# Distributed Computing and Introduction to High Performance Computing

Ahmed Ratnani<sup>1</sup>

<sup>1</sup>Mohammed VI Polytechnic University, Benguerir, Morocco



A. Ratnani AL-KHWARIZMI 1/3

### Outline of this lecture

- Matrix multiplication example
- Profiling a Python code
- Accelerate a Python code
  - Using Numpy
  - Using Cython
  - Using Numba
  - Using Pyccel

A. Ratnani AL-KHWARIZMI 2/16

# Matrix multiplication example

See the following Jupyter Notebook on github:

**Advanced-Python-Motivations** 

A. Ratnani AL-KHWARIZMI 3/16

Why?

- It's mandatory to know what sections of code are bottlenecks in order to improve performance.
- You need to measure it, not to guess it
- Premature optimization is the root of all evil D. Knuth
- First make it work. Then make it right. Then make it fast. K. Beck
- How?

A. Ratnani AL-KHWARIZMI 4/16

### Different kinds of profilers

- Deterministic and statistical profiling
  - the profiler will be monitoring all the events
  - it will sample after time intervals to collect that information
- The level at which resources are measured; module, function or line level
- Profile visualizers

A. Ratnani AL-KHWARIZMI 5/16

#### Available tools

- Inbuilt timing modules
- profile and cProfile
- pstats
- line\_profiler
- Yappi
- vmprof-python
- pyinstrument
- gprof2dot
- pyprof2calltree, KCacheGrind
- snakeviz
- Scalene

A. Ratnani AL-KHWARIZMI 6/16

Use case

```
def linspace(start, stop, n):
      step = float(stop - start) / (n - 1)
      return [start + i * step for i in range(n)]
 5
    def mandel(c. maxiter):
      z = c
      for n in range(maxiter):
        if abs(z) ; 2:
           return n
10
        z = z*z + c
11
      return n
12
13
    def mandel_set(xmin=-2.0, xmax=0.5, ymin=-1.25, ymax=1.25,
14
                     width=1000. height=1000. maxiter=80):
15
      r = linspace(xmin, xmax, width)
16
      i = linspace(ymin, ymax, height)
17
      n = [[0]*width for _ in range(height)]
18
      for x in range(width):
19
         for y in range (height):
20
          n[y][x] = mandel(complex(r[x], i[y]), maxiter)
21
      return n
```

A. Ratnani AL-KHWARIZMI 7/1

timeit

2

The very naive way

```
import time

start_time = time.time()
mandel_set()
end_time = time.time()
# Time taken in seconds
elapsed_time = end_time - start_time

print('> Elapsed time', elapsed_time)
```

or using the magic method timeit

```
[In] %timeit mandel_set() [Out] 3.01 s +/- 84.6 ms per loop (mean +/- std. dev. of 7 runs, 1 loop each)
```

A. Ratnani AL-KHWARIZMI 8/16

prun

```
[In] %prun -s cumulative mandel_set()
```

which is, in console mode, equivalent to

```
python -m cProfile -s cumulative mandel.py
```

```
25214601 function calls in 5.151 seconds
 3
     Ordered by: cumulative time
 5
     ncalls tottime percall cumtime percall filename:lineno(function)
 6
         1 0.000 0.000 5.151 5.151 {built-in method builtins.exec}
 7
         1 0.002 0.002 5.151 5.151 <string>:1(<module>)
 8
         1000000 3.461 0.000 4.849 0.000 <ipython-input-4-9421bc2016cb>:5(mandel)
10
   24214592 1.388 0.000 1.388 0.000 {built-in method builtins.abs}
11
         1 0.008 0.008 0.008 0.008 cipython-input-4-9421bc2016cb>:17(<1istcomp>)
12
         2 0.000 0.000 0.000 0.000 <ipython-input-4-9421bc2016cb>:1(linspace)
13
         2 0.000 0.000 0.000 0.000 <ipython-input-4-9421bc2016cb>:3(<listcomp>)
         1 0.000 0.000 0.000 0.000 {method 'disable' of '_lsprof.Profiler' objects}
14
```

