Python wrapper performances

Jean Jacquemier, Pierre Aubert, Thomas Vuillaume, Gilles Maurin

2nd ASTERICS-OBELICS Workshop

16-19 October 2017, Barcelona, Spain.

H2020-Astronomy ESFRI and Research Infrastructure Cluster
(Grant Agreement number: 653477).

This project has received funding from the European Union's Horizon 2020 research and innovation programme
under grant agreement No 653477

HPC techniques for big data processing

- CTA telescopes ->TeraByte of data/night

-> high-performance computing techniques

  - Data format generator
  - CPU Data prefectching, Vectorization, Contiguous data, Cache Friendly.

  - HPC algorithms on Intel CPU
    - Vectorization (SIMD)
    - Loop optimization

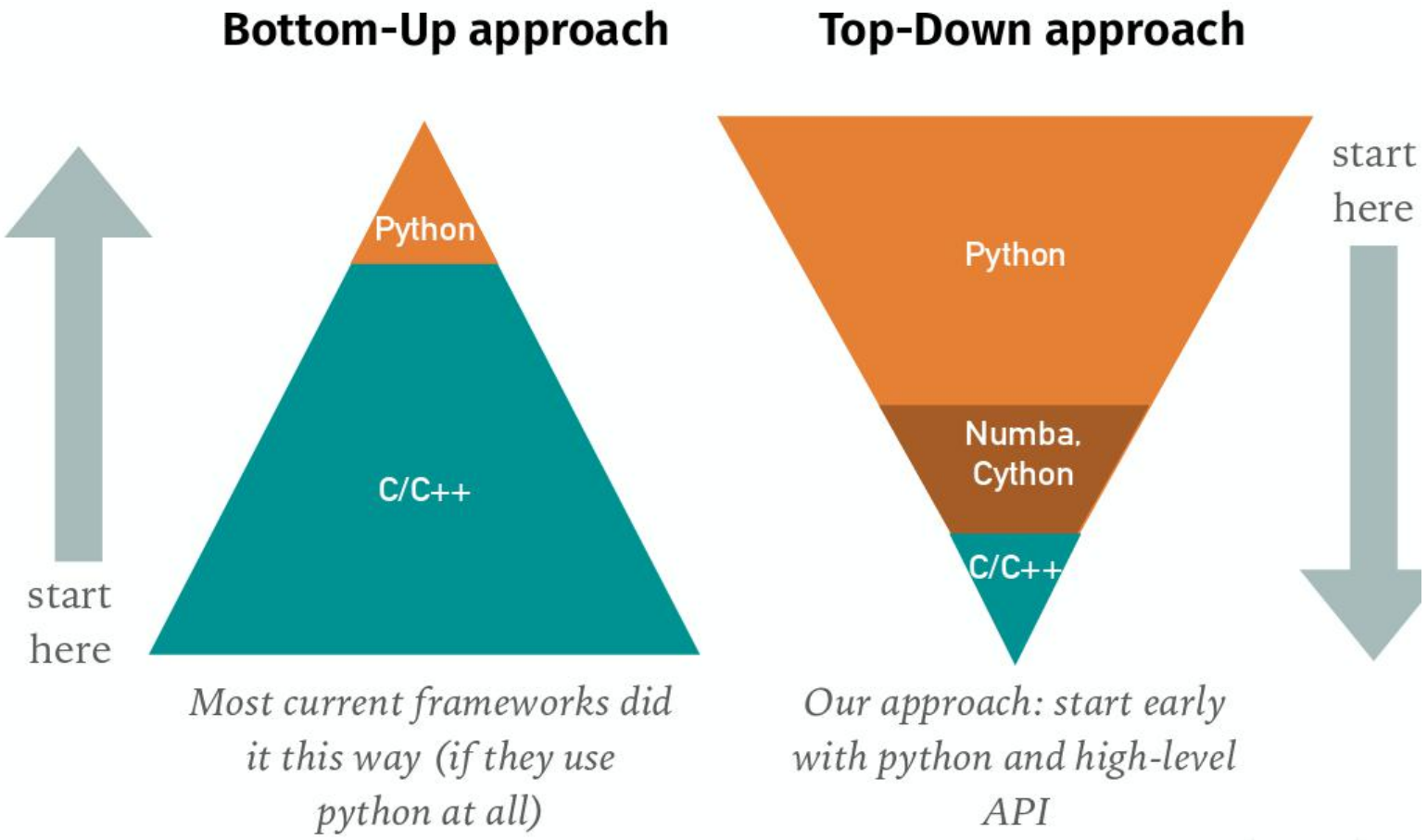
Python for astroparticle physics

- The software langage of astroparticle physics.
  - Mainly use for analysis.

