numba-mpi

Numba @njittable MPI wrappers tested on Linux macOS and Windows

S. Arabas[1], O. Bulenok[1], K. Derlatka[1], M. Manna[1] & D. Zwicker[2]

FOSDEM'23 HPC, Big Data, and Data Science Devroom @ ULB (Feb 5 2023)

[1]: Jagiellonian University, Kraków, Poland
[2]: Max Planck Institute for Dynamics and Self-Organization, Göttingen, Germany

Python & HPC?

photo: Nature, doi:10.1038/d41586-019-03167-2

Perkel 2019 (Nature)
doi:10.1038/d41586-019-02310-3

“*In scripting languages such as Python,*
users type code into an interactive editor line by line”

Perkel 2019 (Nature)
doi:10.1038/d41586-019-02310-3

“*In scripting languages such as Python,*
users type code into an interactive editor line by line”

Perkel 2020 (Nature)
doi:10.1038/d41586-020-03382-2

“*level of computational performance*
that Python simply couldn’t deliver”

Perkel 2019 (Nature)
doi:10.1038/d41586-019-02310-3

“*In scripting languages such as Python, users type code into an interactive editor line by line*”

Perkel 2020 (Nature)
doi:10.1038/d41586-020-03382-2

“*level of computational performance that Python simply couldn’t deliver*”

Harris et al. 2020 (Nature)
doi:10.1038/s41586-020-2649-2

“NumPy ... runs on machines ranging from embedded devices to the world’s largest supercomputers, with performance approaching that of compiled languages”

Perkel 2019 (Nature)
doi:10.1038/d41586-019-02310-3

“In scripting languages such as Python, users type code into an interactive editor line by line”

Perkel 2020 (Nature)
doi:10.1038/d41586-020-03382-2

“level of computational performance that Python simply couldn’t deliver”

Zwart 2020 (Nature Astronomy)
doi:10.1038/s41550-020-1208-y

“astronomers should avoid interpreted scripting languages such as Python … in principle, Numba and NumPy can lead to an enormous increase in speed …
… reconsider teaching Python to university students.”

Harris et al. 2020 (Nature)
doi:10.1038/s41586-020-2649-2

“NumPy … runs on machines ranging from embedded devices to the world’s largest supercomputers, with performance approaching that of compiled languages”

Perkel 2019 (Nature)
doi:10.1038/d41586-019-02310-3

“*In scripting languages such as Python, users type code into an interactive editor line by line*”

Perkel 2020 (Nature)
doi:10.1038/d41586-020-03382-2

“*level of computational performance that Python simply couldn’t deliver*”

Zwart 2020 (Nature Astronomy)
doi:10.1038/s41550-020-1208-y

“*astronomers should avoid interpreted scripting languages such as Python ... in principle, Numba and NumPy can lead to an enormous increase in speed ... ... reconsider teaching Python to university students.*”

Harris et al. 2020 (Nature)
doi:10.1038/s41586-020-2649-2

“NumPy ... runs on machines ranging from embedded devices to the world’s largest supercomputers, with performance approaching that of compiled languages”

Virtanen et al. 2020 (Nature Methods)
doi:10.1038/s41592-019-0686-2

“implementing new functionality, Python is the still the language of choice ... [SciPy] full test suite passed with the PyPy JIT compiler at the 1.0 release point”

Perkel 2019 (Nature)
doi:10.1038/d41586-019-02310-3

"*In scripting languages such as Python, users type code into an interactive editor line by line*"

Perkel 2020 (Nature)
doi:10.1038/d41586-020-03382-2

"*level of computational performance that Python simply couldn't deliver*"

Zwart 2020 (Nature Astronomy)
doi:10.1038/s41550-020-1208-y

"*astronomers should avoid interpreted scripting languages such as Python ... in principle, Numba and NumPy can lead to an enormous increase in speed ... ... reconsider teaching Python to university students.*"

↑

papers promoting Julia, Rust, ...

Harris et al. 2020 (Nature)
doi:10.1038/s41586-020-2649-2

"NumPy ... runs on machines ranging from embedded devices to the world's largest supercomputers, with performance approaching that of compiled languages"

Virtanen et al. 2020 (Nature Methods)
doi:10.1038/s41592-019-0686-2

"implementing new functionality, Python is the still the language of choice ... [SciPy] full test suite passed with the PyPy JIT compiler at the 1.0 release point"

↑

papers on Python packages

python™

- as a language, lacks support for multi-dimensional arrays or number-crunching facilities

python™

- as a language, lacks support for multi-dimensional arrays or number-crunching facilities
- ⇝ leaving it to be provided by external packages

python™

- as a language, lacks support for multi-dimensional arrays or number-crunching facilities
- ⇝ leaving it to be provided by external packages
- CPython interpretter is not the only Python implementation

- as a language, lacks support for multi-dimensional arrays or number-crunching facilities
- ⇝ leaving it to be provided by external packages
- CPython interpretter is not the only Python implementation
- ⇝ solutions exist streamlining just-in-time compilation for Python code

python™

- as a language, lacks support for multi-dimensional arrays or number-crunching facilities
- ⇝ leaving it to be provided by external packages
- CPython interpretter is not the only Python implementation
- ⇝ solutions exist streamlining just-in-time compilation for Python code
- NumPy is not the only implementation of the NumPy API

python™

- as a language, lacks support for multi-dimensional arrays or number-crunching facilities
- ⇝ leaving it to be provided by external packages
- CPython interpretter is not the only Python implementation
- ⇝ solutions exist streamlining just-in-time compilation for Python code
- NumPy is not the only implementation of the NumPy API
- ⇝ alternatives embedded in JIT/GPU frameworks leverage typing & concurrency

python™

- as a language, lacks support for multi-dimensional arrays or number-crunching facilities
- ⇝ leaving it to be provided by external packages
- CPython interpretter is not the only Python implementation
- ⇝ solutions exist streamlining just-in-time compilation for Python code
- NumPy is not the only implementation of the NumPy API
- ⇝ alternatives embedded in JIT/GPU frameworks leverage typing & concurrency

- **Python lets you glue (and package) together these technologies**

JIT-compiled Python & NumPy API

Numba

Numba is an open source JIT compiler

that translates a subset of Python and NumPy code into fast machine code ...

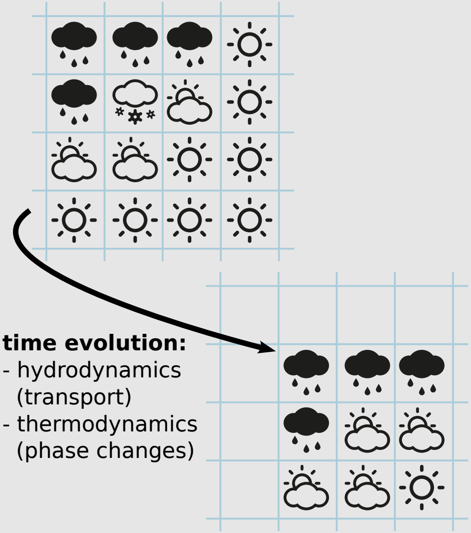
Numba

`Numba` is an open source JIT compiler

that translates a subset of Python and NumPy code into fast machine code ...

