avg Algorithm



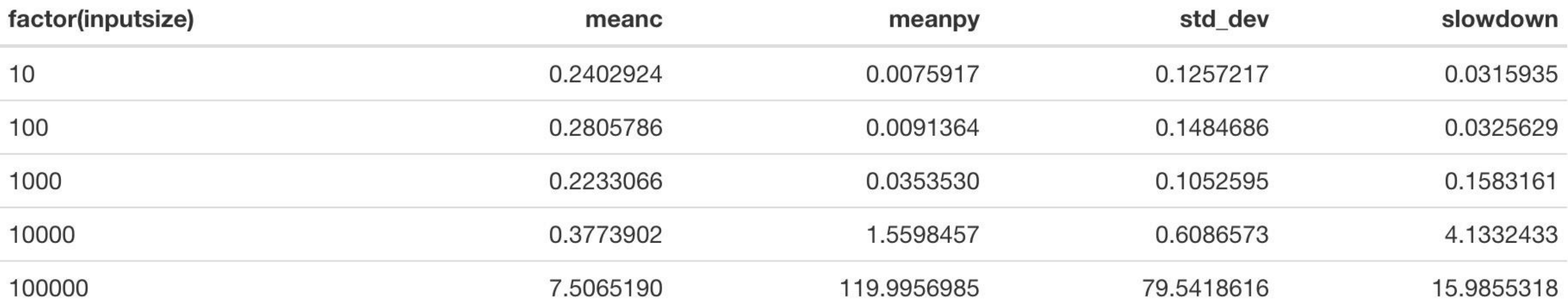
factor(inputsize)	meanc	meanpy	std_dev	slowdown
10	0.0412737	0.0113911	0.0251847	0.2759898
100	0.0361956	0.0131431	0.0210170	0.3631139
1000	0.0400417	0.0157761	0.0203287	0.3939933
10000	0.0289645	0.0276137	0.0152132	0.9533651
100000	0.0372405	0.1271012	0.0517671	3.4129826
1000000	0.0726794	1.1628943	0.5724871	16.0003319
10000000	0.4806270	10.3233726	5.1592706	21.4789694


RESULTS/PRIMES
Run Time (sec)
Input Size
c
python
10^2
10^1
10^0
10^{-1}
10^{-2}
10^1
10^2
10^3
10^4
10^5