- CTA pipeline in Python
  - performances is still challenging

- C++ HPC algorithms and data format to Python library.

ctapipe framework approach



Python wrapper performances

- We studied different ways on wrap C++ code to Python.
  - ctypes, swig, pybind11, Python/C_API

- Feedback about experiences

Data format Performances

- Performance depends on wrapper technics.

- Pure Python

SWIG

– library generated by Swig

– Python/C API

Data format performances

- Object attribute getter and setter
  - Python/C API 75 times faster than Swig generated code

- Load / Save (binary file) in memory
  - Python/C API equivalent to C++
  - Pure Python 5 times slower Read
  - Pure Python 20 times slower Write

Mathematical kernel library

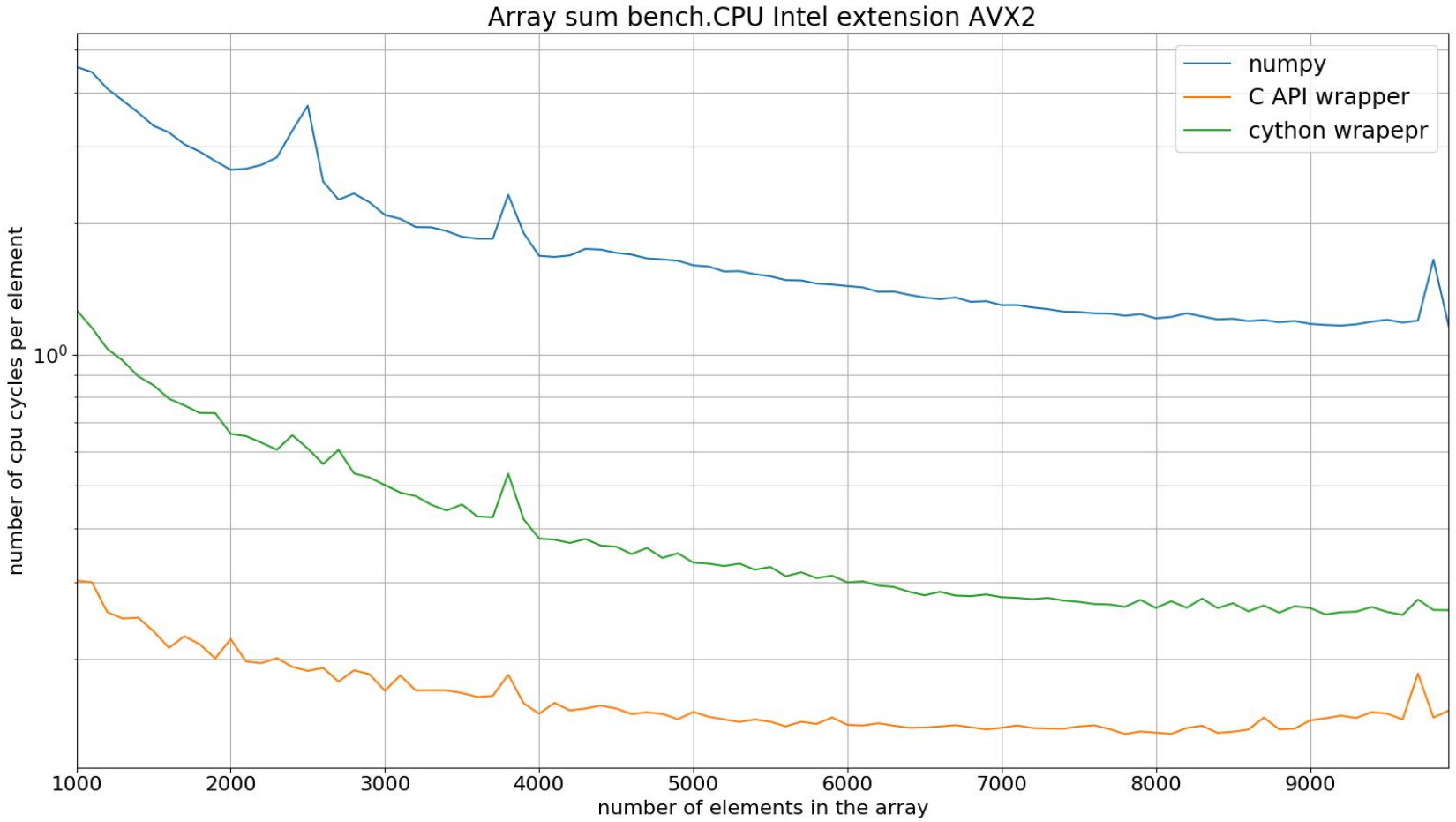
- **Numpy**
  - Python package for scientific computing.
  - common mathematical routines in pre-compiled C/C++ language.

