Deep issues of the Python ecosystem soon fixed through the HPy project?

Pierre Augier

CNRS
Université Grenoble Alpes
LEGI Laboratoire des Ecoulements Geophysiques et Industriels

Yet another presentation about Python problems...

- “Python slow”
- Python ~= CPython
- Role of 3rd party libraries and extensions

But first, some good news!

News and progresses...

- Usability for different applications
  - Web frontend, WebAssembly, Pyodine, Pyscript
  - Beeware
  - Textual
- Packaging
  - Pip, isolated build, Poetry, ...
  - Conda-forge, Mamba, ...
  - PyPy + Conda-forge

News performance

- No GIL proof of concept
- Python array API standard
- Pythran in Scipy and Scikit-image
- Faster CPython project (Microsoft, with Guido van Rossum)
  - 3.11 (2022/10) x1.25 faster than 3.10
  - 3.12 (2023/10) ?
  - 3.13 (2024/10) ?
  - 3.14 (2025/10) ?

Only pure Python code...

Target: x5 faster!

About numerical Python problems...

- “Python slow”
- Python ~= CPython
- Role of 3rd party libraries and extensions

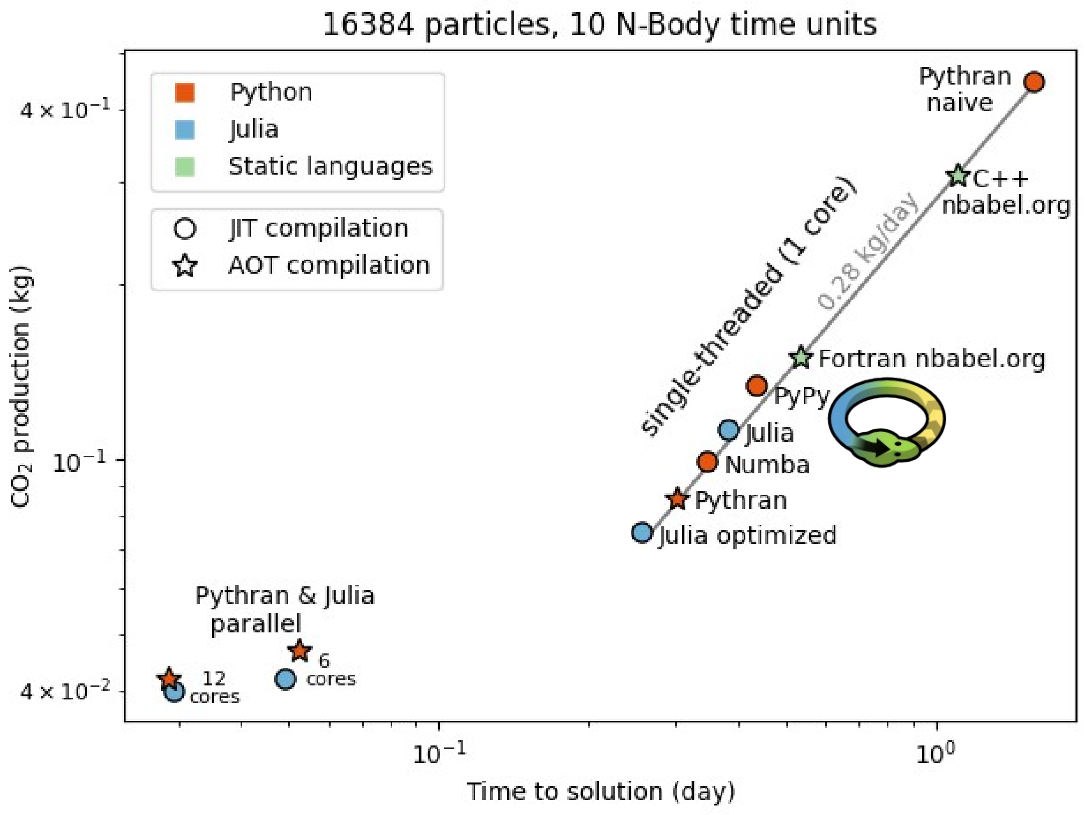
Why?