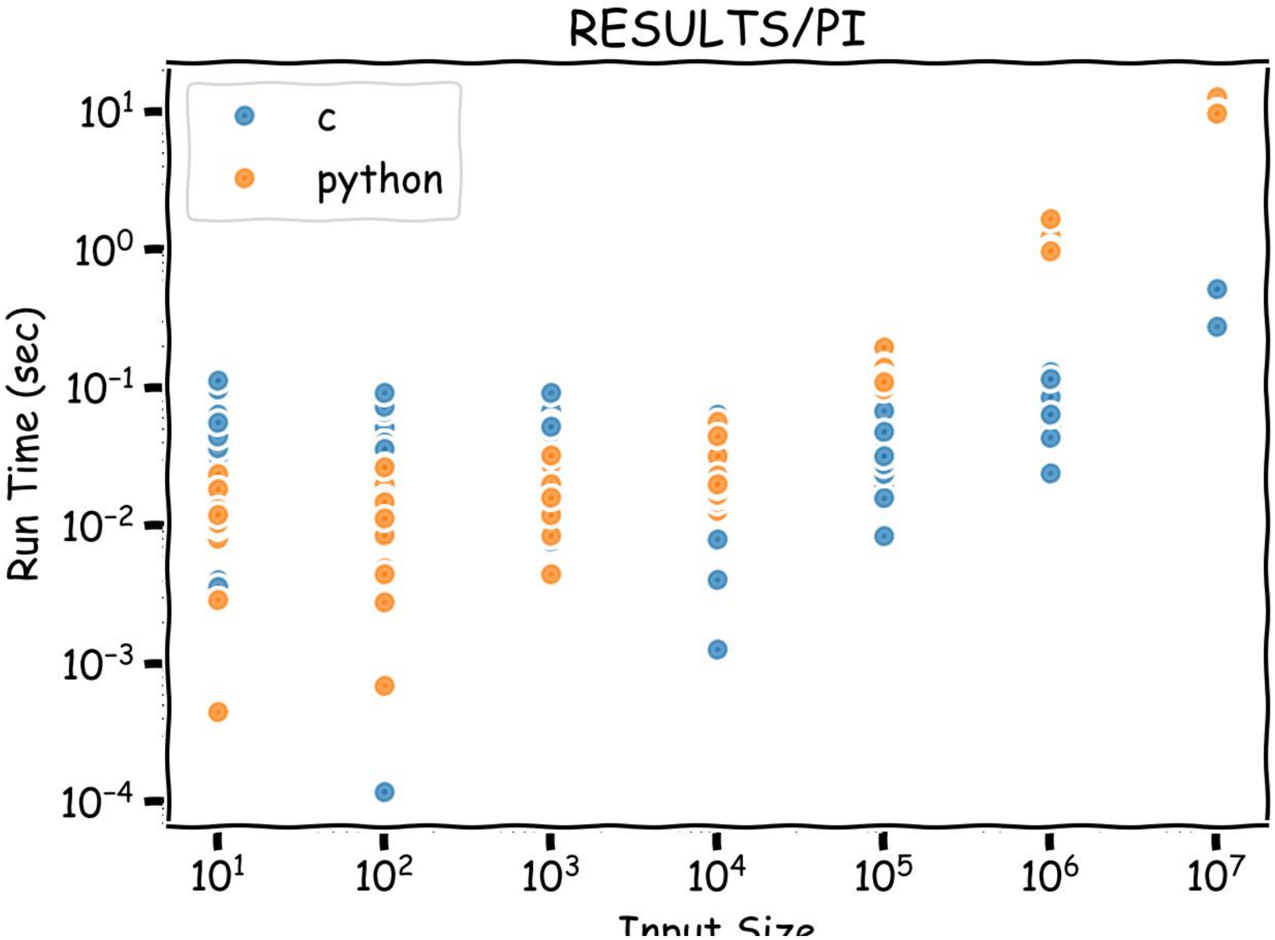
16x slowdown

For Python vs C for first 100,000 natural numbers

Primes Algorithm



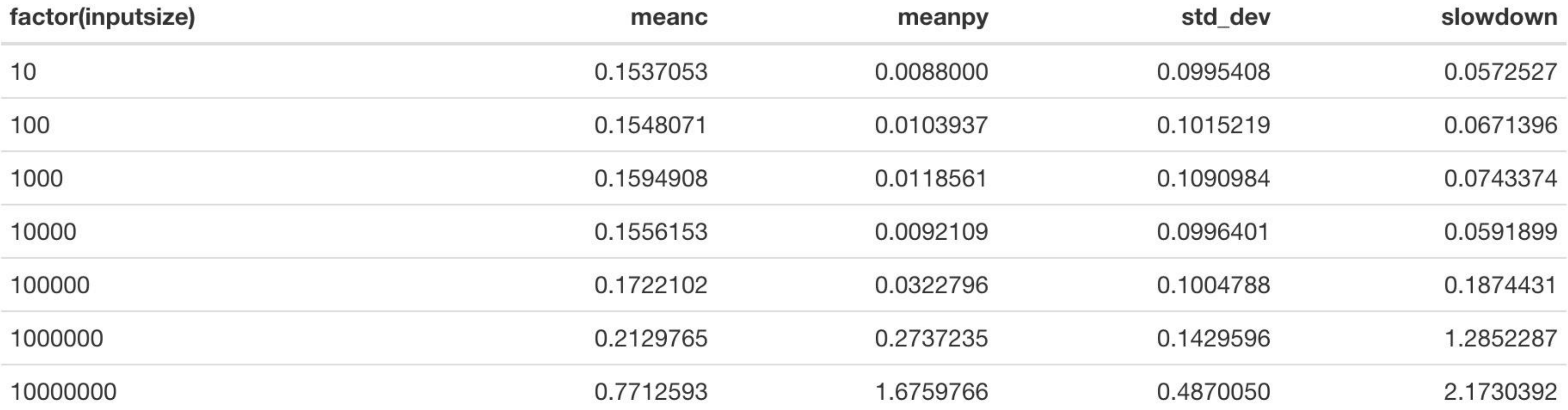
factor(inputsize)	meanc	meanpy	std_dev	slowdown
10	0.2402924	0.0075917	0.1257217	0.0315935
100	0.2805786	0.0091364	0.1484686	0.0325629
1000	0.2233066	0.0353530	0.1052595	0.1583161
10000	0.3773902	1.5598457	0.6086573	4.1332433
100000	7.5065190	119.9956985	79.5418616	15.9855318


RESULTS/PI
c
python
Run Time (sec)
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^1
10^2
10^3
10^4
10^5
10^6
10^7
Input Size

2x slowdown

For Python vs C for 10 million iterations

Pi Algorithm



factor(inputsize)	meanc	meanpy	std_dev	slowdown
10	0.1537053	0.0088000	0.0995408	0.0572527
100	0.1548071	0.0103937	0.1015219	0.0671396
1000	0.1594908	0.0118561	0.1090984	0.0743374
10000	0.1556153	0.0092109	0.0996401	0.0591899
100000	0.1722102	0.0322796	0.1004788	0.1874431
1000000	0.2129765	0.2737235	0.1429596	1.2852287
10000000	0.7712593	1.6759766	0.4870050	2.1730392