- **plibs_8**
  - **Python 3** math kernel library for Intel.
  - Few mathematical routines in pre-compiled optimized C/C++ language.

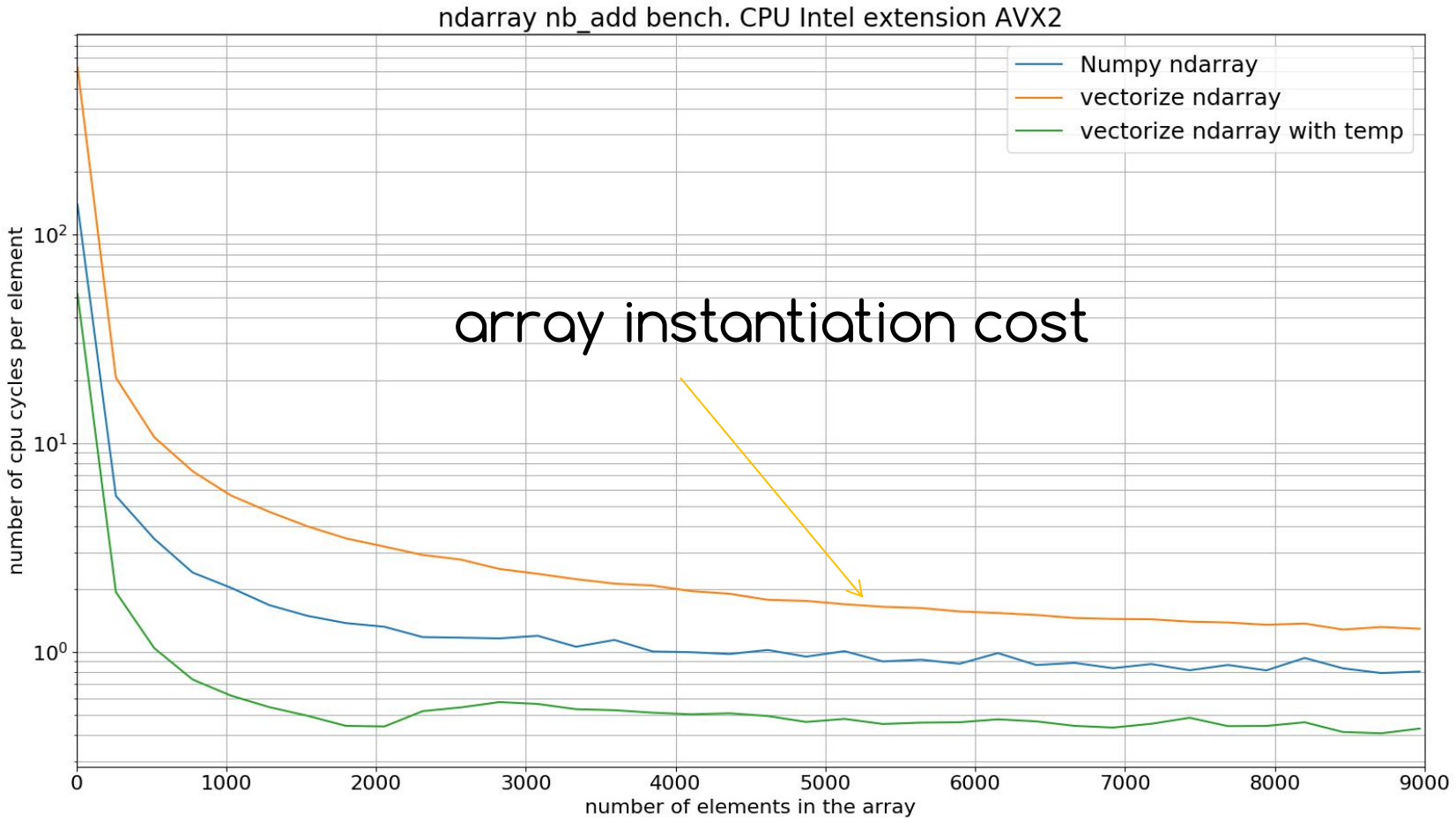
Array sum


Array sum. CPU Intel extension AVX2
Numpy
plibs_8
Python
Optimized C
number of cpu cycles per element
10^2
10^1
10^0
10^{-1}
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
number of elements in the array