- Most of the time is spent in the mandel function
- profiling introduces some overhead 5.14(s) instead of 3.01(s)

A. Ratnani AL-KHWARIZMI

9/16

Style: Icicle

#### Visualization

- Profiling results can be visualized with SnakeViz
- We must be in console mode

```
1 python3 -m cProfile -o mandel.prof mandel.py
2 snakeviz --port 6542 --hostname localhost --server mandel.prof
```



A. Ratnani AL-KHWARIZMI 10/16

#### Details at the line level

- We know that most of the time is spent in the mandel function
- We shall use the line\_profiler package on this function to get details at the line level

```
[In] %load_ext line_profiler
[In] %lprun -f mandel mandel_set()
```

### Which gives the result

1

```
Timer unit: 1e-06 s
```

Total time: 22.8401 s

File: <ipython-input-2-9421bc2016cb>

Function: mandel at line 5

#Line	Hits	Time	Per Hit	% Time	Line Contents
5					def mandel(c, maxiter):
6	1000000	250304.0	0.3	1.1	z = c
7	24463110	6337732.0	0.3	27.7	<pre>for n in range(maxiter):</pre>
8	24214592	8327289.0	0.3	36.5	if $abs(z) > 2$ :
9	751482	201108.0	0.3	0.9	return n
10	23463110	7658255.0	0.3	33.5	z = z*z + c

A. Ratnani AL-KHWARIZMI 11/16

Details at the line level

This can be done in console mode as well

Then on the command line

```
1 kernprof -1 -v mandel.py
```

A. Ratnani AL-KHWARIZMI 12/16

### Numpy

#### Default BLAS - LAPACK

- 1 Dotted two 4096x4096 matrices in 64 22 s
- 2 Dotted two vectors of length 524288 in 0.80 ms.
- 3 SVD of a 2048x1024 matrix in 10.31 s.
- 4 Cholesky decomposition of a 2048x2048 matrix in 6.74 s.
- 5 Eigendecomposition of a 2048x2048 matrix in 53.77 s.

#### **ATLAS**

- 1 Dotted two 4096x4096 matrices in 3.46 s.
- 2 Dotted two vectors of length 524288 in 0.73 ms.
- 3 SVD of a 2048x1024 matrix in 2.02 s.
- 4 Cholesky decomposition of a 2048x2048 matrix in 0.51 s.
- 5 Eigendecomposition of a 2048x2048 matrix in 29.90 s.

A. Ratnani AL-KHWARIZMI 13/16

### Numpy

#### Intel MKL

- 1 Dotted two 4096x4096 matrices in 2.44 s.
- 2 Dotted two vectors of length 524288 in 0.75 ms.
- 3 SVD of a 2048x1024 matrix in 1.34 s.
- 4 Cholesky decomposition of a 2048x2048 matrix in 0.40 s.
- 5 Eigendecomposition of a 2048x2048 matrix in 10.07 s.

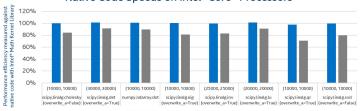
#### OpenBLAS

- 1 Dotted two 4096x4096 matrices in 3.97 s.
- 2 Dotted two vectors of length 524288 in 0.74 ms.
- 3 SVD of a 2048x1024 matrix in 1.96 s.
- 4 Cholesky decomposition of a 2048x2048 matrix in 0.46 s.
- 5 Eigendecomposition of a 2048x2048 matrix in 32.95 s.

A. Ratnani AL-KHWARIZMI 14/16

Numpy using MKL

### Intel Optimizations Improve Python\* Linear Algebra Efficiency Closer to Native Code Speeds on Intel® Core™ Processors



■ Intel® Distribution for Python\* 2020
■ Stock Python

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer. Performance results are based on testing as of November 27, 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchase, including the performance of that product when combined with other products. For more complete information sets yearing the Computer Changes.