... at runtime using the industry-standard `LLVM` compiler library

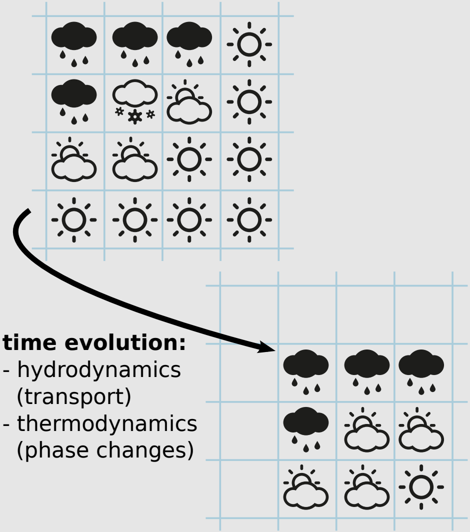
NWP-related prototype problem



time evolution:

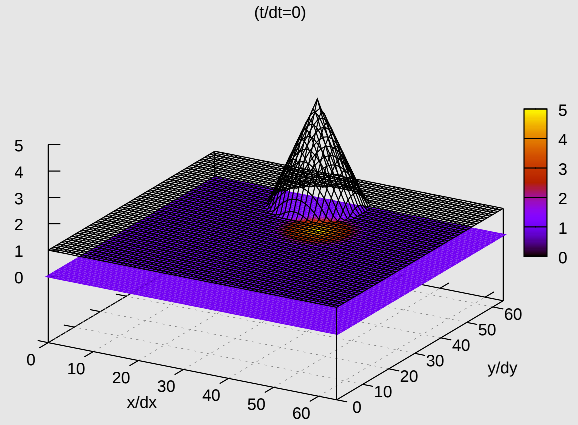
- hydrodynamics (transport)
- thermodynamics (phase changes)

NWP-related prototype problem

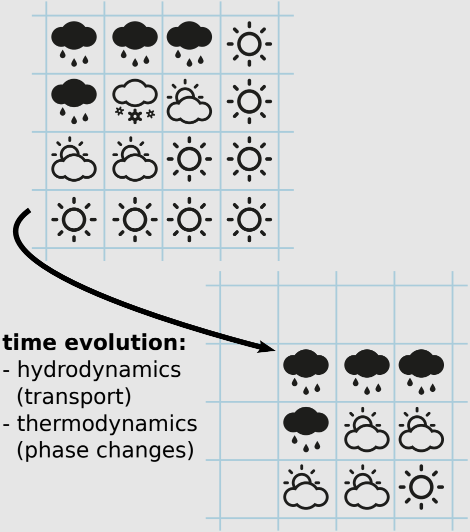


time evolution:
- hydrodynamics (transport)
- thermodynamics (phase changes)

numerical solution for transport-only PDE (2D)

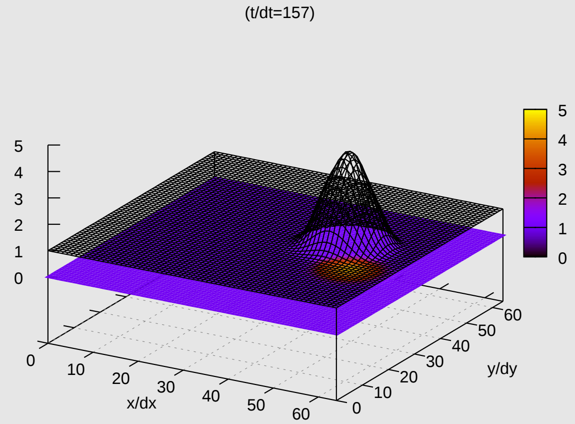


NWP-related prototype problem

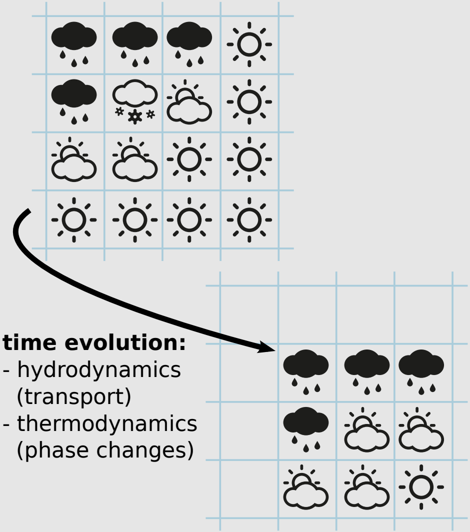


time evolution:
- hydrodynamics (transport)
- thermodynamics (phase changes)

numerical solution for transport-only PDE (2D)

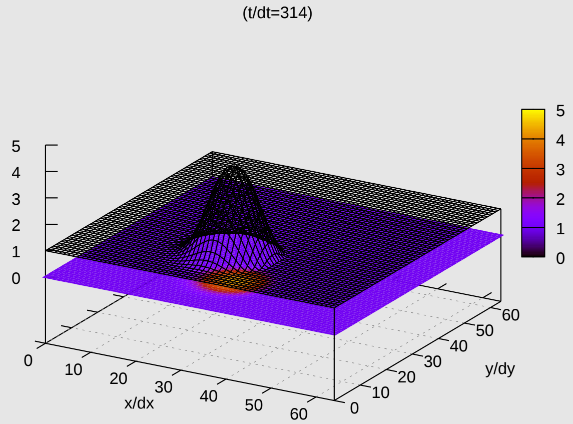


NWP-related prototype problem

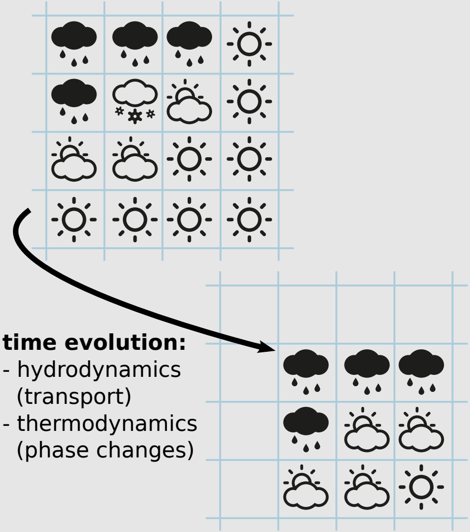


time evolution:
- hydrodynamics (transport)
- thermodynamics (phase changes)

numerical solution for transport-only PDE (2D)