Python slow ? Bad for CO_2 ?

- Actually, CPython slow (pure Python code and extensions)
- Slow interpreter != inefficient programs!
  - Avoid the interpreter in hot loops (long calls native functions)
  - Numpy with high level prog.
  - Cython, Pythran, Numba, Numexpr...
- Other implementations (PyPy, GraalPy) much faster

GraalVM™

Python slow ? Bad for CO_2 ?



Taken from Augier et al.
Reducing the ecological impact of computing through education and python compilers.
Nature Astronomy, 2021

Benchmark of different **implementations** in different languages.

CPython slow

- Generalist interpreter
- Performance less important than simplicity of implementation and robustness
- Old internal design
  - Garbage collector: reference counting
  - No specialized objects (for example lists)
  - No JIT
  - ...

It is changing with the Faster CPython project

CPython slow, consequences

- Hard to write numerical code in pure Python
- Language not designed in this direction (lacks “math arrays”, “arrays of objects”, ...)
- Needs things like NumPy

  (C extensions, using the CPython C API)

C API and ABI

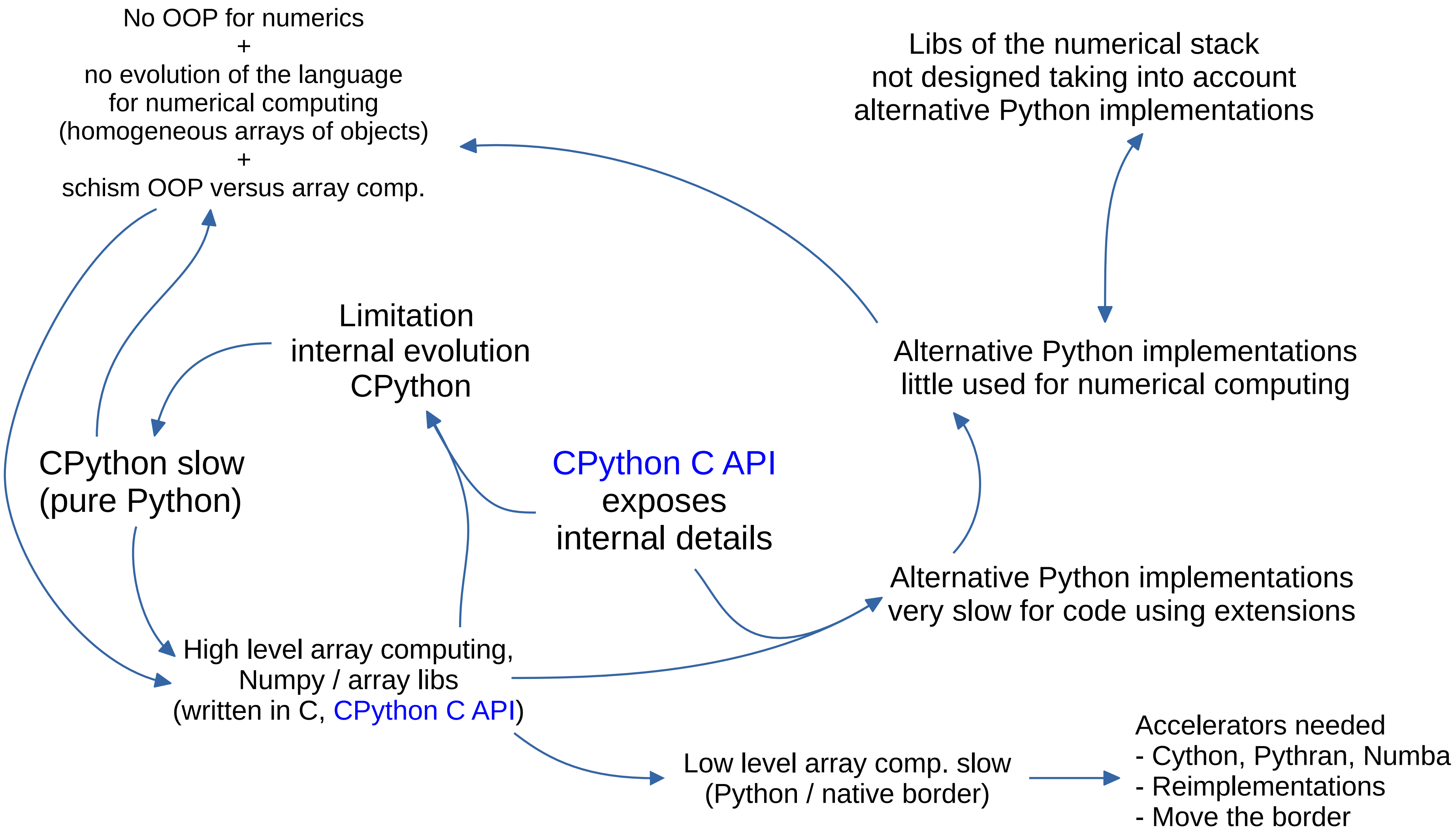
- API (Application Programming Interface) involves source code: content of headers, function declarations, macros, structs, ...
- ABI (Application Binary Interface) involves compiled code: function objects, struct memory layout, etc.