Array sum Wrapper thechnics



Array addition

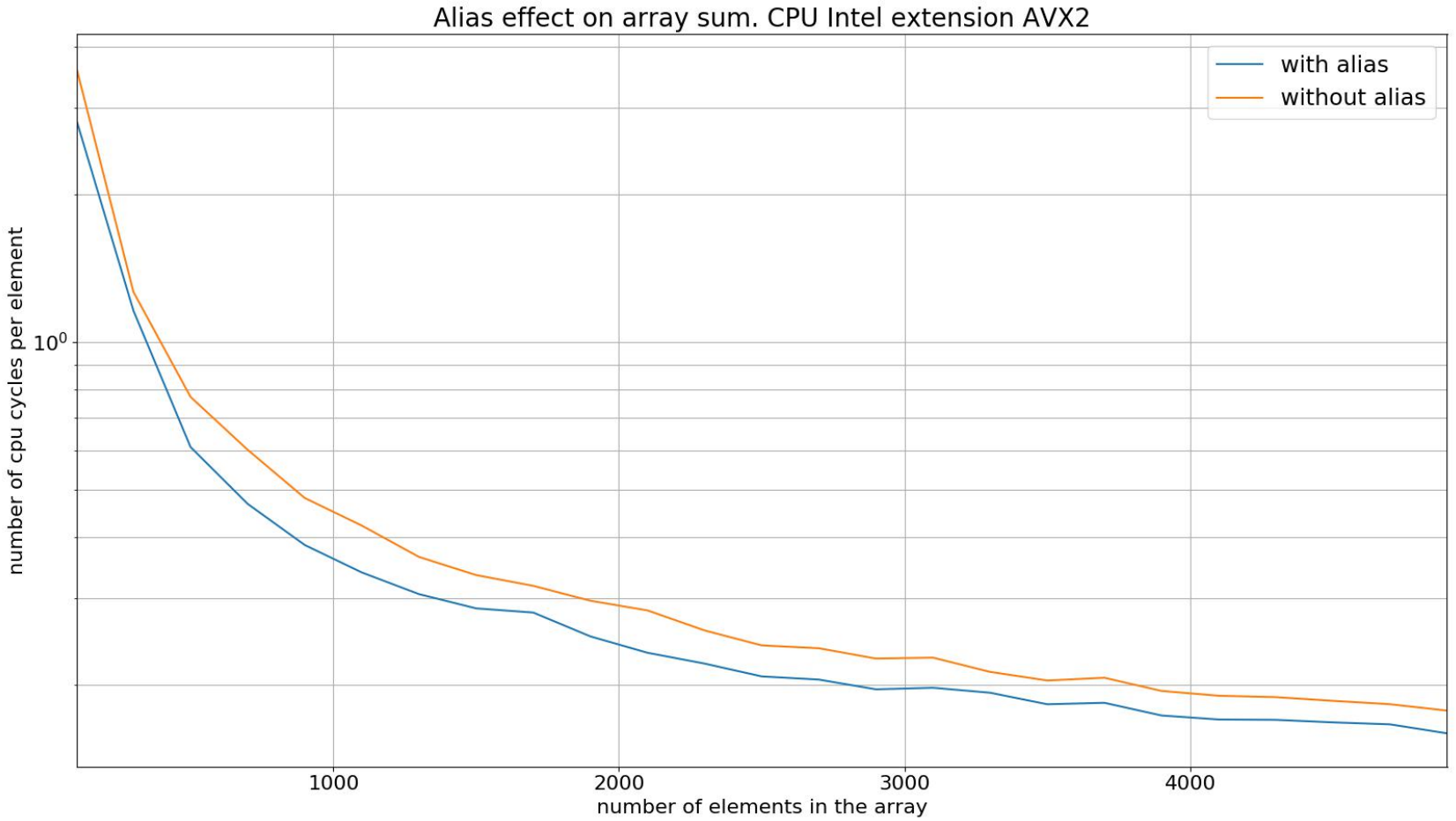


Python Aliasing

```
%import module
%for i in range(loop):
      module.sum(data)
```

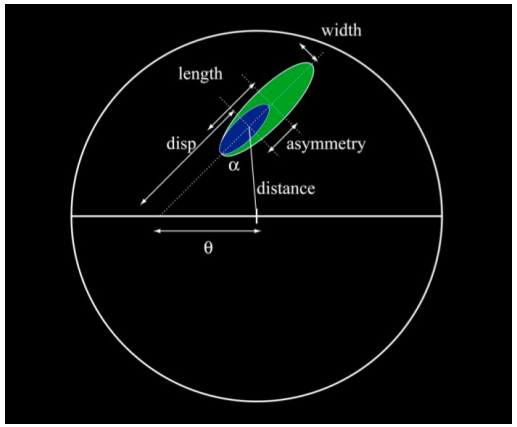
```
%import module
%module_sum = module.sum
%for i in range(loop):
      module_sum(data)
```