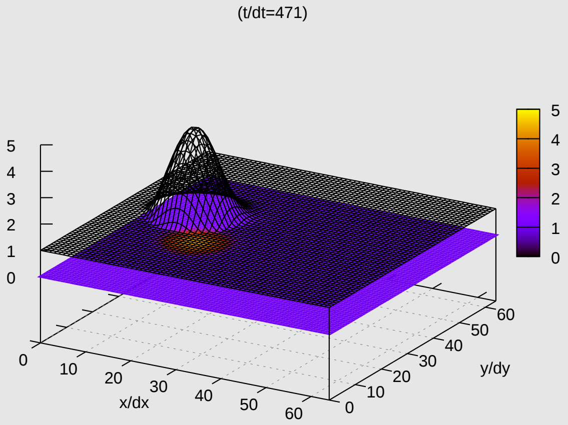


NWP-related prototype problem

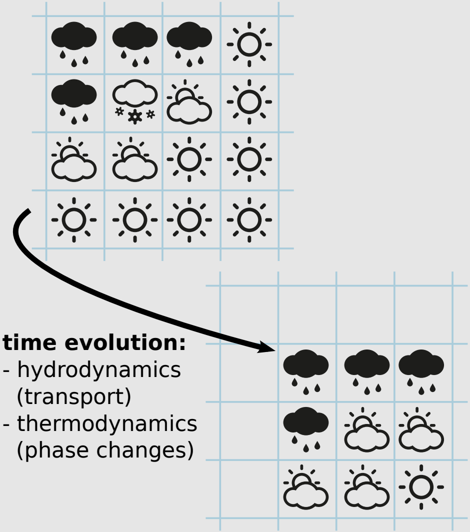


time evolution:
- hydrodynamics (transport)
- thermodynamics (phase changes)

numerical solution for transport-only PDE (2D)

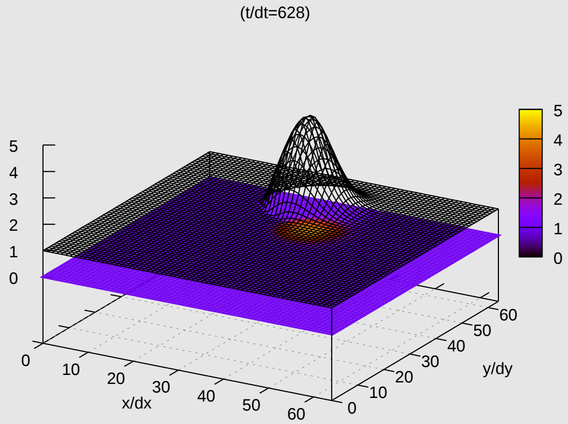


NWP-related prototype problem

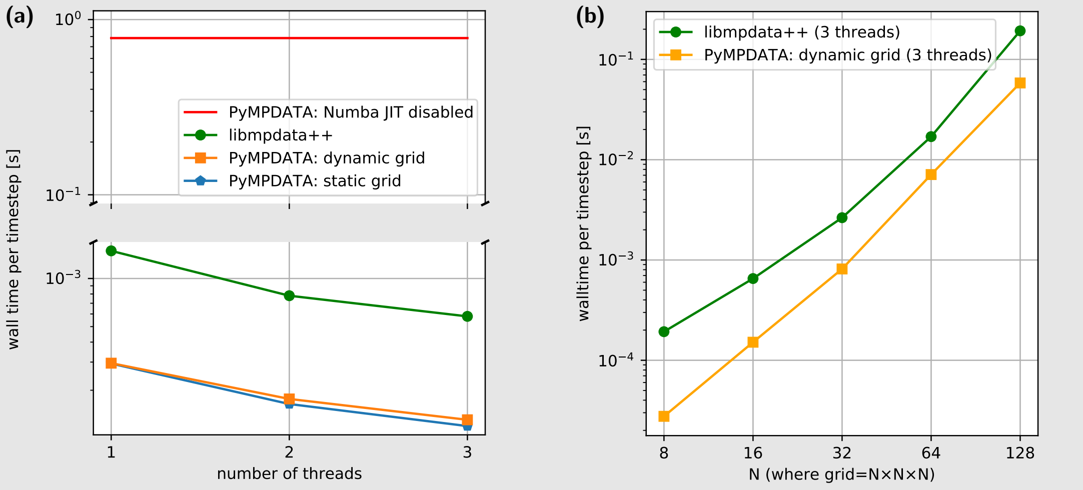


time evolution:
- hydrodynamics
  (transport)
- thermodynamics
  (phase changes)

numerical solution for transport-only PDE (2D)



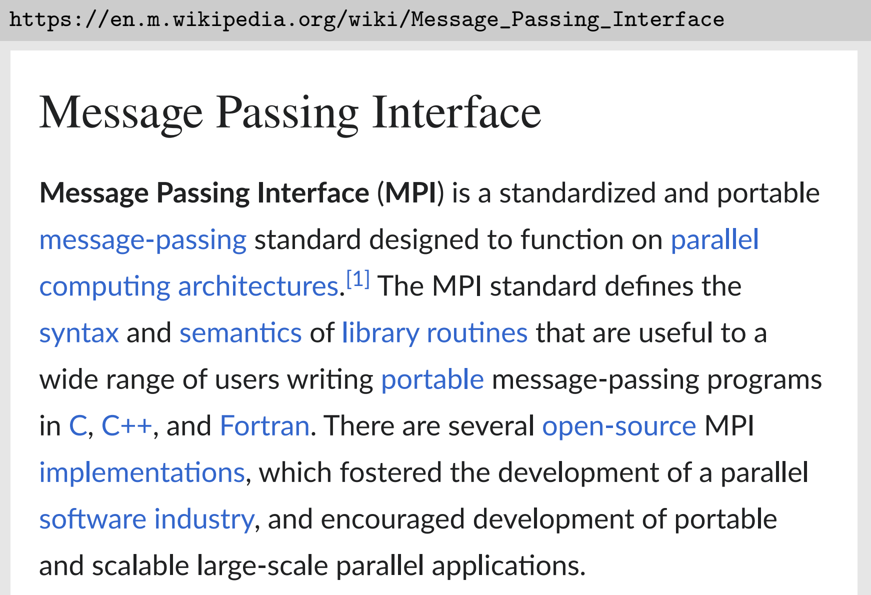
example performance comparison: Bartman et al. 2022 (JOSS) doi:10.21105/joss.03896



PyMPDATA ⇝ Numba (loop-based code, tricky for NumPy/CPython)

libmpdata++ ⇝ Blitz++ (OOP code; 5×slower than F77 for small domains, on par for larger ones)

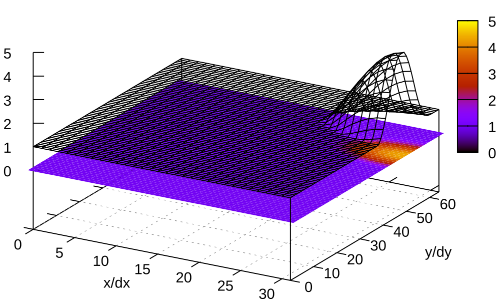
what if we need MPI?