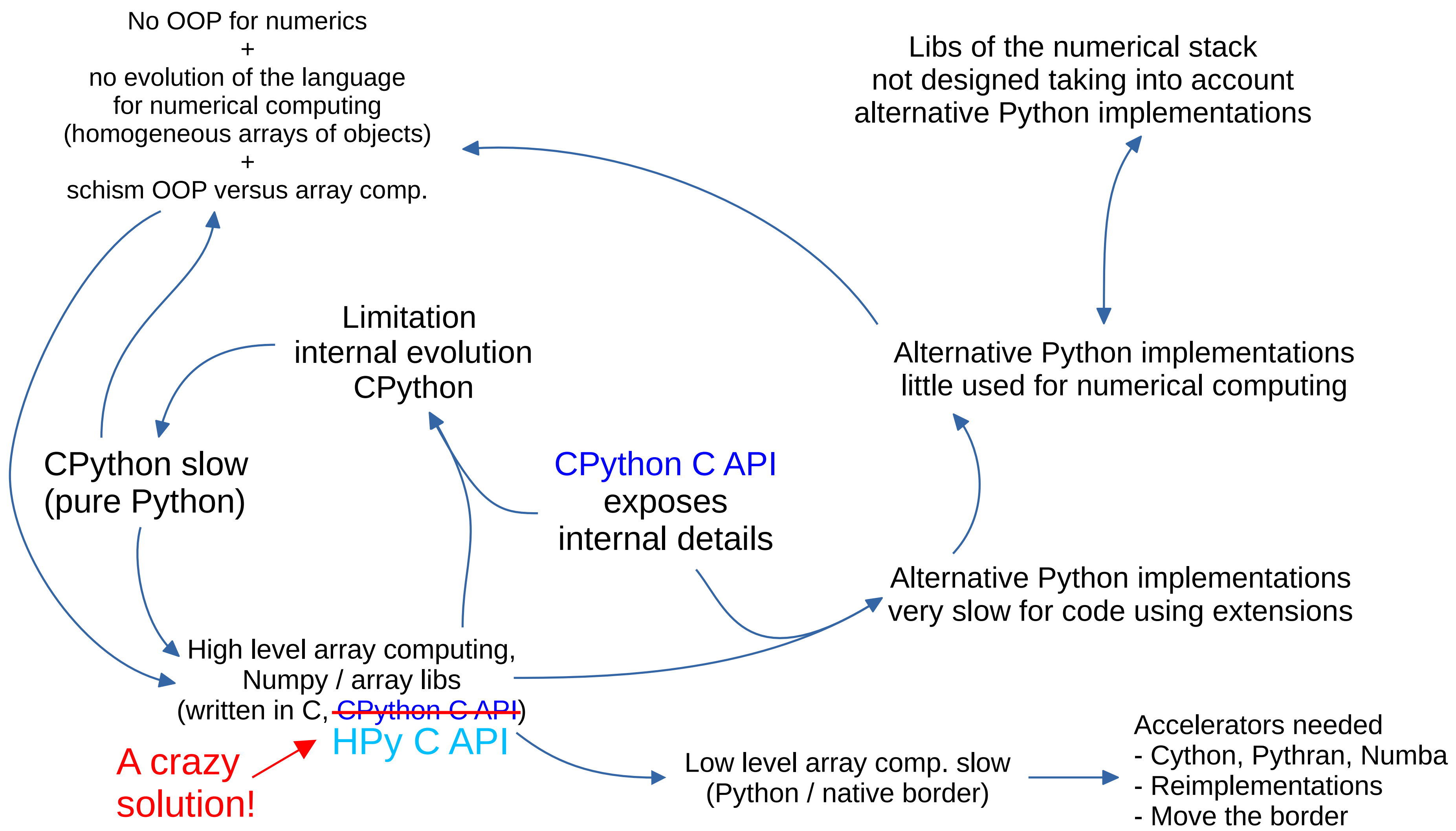
CPython C API

- **A huge success!**
- No proper design, evolve organically
- Exposes (assumptions about) **too many implementation details**
- A nightmare for alternative Python implementations (supporting extensions requires mimicking the internals of CPython)

Alternative Python implementations

- PyPy
  (fast, tracing JIT, tracing garbage collector)
- MicroPython
- GraalPy (GraalVM™, Oracle Labs, Java, LLVM)
- RustPython
- ...
- Multiple projects failed...


No OOP for numerics + no evolution of the language for numerical computing (homogeneous arrays of objects) + schism OOP versus array comp.
Libs of the numerical stack not designed taking into account alternative Python implementations
Limitation internal evolution CPython
Alternative Python implementations little used for numerical computing
CPython slow (pure Python)
CPython C API exposes internal details
Alternative Python implementations very slow for code using extensions
High level array computing, Numpy / array libs (written in C, CPython C API)
Low level array comp. slow (Python / native border)
Accelerators needed
- Cython, Pythran, Numba
- Reimplementations
- Move the border


No OOP for numerics + no evolution of the language for numerical computing (homogeneous arrays of objects) + schism OOP versus array comp.
Libs of the numerical stack not designed taking into account alternative Python implementations
Limitation internal evolution CPython
Alternative Python implementations little used for numerical computing
CPython slow (pure Python)
CPython C API exposes internal details
Alternative Python implementations very slow for code using extensions
High level array computing, Numpy / array libs (written in C, ~~CPython C API~~)
HPy C API
A crazy solution!
Low level array comp. slow (Python / native border)
Accelerators needed
- Cython, Pythran, Numba
- Reimplementations
- Move the border

HPy - A better C API for Python

Principles:

- Based on handles to represent references to Python objects
- Hides implementations details (no assumption)
- 1 API and different ABIs (CPython ABIs and 1 HPy Universal ABI)

Advantages:

- **Zero overhead on CPython**
- **Much faster on alternative implementations** such as PyPy, GraalPython
- **Universal binaries**: extensions built for the HPy Universal ABI can be loaded unmodified on CPython, PyPy, GraalPython, etc. No need to rebuild for different versions (3.9, 3.10, …)!
- **A migration path** for mixing legacy C-API with HPy API
- **Debug mode**: to identify common problems such as memory leaks, invalid lifetime of objects, invalid usage of API, … No need to rebuild!
- **Nicer API**: smaller, simpler, more consistent