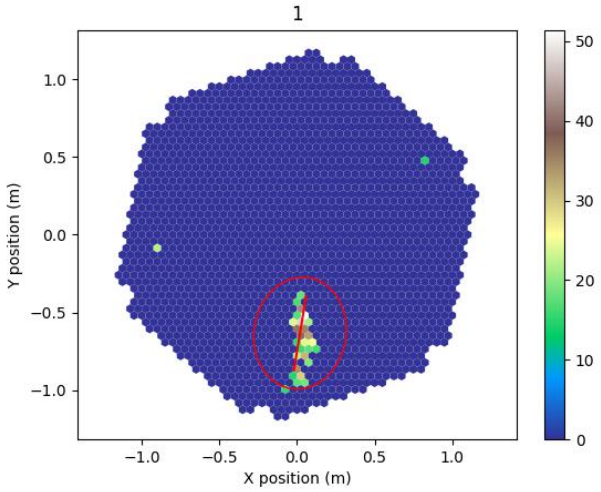
Python Aliasing



HPC Algorithm Hillas

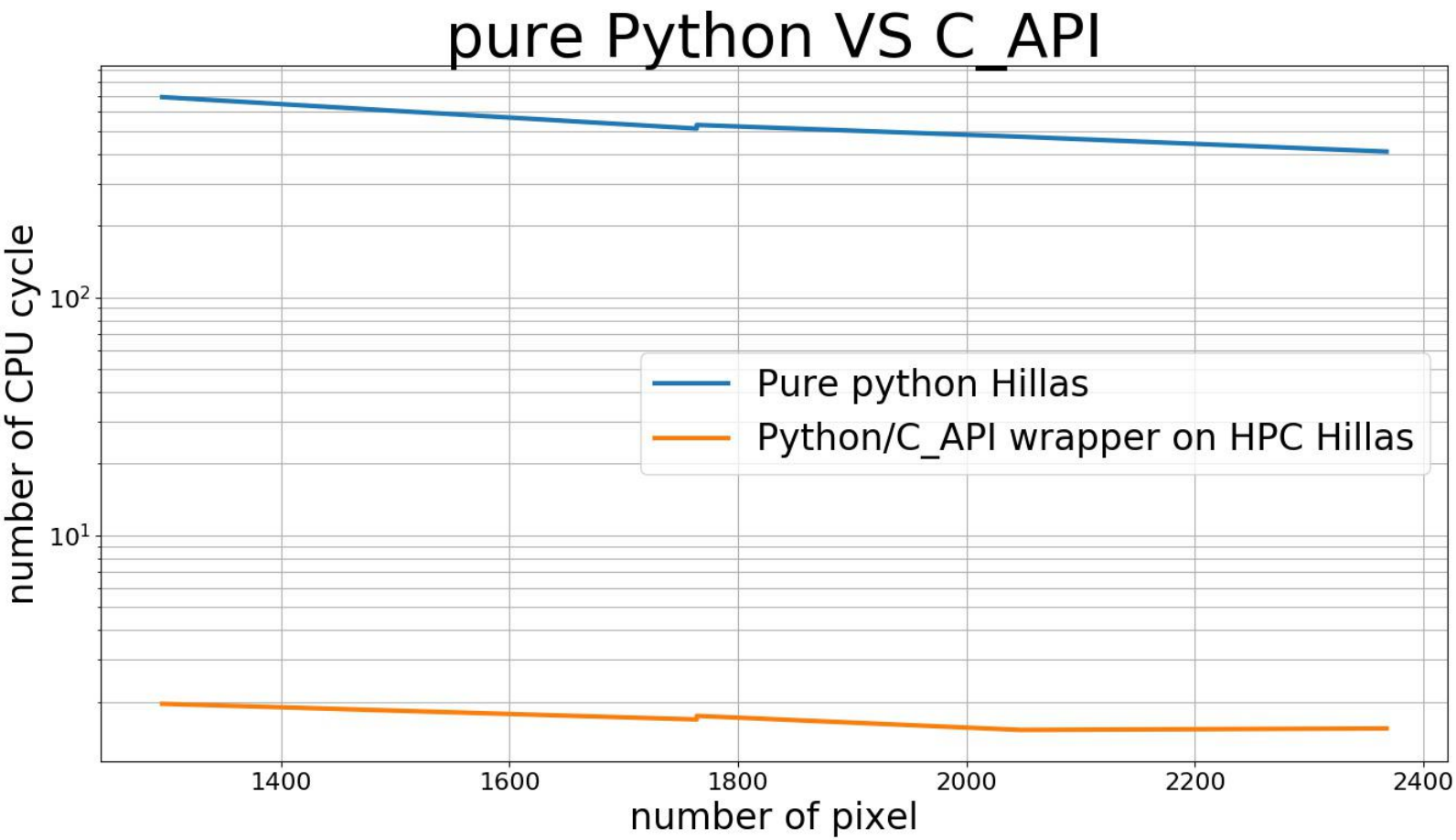
- It parameterizes particle in a cosmic ray air shower.





- vector reduction + momenta (1,2,3,4)

Hillas


pure Python VS C_API
number of CPU cycle
10^2
10^1
Pure python Hillas
Python/C_API wrapper on HPC Hillas
1400
1600
1800
2000
2200
2400
number of pixel

Best practices for python developers

Use compiled code when program contains some high CPU usage computation.

Numpy effectiveness could be improved by manually allocating array memory.

Most of Numpy performances could be outperformed by optimized compiled code.

Avoiding memory allocation drastically reduces the amount of CPU time.

Best practices for python developers

CPU consumption is dominated by Python function call.

It is recommended to use alias in hot spots.

Python and Numpy C API is much more faster than wr cython or automatic tool like swig.

Conclusion

- Good practices for Python programming language.
- These good practices can substantially improve performances.
- Importance compiled libraries for intensive computing.
- Top-Down approach is viable.