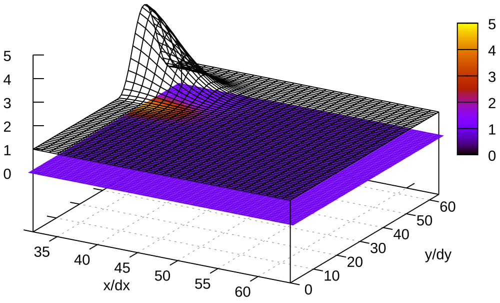


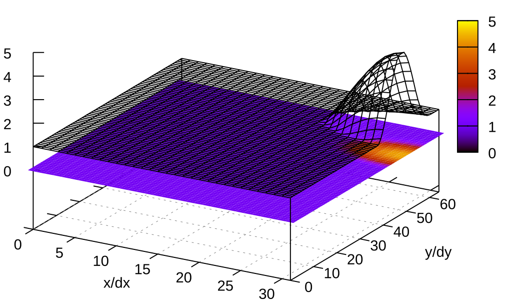
https://en.m.wikipedia.org/wiki/Message_Passing_Interface

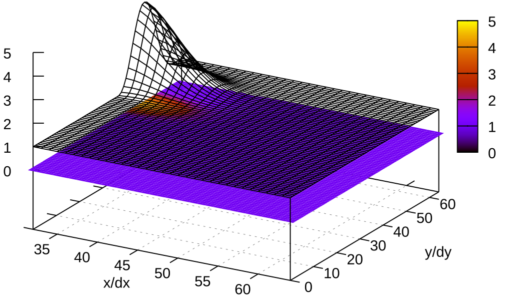
Message Passing Interface

Message Passing Interface (**MPI**) is a standardized and portable message-passing standard designed to function on parallel computing architectures.[1] The MPI standard defines the syntax and semantics of library routines that are useful to a wide range of users writing portable message-passing programs in C, C++, and Fortran. There are several open-source MPI implementations, which fostered the development of a parallel software industry, and encouraged development of portable and scalable large-scale parallel applications.


5
4
3
2
1
0
0
5
10
15
20
25
30
x/dx
0
10
20
30
40
50
60
y/dy
5
4
3
2
1
0


5
4
3
2
1
0
35
40
45
50
55
60
x/dx
0
10
20
30
40
50
60
y/dy
5
4
3
2
1
0





Bangerth & Heister 2013 (Comput. Sci. Discov.) doi:10.1088/1749-4699/6/1/015010

„despite the immense expansion of parallel computation both in the number of machines available as well as in the number of cores per parallel machine since then,

no other parallel programming paradigm has replaced MPI –

even though it is universally acknowledged that MPI is a rather crude way of programming these machines and that MPI might not be successful for machines much larger than the ones available today"

```python
import numba
from mpi4py.MPI import COMM_WORLD

def number_crunching():
    rank = COMM_WORLD.Get_rank()

numba.njit(number_crunching)()
```

```python
import numba
from mpi4py.MPI import COMM_WORLD

def number_crunching():
    rank = COMM_WORLD.Get_rank()

numba.njit(number_crunching)()
```

```
Traceback (most recent call last):
  File ".../numba_plus_mpi4py.py", line 7, in <module>
    numba.njit(number_crunching)()
  File ".../numba/core/dispatcher.py", line 468, in _compile_for_args
    error_rewrite(e, 'typing')
  File ".../numba/core/dispatcher.py", line 409, in error_rewrite
    raise e.with_traceback(None)
numba.core.errors.TypingError: Failed in nopython mode pipeline (step: nopython frontend)

Untyped global name 'COMM_WORLD': Cannot determine Numba type of <class 'mpi4py.MPI.Intracomm'>


File "numba_plus_mpi4py.py", line 5:
def number_crunching():
    rank = COMM_WORLD.Get_rank()
    ^
```

- Numba is the way to make Python code HPC ready
- MPI is the way to do distributed-memory parallelism for HPC

[1]

[2]

- `Numba` is the way to make Python code HPC ready
- `MPI` is the way to do distributed-memory parallelism for HPC
- ... so how come one cannot use `MPI` with `Numba` JIT-compiled code?!

python™ + Numba + MPI $\overset{?!}{=}$ 👎

[1]

[2]

- Numba is the way to make Python code HPC ready
- MPI is the way to do distributed-memory parallelism for HPC
- ... so how come one cannot use MPI with Numba JIT-compiled code?!

python™ + Numba + MPI $\overset{?!}{=}$ 👎

- let's DuckDuckGo it...

1

2

- Numba is the way to make Python code HPC ready

- MPI is the way to do distributed-memory parallelism for HPC

- … so how come one cannot use MPI with Numba JIT-compiled code?!

python™ + Numba + MPI $\overset{?!}{=}$ 👎

- let's DuckDuckGo it… nope, Qwant…

1

2

- Numba is the way to make Python code HPC ready
- MPI is the way to do distributed-memory parallelism for HPC
- ... so how come one cannot use MPI with Numba JIT-compiled code?!

python™ + Numba + MPI $\overset{?!}{=}$ 👎

- let's DuckDuckGo it... nope, Qwant... nope, let's Google...

[1]

[2]

- Numba is the way to make Python code HPC ready
- MPI is the way to do distributed-memory parallelism for HPC
- … so how come one cannot use MPI with Numba JIT-compiled code?!

python™ + Numba + MPI $\overset{?!}{=}$ 👎

- let’s DuckDuckGo it… nope, Qwant… nope, let’s Google… **nothing!**

[1]

[2]

- Numba is the way to make Python code HPC ready
- MPI is the way to do distributed-memory parallelism for HPC
- ... so how come one cannot use MPI with Numba JIT-compiled code?!

python™ + Numba + MPI $\overset{?!}{=}$ 👎

- let's DuckDuckGo it... nope, Qwant... nope, let's Google... **nothing!**
- let's ask on Numba[1] and mpi4py[2] forums (June 2020)...
  ⇝ " *You will not be able to use mpi4py's Cython code, it was not designed for such low-level usage...*"

[1] https://github.com/numba/numba/issues/4115#issuecomment-642474009

[2] https://bitbucket.org/mpi4py/mpi4py/issues/164

- Numba is the way to make Python code HPC ready
- MPI is the way to do distributed-memory parallelism for HPC
- ... so how come one cannot use MPI with Numba JIT-compiled code?!

python™ + Numba + MPI $\overset{?!}{=}$ 👎

- let's DuckDuckGo it... nope, Qwant... nope, let's Google... **nothing!**
- let's ask on Numba[1] and mpi4py[2] forums (June 2020)...
  ⇝ *" You will not be able to use `mpi4py`'s `Cython` code, it was not designed for such low-level usage..."*
- **but it must be gluable!**