Scikit-image example

- Scikit-image written in Python and Cython
- Depends on Numpy, Matplotlib, ...
- Extensions
  - C code produced via Cython and Pythran
  - Using the CPython C API and the Numpy C API

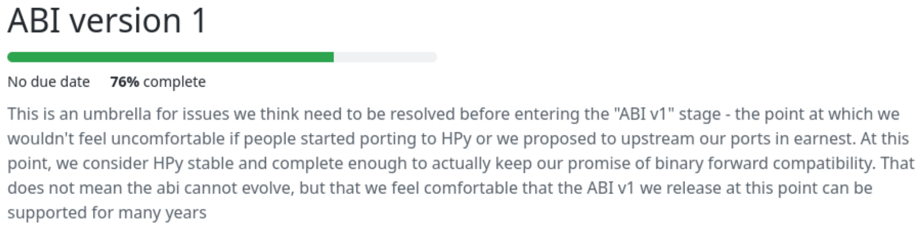
Scikit-image fully using HPy?

- Numpy HPy porting
- New Numpy HPy C API
- Matplotlib HPy porting
- Cython backend producing HPy code
- Pythran backend producing HPy code

And a HPy universal wheel can be created!

HPy: current status in 2023? Probability of success?

- 4 years after the first commit, still in development (HPy 0.0.4, alpha testing)
- Actively developed (by PyPy, GraalPy and Numpy core devs)
- Milestone “ABI version 1” soon reached!