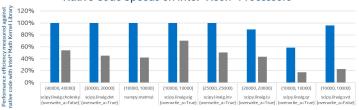
Configuration Testing by Intel as of Newmber 27, 2015. Stock Python python 3.75 NO37 1580.0 Intellide from conds, numpy 1,174, numbe 0.466.1, Birelline 0.200, stopy 1.32, scilid-larm 0.213 installated from pix, intelligent perfect of positions for python 2.000 disc) upthon 3.44 HAR448.2 minuty 1.137 a 9773-86848.9 Minuty 1.137 a 9773-86849.9 Minuty 1.137 a 9773-86

Intel's compliers may or may not optimize to the same degree for non-held incorporcessors for optimizations that are not unique to Intel microporcessors. These optimizations include SSE2, SSE2, and SSSE2 instruction sets and other optimizations in the long or magnitude the availability, furthermost end any optimization on microporcessors. On microporcessors department instruction is state and other optimizations and the size and patients of a positive production of microporcessors. On microporcessors department of the product is product as intellected for use with first indirect patients of the product in the microporcessors. Please refer to the applicable product User and Reference ducked for more information regarded the second instruction sets covered by this notice, Reference presents (SCI) 1000-000.

A. Ratnani AL-KHWARIZMI 15/16

Numpy using MKL

### Intel Optimizations Improve Python\* Linear Algebra Efficiency Closer to Native Code Speeds on Intel® Xeon® Processors



■ Intel® Distribution for Python\* 2020
■ Stock Python

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer. Performance results are based on testing as of Movember 27, 2019 and may not reflect all guidely availables equiry updates. See configuration disclosure for details. No product can be absolutely secure.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to wary. You should consult other information and performance tests to assist you in fully evaluating your contemplated profress, including the performance of that product when combined with other products. For more completed profression to vite weight combined to with the performance of that product when combined with other products. For more completed most not vite weighted to only the profression of the performance of that product when combined with other products. For more completed profression to vite weighted to the profression of the performance of the profression of the performance of the

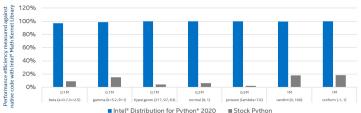
Configuration: Testing by Intel as of Herentee 27, 2016. Socio: Pythors python 3.5 h037 1600. Drinkled from moreia, mumy 1.17.4 norths 0.660, limite 0.000, stay 1.32, social-tesm 0.233 intellation may intelligent the process of the production for Pythors. 2000 data python 3.4 h44644.83, among 1.17.3 pythosidatis 10, may be and 2000 intell 1.30, and 1.01.01 intelligent 2000, and 1.01.01 pythosidatis 10, may be an extra 1.00 pythosidatis 10, may be

Intal's compliers may or may not optimize to the same degree for non-leaf intergopersons for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instructions sets and other optimizations in this does not parametre he availability, instructions set and other optimizations in indicated by Intel. Management of the product are intended for use with their incroprocessors. Certain optimizations in this product are intended for use with their incroprocessors. Certain optimizations and specific to Intel microprocessors. Persistent in the product are intended for use with their incroprocessors. Certain optimizations and specific to Intel microprocessors. Persistent in the product are intended for use with their incroprocessors. Persistent in the incroprocessors. Persistent in the incroprocessors. Persistent in the increprocessors. Persistent in the increprocessors.

A. Ratnani AL-KHWARIZMI 15/16

Numpy using MKL

# Intel Optimizations Improve Python\* Random Number Generation Efficiency Closer to Native Code Speeds on Intel® Core™ Processors



Intel technologies features and benefits depend on system configuration and may require enabled hardware software or service activation. Learn more at intel.com, or from the OEM or retailer. Performance regulat are based on testing as of Nevember 27, 2019 and may not reflect all inabilities wealists. See configuration disclosure for details. No product can be absolutely secure

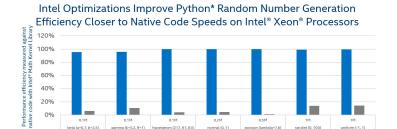
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any shange to any of those factors may cause the results to vay; You should consult other information and performance tests to assist you in fully evaluating your contemplated purchase, including the performance of that product, when combined with other products. To more complete information wish <u>www.mirel.complex.marks</u>.