[1] https://github.com/numba/numba/issues/4115#issuecomment-642474009

[2] https://bitbucket.org/mpi4py/mpi4py/issues/164

- Numba is the way to make Python code HPC ready
- MPI is the way to do distributed-memory parallelism for HPC
- ... so how come one cannot use MPI with Numba JIT-compiled code?!

python™ + Numba + MPI $\overset{?!}{=}$ 👎

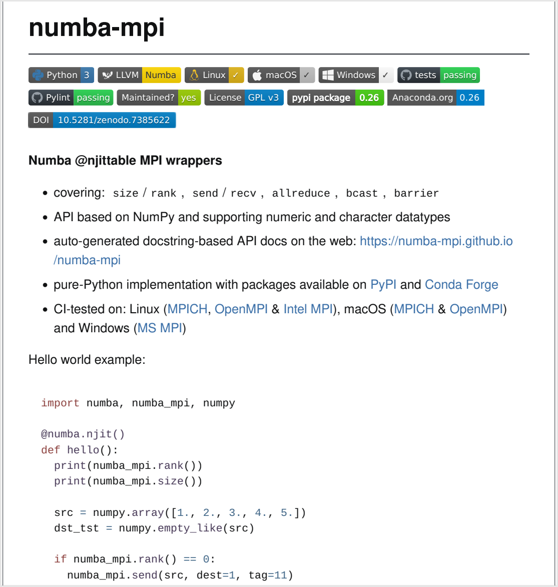
- let's DuckDuckGo it... nope, Qwant... nope, let's Google... **nothing!**
- let's ask on Numba[1] and mpi4py[2] forums (June 2020)...
  ⇝ *"You will not be able to use `mpi4py`'s `Cython` code, it was not designed for such low-level usage..."*
- **but it must be gluable!**
- 30 months, 120 commits and 50 PRs from 5 contributors later... (unplanned side project!)

[1] `https://github.com/numba/numba/issues/4115#issuecomment-642474009`

[2] `https://bitbucket.org/mpi4py/mpi4py/issues/164`

introducing: numba-mpi

https://github.com/numba-mpi/numba-mpi



numba-mpi

Python 3 | LLVM Numba | Linux ✓ | macOS ✓ | Windows ✓ | tests passing

Pylint passing | Maintained? yes | License GPL v3 | pypi package 0.26 | Anaconda.org 0.26

DOI 10.5281/zenodo.7385622

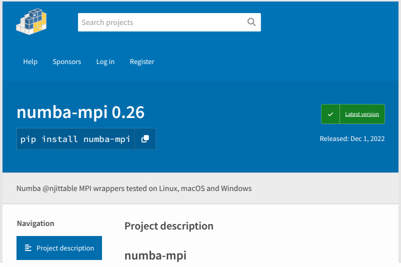
Numba @njittable MPI wrappers

- covering: `size` / `rank` , `send` / `recv` , `allreduce` , `bcast` , `barrier`
- API based on NumPy and supporting numeric and character datatypes
- auto-generated docstring-based API docs on the web: https://numba-mpi.github.io/numba-mpi
- pure-Python implementation with packages available on PyPI and Conda Forge
- CI-tested on: Linux (MPICH, OpenMPI & Intel MPI), macOS (MPICH & OpenMPI) and Windows (MS MPI)

Hello world example:

```
import numba, numba_mpi, numpy

@numba.njit()
def hello():
  print(numba_mpi.rank())
  print(numba_mpi.size())

  src = numpy.array([1., 2., 3., 4., 5.])
  dst_tst = numpy.empty_like(src)

  if numba_mpi.rank() == 0:
    numba_mpi.send(src, dest=1, tag=11)
```

https://pypi.org/project/numba-mpi



numba-mpi 0.26

```
pip install numba-mpi
```

✓ Latest version

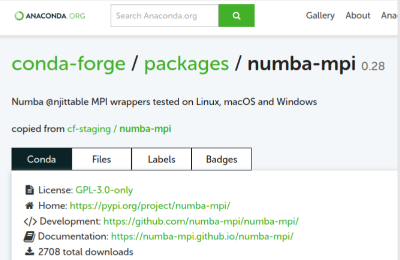
Released: Dec 1, 2022

Numba @njittable MPI wrappers tested on Linux, macOS and Windows

Project description

numba-mpi

https://anaconda.org/conda-forge/numba-mpi



conda-forge / packages / numba-mpi 0.28

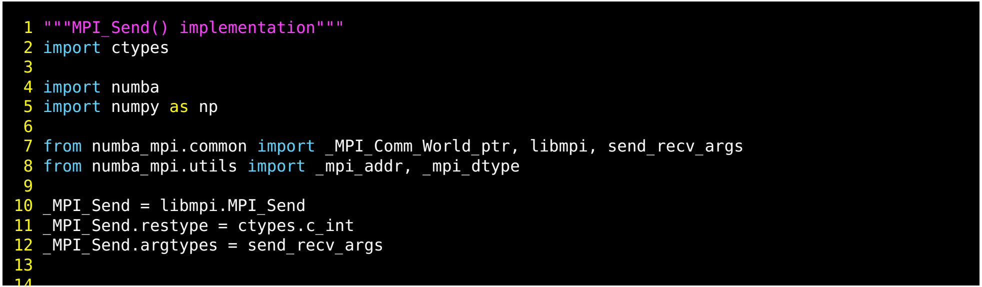
Numba @njittable MPI wrappers tested on Linux, macOS and Windows

copied from cf-staging / numba-mpi

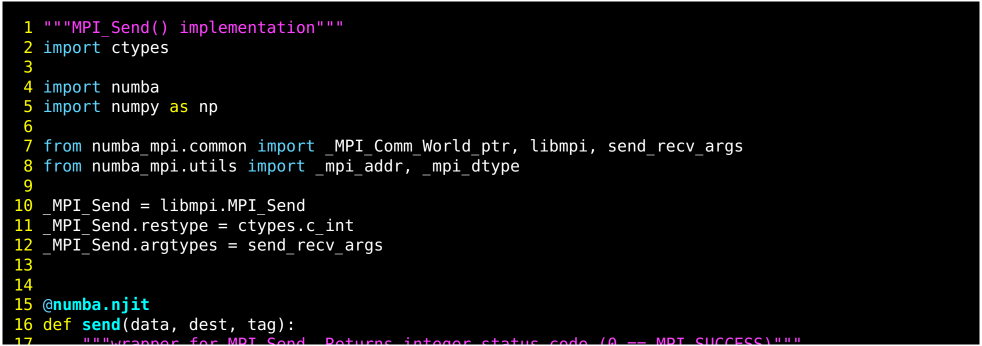
License: GPL-3.0-only
Home: https://pypi.org/project/numba-mpi/
Development: https://github.com/numba-mpi/numba-mpi/
Documentation: https://numba-mpi.github.io/numba-mpi/
2708 total downloads

