High-**P**erformance **C**omputing 20**25**

Some notes on Python for HPC

Aryan Eftekhari | **U**niversità della **S**vizzera **i**taliana (USI) | **I**nstitute of **C**omputing (CI)

Python for HPC?

Not specifically ***designed*** for HPC, but a language widely used in ***many areas***, including HPC.

Python vs C++

Python

- **interpreted:** The code is interpreted by python.
- **untyped:** Variable types are automatically assigned.
- **dynamically typed:** Variable ↔ value associated at runtime.
- **code style:** code blocks by “indentation”!

C++

- **compiled:** The code is compiled by a C++ compiler to machine code.
- **typed:** Variable types need to be assigned.
- **statically typed:** Variable ↔ value associated at compile time.
- **code style**: code blocks by {}, don't care about spacing.

What is `python` ? It’s a program that runs python code, i.e., an *interpreter*.

Python For HPC
Why?

The time invested in doing an experiment is short for Python.

Python can be very efficient way to attain results, and there are tools to make it applicable in a quasi-HPC environment.

TIOBE programming community index

Nov 2024	Nov 2023	Change	Programming Language	Ratings	Change
1	1		Python	22.85%	+8.69%
2	**3**	↑	**C++**	**10.64%**	**+0.29%**
3	4	↑	Java	9.60%	+1.26%
4	2	↓	C	9.01%	-2.76%
5	5		C#	4.98%	-2.67%
6	6		JavaScript	3.71%	+0.50%
7	13	⇈	Go	2.35%	+1.16%
8	12	⇈	Fortran	1.97%	+0.67%
9	8	↓	Visual Basic	1.95%	-0.15%
10	9	↓	SQL	1.94%	+0.05%

Global Interpreter Lock (GIL):

(–) The thread within each process cannot perform task simultaneously!

(+) This is intended to improve performance for single-threaded programs.

```
# Thread library
import threading
# Only ONE thread per process executes Python bytecode at a time.


# Multiprocessing library
import multiprocessing
# True parallelism via multiple OS processes (bypasses the GIL) on a single
machine. No support for parallelization over multiple compute nodes.
```

Python For HPC

NUMBA—JIT compiler for Python

Just-**I**n-**T**ime compiler, compiles code during runtime, as opposed to GCC, which is *before* execution.

```
// install
pip3 install numba
```

```
from numba import jit,prange

@jit
def jit_monte_carlo_pi(nsamples):
    sum = 0
    for i in range(nsamples):
        sum+=calculation(i)
    return sum

@jit(parallel=True)
def prl_monte_carlo_pi(nsamples):
    sum = 0
    for i in prange(nsamples):
        sum+=calculation(i)
    return sum
```

```
//run
python3 main.py
```

Numba is a **JIT compiler** implemented as a library for Python, enabling significant performance enhancements with minimal effort!

See numba.pydata.org for details.

Python For HPC
MPI4PY—MPI for Python

"Bindings for the MPI standard, allowing Python to exploit multiple processors on workstations, clusters and supercomputers."

```
// install
pip3 install mpi4py
```

```
from mpi4py import MPI
import numpy as np

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

data=np.zeros(10)

if rank == 0:
   comm.send(np.ones(10), dest=1, tag=100)
elif rank == 1:
   data=comm.recv(source=0, tag=100)
```

```
//run
mpirun -np 4 python3 main.py
```

mpi4py is the MPI standard (a library) in python, and it supports *NumPy* !

See mpi4py.readthedocs.io for details.

Leverage C++ **performance** and keep the **simplicity/functionality** of Python.

```cpp
#include <pybind11/pybind11.h>

double bigcalc(double x){
   Return x+1.0;
};

PYBIND11 MODULE(module, m) {
m.def("bigcalc", &bigcalc,
"some_info");
}
```


```
Import module
c=module.bigcalc(2)
```

See pybind11.readthedocs.io for details.