Configuration: Testing by Intel as of Movember 22, 2019. Social Policies pathon 3.75 60.27 1650, 0 installated from conds, namey 1.174, rumbs 0.46.01, Breitle 1, 2013. Social Policies (1.2. social Policies Intel Poli

Intal is complient may or may not optimize to the same degree for non-Intal increposessors for optimizations that are not unique to Intel microprocessors. These optimizations include SEE, SEEL, and SSEEL instructions sets and other optimizations in the limit does not apparent the evaluability. Instructions, or effectiveness of any optimization on instruptoposessors instruction sets and instruction sets and instruction sets and instructions of the set of the product are intended for use with Intel microprocessors, certain optimizations in this product are intended for use with Intel microprocessors. Certain optimizations in optimization in the product are intended for use with Intel microprocessors. Please refer to the applicable product. User and Reference decides for more information regarding the specific instruction sets overed by this notice. Equipment seems 2011/10(4).

A. Ratnani AL-KHWARIZMI 15/16

Numpy using MKL



■ Intel® Distribution for Python\* 2020 Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation, Learn more at intel.com, or from the OEM or retailer, Performance results are based on testing as of November 27, 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary, You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

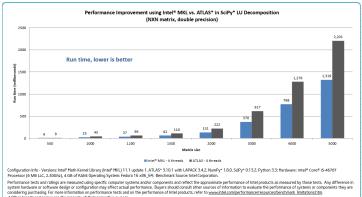
Configuration: Testing by Intel as of November 27, 2019. Stock Python: python 3.7.5 h0371630 0 installed from conda, numpy 1.17.4, numba 0.46.0, llymlite 0.30.0, scipy 1.3.2, scikit-learn 0.21.3 installed from pip; Configurations: letting by Initia si of November 27, 2019. Stock 1900or 19thon 1.25 00371850; 0 Installment moni crossa, numpy 1.17.4, number 0.4640, Immilities 2019, 2

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent User and Reference Guides for more information reparding the specific instruction sets covered by this notice. Notice revision #20110804

> A. Ratnani AL-KHWARIZMI 15/16

■ Stock Python

### Numpy using MKL

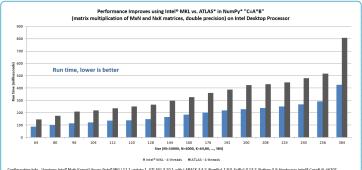


\* Other brands and names are the property of their respective owners

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations, Intel does not quarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

> A. Ratnani AI-KHWARIZMI 16/16

### Numpy using MKL



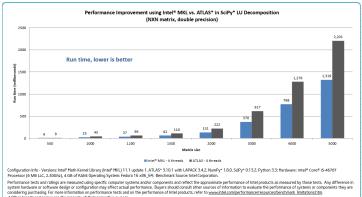
Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.1 update 1, ATLAS® 3.10.1 with LAPACK 3.4.2, NumPy® 1.8.0, SciPy® 0.13.2, Python 3.3; Hardware: Intel® Core® 15-4670T Processor (6 MB LLC, 2.30Ghz), 4 GB of RANt; Operating System: Fedora 16 x86\_64; Benchmark Source: Intel Corporation.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference is system hardware or software design or contiguent one agrid est, under performance. Being should consult offer sources of information to evaluate the performance tests and on the performance of Intel products, refler to <u>wow.lintel.com/performance/persources/persour</u>

Optimization Notice: Intel's compliers may or may not optimize to the same degree for non-intel microprocessors for optimizations that are not unique to Intel microprocessors. The optimizations that are not unique to Intel microprocessors from post intelligence of the optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with mise of the optimization optimization of the optimization optimization of the optimization optim

A. Ratnani AL-KHWARIZMI 16/16

### Numpy using MKL



\* Other brands and names are the property of their respective owners

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations, Intel does not quarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

> A. Ratnani AI-KHWARIZMI 16/16