numba-mpi: implementation

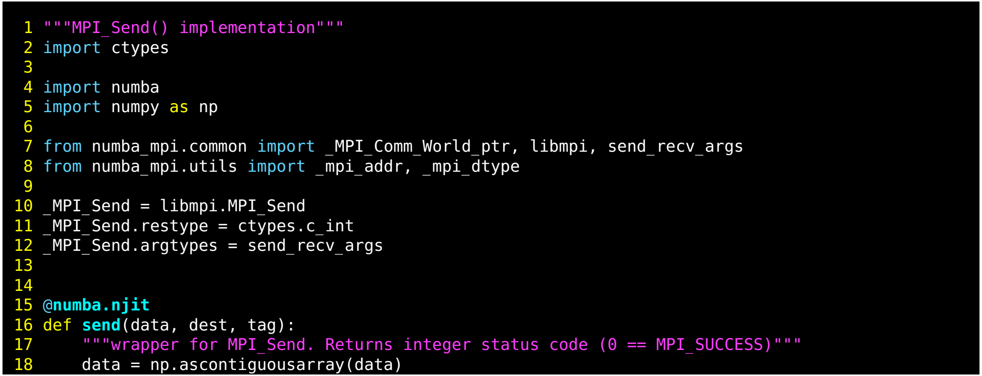
```python
"""MPI_Send() implementation"""
import ctypes

import numba
import numpy as np

```



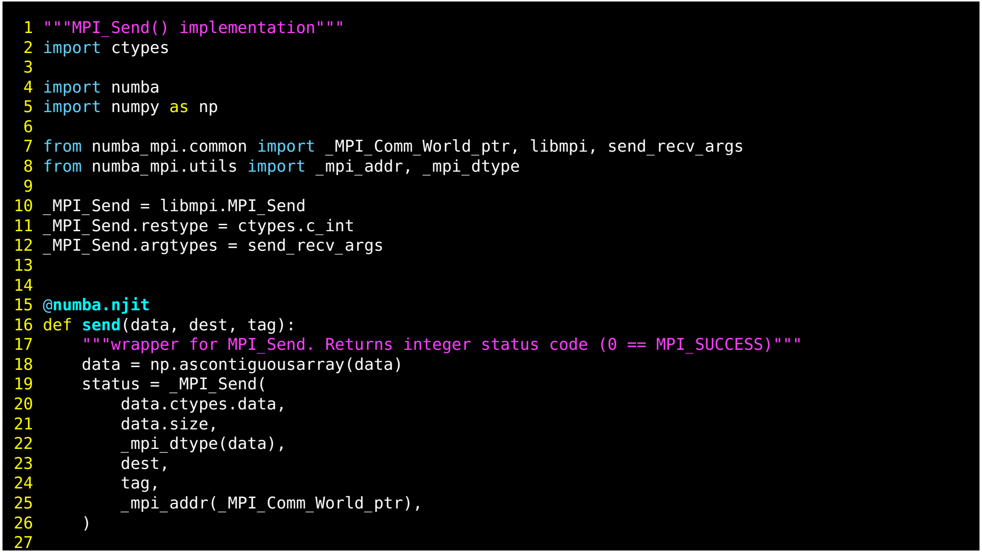
```python
"""MPI_Send() implementation"""
import ctypes

import numba
import numpy as np

from numba_mpi.common import _MPI_Comm_World_ptr, libmpi, send_recv_args
from numba_mpi.utils import _mpi_addr, _mpi_dtype

_MPI_Send = libmpi.MPI_Send
_MPI_Send.restype = ctypes.c_int
_MPI_Send.argtypes = send_recv_args

```



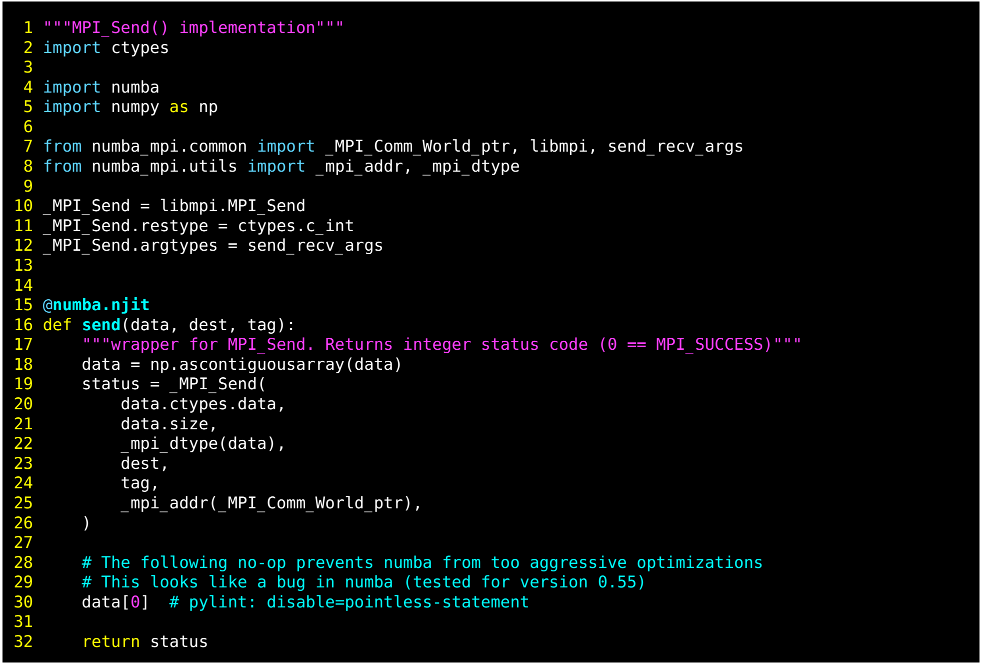
```python
"""MPI_Send() implementation"""
import ctypes

import numba
import numpy as np

from numba_mpi.common import _MPI_Comm_World_ptr, libmpi, send_recv_args
from numba_mpi.utils import _mpi_addr, _mpi_dtype

_MPI_Send = libmpi.MPI_Send
_MPI_Send.restype = ctypes.c_int
_MPI_Send.argtypes = send_recv_args


@numba.njit
def send(data, dest, tag):
    """wrapper for MPI_Send. Returns integer status code (0 == MPI_SUCCESS)"""
```



```python
"""MPI_Send() implementation"""
import ctypes

import numba
import numpy as np

from numba_mpi.common import _MPI_Comm_World_ptr, libmpi, send_recv_args
from numba_mpi.utils import _mpi_addr, _mpi_dtype

_MPI_Send = libmpi.MPI_Send
_MPI_Send.restype = ctypes.c_int
_MPI_Send.argtypes = send_recv_args


@numba.njit
def send(data, dest, tag):
    """wrapper for MPI_Send. Returns integer status code (0 == MPI_SUCCESS)"""
    data = np.ascontiguousarray(data)
```



```python
"""MPI_Send() implementation"""
import ctypes

import numba
import numpy as np

from numba_mpi.common import _MPI_Comm_World_ptr, libmpi, send_recv_args
from numba_mpi.utils import _mpi_addr, _mpi_dtype

_MPI_Send = libmpi.MPI_Send
_MPI_Send.restype = ctypes.c_int
_MPI_Send.argtypes = send_recv_args


@numba.njit
def send(data, dest, tag):
    """wrapper for MPI_Send. Returns integer status code (0 == MPI_SUCCESS)"""
    data = np.ascontiguousarray(data)
    status = _MPI_Send(
        data.ctypes.data,
        data.size,
        _mpi_dtype(data),
        dest,
        tag,
        _mpi_addr(_MPI_Comm_World_ptr),
    )