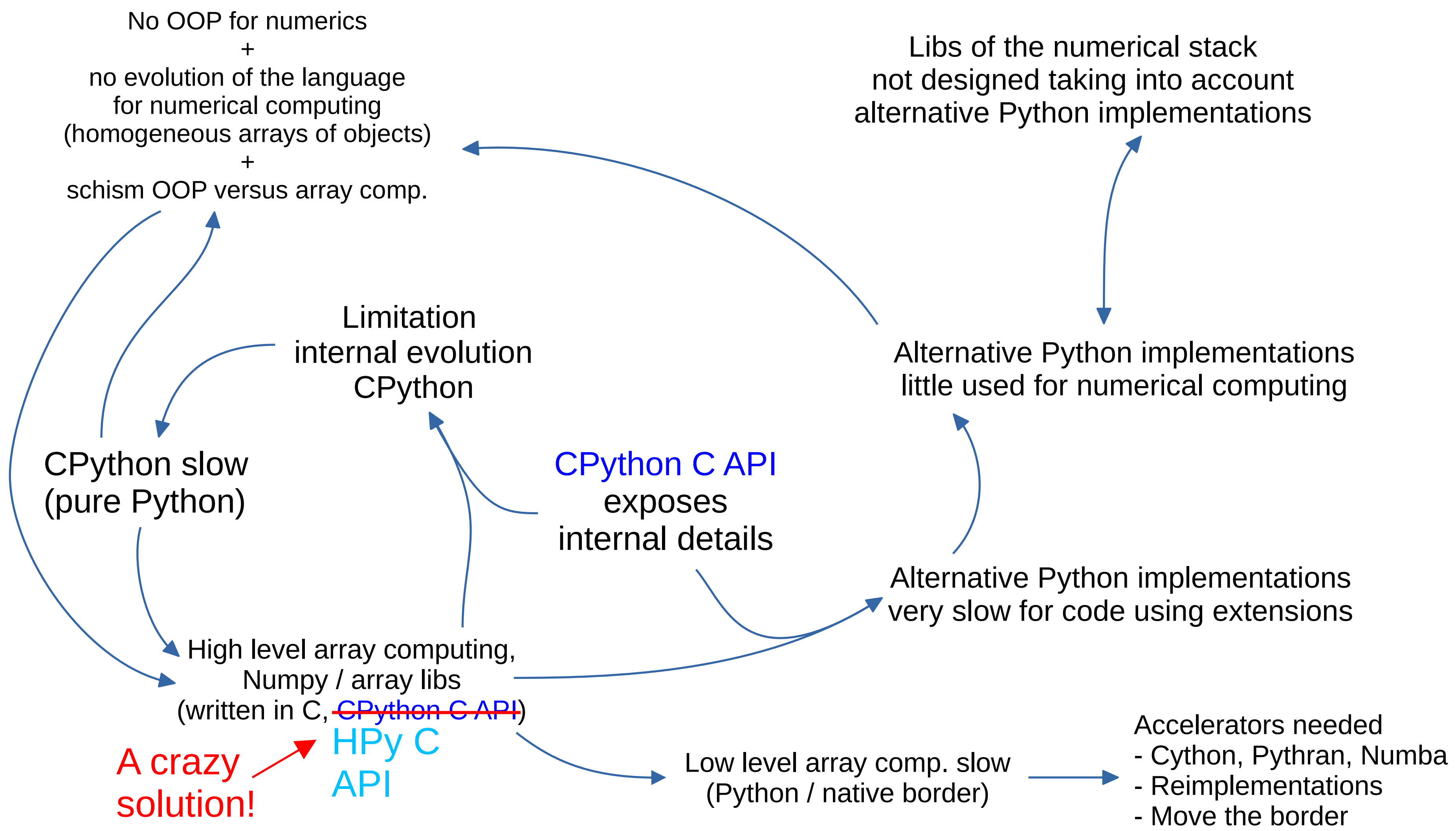
ABI version 1

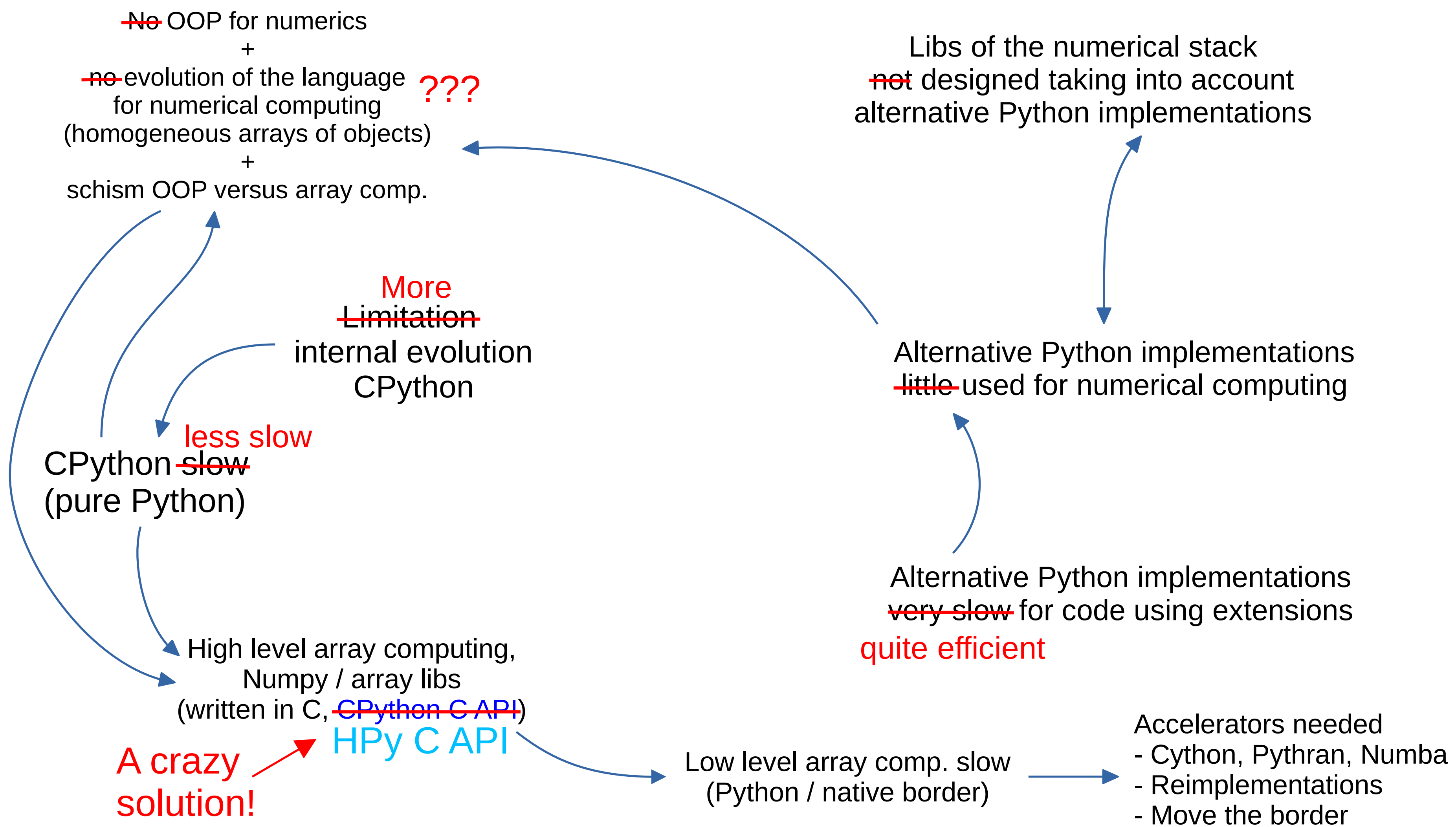
No due date **76%** complete

This is an umbrella for issues we think need to be resolved before entering the "ABI v1" stage - the point at which we wouldn't feel uncomfortable if people started porting to HPy or we proposed to upstream our ports in earnest. At this point, we consider HPy stable and complete enough to actually keep our promise of binary forward compatibility. That does not mean the abi cannot evolve, but that we feel comfortable that the ABI v1 we release at this point can be supported for many years

- Few packages ported (Matplotlib!)
- Numpy port in progress but no more big technical uncertainties!

My guess: end users may be able to get the first benefeces of HPy in few months


No OOP for numerics + no evolution of the language for numerical computing (homogeneous arrays of objects) + schism OOP versus array comp.
Libs of the numerical stack not designed taking into account alternative Python implementations
Limitation internal evolution CPython
CPython slow (pure Python)
CPython C API exposes internal details
Alternative Python implementations little used for numerical computing
Alternative Python implementations very slow for code using extensions
High level array computing, Numpy / array libs (written in C, ~~CPython C API~~)
HPy C API