```



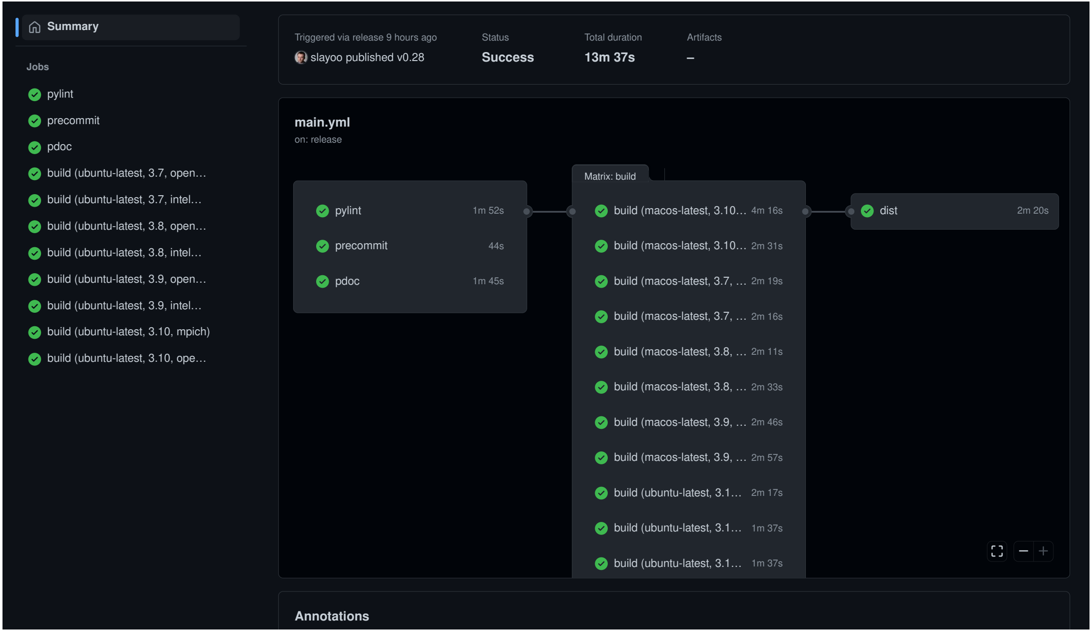
```python
"""MPI_Send() implementation"""
import ctypes

import numba
import numpy as np

from numba_mpi.common import _MPI_Comm_World_ptr, libmpi, send_recv_args
from numba_mpi.utils import _mpi_addr, _mpi_dtype

_MPI_Send = libmpi.MPI_Send
_MPI_Send.restype = ctypes.c_int
_MPI_Send.argtypes = send_recv_args


@numba.njit
def send(data, dest, tag):
    """wrapper for MPI_Send. Returns integer status code (0 == MPI_SUCCESS)"""
    data = np.ascontiguousarray(data)
    status = _MPI_Send(
        data.ctypes.data,
        data.size,
        _mpi_dtype(data),
        dest,
        tag,
        _mpi_addr(_MPI_Comm_World_ptr),
    )

    # The following no-op prevents numba from too aggressive optimizations
    # This looks like a bug in numba (tested for version 0.55)
    data[0]  # pylint: disable=pointless-statement

    return status
```

numba-mpi: hacks :(

… but there is also the `utils.py` …

```python
@numba.extending.overload(_mpi_addr)
def _mpi_addr_njit(ptr):
    def impl(ptr):
        return numba.carray(
            # pylint: disable-next=no-value-for-parameter
            _address_as_void_pointer(ptr),
            shape=(1,),
            dtype=np.intp,
        )[0]

    return impl


# https://stackoverflow.com/questions/61509903/how-to-pass-array-pointer-to-numba-function
@numba.extending.intrinsic
def _address_as_void_pointer(_, src):
    """returns a void pointer from a given memory address"""
    sig = types.voidptr(src)

    def codegen(__, builder, ___, args):
        return builder.inttoptr(args[0], cgutils.voidptr_t)

    return sig, codegen
```

numba-mpi: CI, OSes, MPI impls



Summary

Jobs

- ✓ pylint
- ✓ precommit
- ✓ pdoc
- ✓ build (ubuntu-latest, 3.7, open…
- ✓ build (ubuntu-latest, 3.7, intel…
- ✓ build (ubuntu-latest, 3.8, open…
- ✓ build (ubuntu-latest, 3.8, intel…
- ✓ build (ubuntu-latest, 3.9, open…
- ✓ build (ubuntu-latest, 3.9, intel…
- ✓ build (ubuntu-latest, 3.10, mpich)
- ✓ build (ubuntu-latest, 3.10, ope…

Triggered via release 9 hours ago	Status	Total duration	Artifacts
slayoo published v0.28	**Success**	**13m 37s**	–

main.yml

on: release

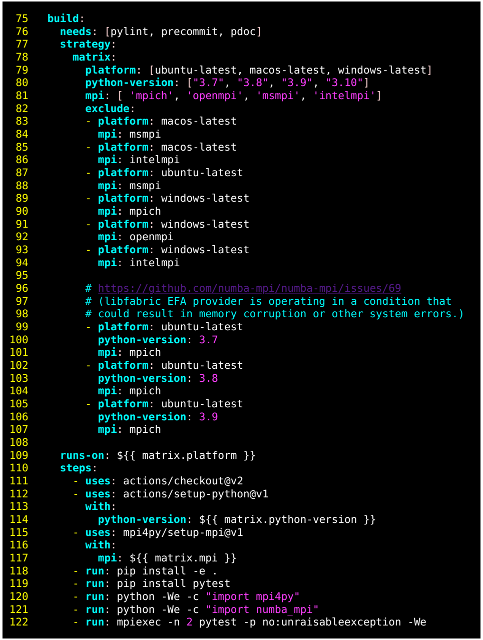
```
✓ pylint        1m 52s
✓ precommit        44s
✓ pdoc          1m 45s
```