A crazy solution!
Low level array comp. slow (Python / native border)
Accelerators needed
- Cython, Pythran, Numba
- Reimplementations
- Move the border


~~No~~ OOP for numerics
+
~~no~~ evolution of the language for numerical computing (homogeneous arrays of objects)
+
schism OOP versus array comp.
???
Libs of the numerical stack ~~not~~ designed taking into account alternative Python implementations
More
~~Limitation~~
internal evolution CPython
Alternative Python implementations ~~little~~ used for numerical computing
less slow
CPython ~~slow~~
(pure Python)
Alternative Python implementations ~~very slow~~ for code using extensions
quite efficient
High level array computing, Numpy / array libs (written in C, ~~CPython C API~~)
HPy C API
A crazy solution!
Low level array comp. slow (Python / native border)
Accelerators needed
- Cython, Pythran, Numba
- Reimplementations
- Move the border

Conclusions and perspectives

During the next years, we’ll see remarkable performance improvements for Python

- Faster CPython project
- HPy

Conclusions and perspectives

About HPy

- Towards a multi implementations ecosystem
- Universal wheels great for users!
- HPy opens a lot of possibilities
- What about low level code using extensions?
  - Python/Numpy accelerators still useful
  - GraalPy very promising (JIT across language boundaries)…

How to help? https://hpyproject.org/