Matrix: build

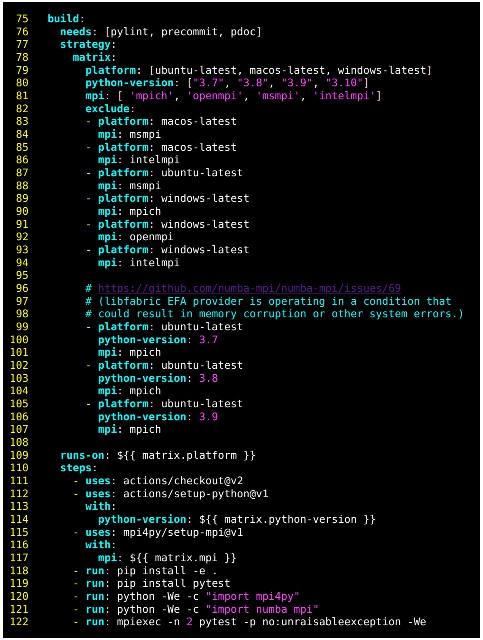
```
✓ build (macos-latest, 3.10…  4m 16s
✓ build (macos-latest, 3.10…  2m 31s
✓ build (macos-latest, 3.7, …  2m 19s
✓ build (macos-latest, 3.7, …  2m 16s
✓ build (macos-latest, 3.8, …  2m 11s
✓ build (macos-latest, 3.8, …  2m 33s
✓ build (macos-latest, 3.9, …  2m 46s
✓ build (macos-latest, 3.9, …  2m 57s
✓ build (ubuntu-latest, 3.1…  2m 17s
✓ build (ubuntu-latest, 3.1…  1m 37s
✓ build (ubuntu-latest, 3.1…  1m 37s
```

```
✓ dist  2m 20s
```

Annotations



```yaml
  build:
    needs: [pylint, precommit, pdoc]
    strategy:
      matrix:
        platform: [ubuntu-latest, macos-latest, windows-latest]
        python-version: ["3.7", "3.8", "3.9", "3.10"]
        mpi: [ 'mpich', 'openmpi', 'msmpi', 'intelmpi']
        exclude:
        - platform: macos-latest
          mpi: msmpi
        - platform: macos-latest
          mpi: intelmpi
        - platform: ubuntu-latest
          mpi: msmpi
        - platform: windows-latest
          mpi: mpich
        - platform: windows-latest
          mpi: openmpi
        - platform: windows-latest
          mpi: intelmpi

        # https://github.com/numba-mpi/numba-mpi/issues/69
        # (libfabric EFA provider is operating in a condition that
        # could result in memory corruption or other system errors.)
        - platform: ubuntu-latest
          python-version: 3.7
          mpi: mpich
        - platform: ubuntu-latest
          python-version: 3.8
          mpi: mpich
        - platform: ubuntu-latest
          python-version: 3.9
          mpi: mpich

    runs-on: ${{ matrix.platform }}
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-python@v1
        with:
          python-version: ${{ matrix.python-version }}
      - uses: mpi4py/setup-mpi@v1
        with:
          mpi: ${{ matrix.mpi }}
      - run: pip install -e .
      - run: pip install pytest
      - run: python -We -c "import mpi4py"
      - run: python -We -c "import numba_mpi"
      - run: mpiexec -n 2 pytest -p no:unraisableexception -We
```

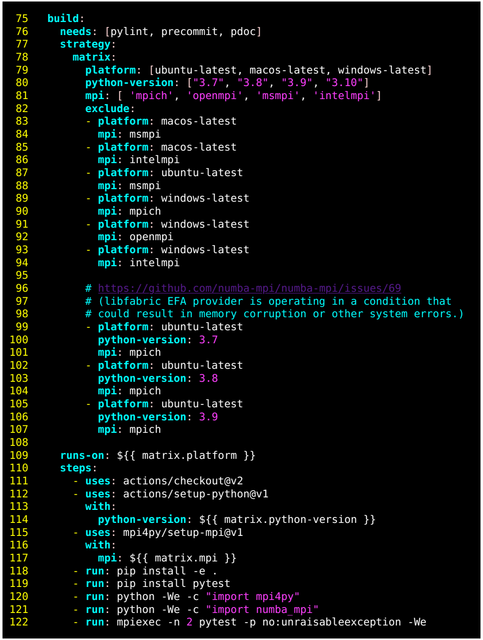


```yaml
  build:
    needs: [pylint, precommit, pdoc]
    strategy:
      matrix:
        platform: [ubuntu-latest, macos-latest, windows-latest]
        python-version: ["3.7", "3.8", "3.9", "3.10"]
        mpi: [ 'mpich', 'openmpi', 'msmpi', 'intelmpi']
        exclude:
        - platform: macos-latest
          mpi: msmpi
        - platform: macos-latest
          mpi: intelmpi
        - platform: ubuntu-latest
          mpi: msmpi
        - platform: windows-latest
          mpi: mpich
        - platform: windows-latest
          mpi: openmpi
        - platform: windows-latest
          mpi: intelmpi

        # https://github.com/numba-mpi/numba-mpi/issues/69
        # (libfabric EFA provider is operating in a condition that
        # could result in memory corruption or other system errors.)
        - platform: ubuntu-latest
          python-version: 3.7
          mpi: mpich
        - platform: ubuntu-latest
          python-version: 3.8
          mpi: mpich
        - platform: ubuntu-latest
          python-version: 3.9
          mpi: mpich

    runs-on: ${{ matrix.platform }}
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-python@v1
        with:
          python-version: ${{ matrix.python-version }}
      - uses: mpi4py/setup-mpi@v1
        with:
          mpi: ${{ matrix.mpi }}
      - run: pip install -e .
      - run: pip install pytest
      - run: python -We -c "import mpi4py"
      - run: python -We -c "import numba_mpi"
      - run: mpiexec -n 2 pytest -p no:unraisableexception -We
```

kudos to `mpi4py` team

for providing `setup-mpi` GitHub Action

this has saved us a lot of time!



```yaml
build:
  needs: [pylint, precommit, pdoc]
  strategy:
    matrix:
      platform: [ubuntu-latest, macos-latest, windows-latest]
      python-version: ["3.7", "3.8", "3.9", "3.10"]
      mpi: [ 'mpich', 'openmpi', 'msmpi', 'intelmpi']
      exclude:
      - platform: macos-latest
        mpi: msmpi
      - platform: macos-latest
        mpi: intelmpi
      - platform: ubuntu-latest
        mpi: msmpi
      - platform: windows-latest
        mpi: mpich
      - platform: windows-latest
        mpi: openmpi
      - platform: windows-latest
        mpi: intelmpi

      # https://github.com/numba-mpi/numba-mpi/issues/69
      # (libfabric EFA provider is operating in a condition that
      # could result in memory corruption or other system errors.)
      - platform: ubuntu-latest
        python-version: 3.7
        mpi: mpich
      - platform: ubuntu-latest
        python-version: 3.8
        mpi: mpich
      - platform: ubuntu-latest
        python-version: 3.9
        mpi: mpich

  runs-on: ${{ matrix.platform }}
  steps:
    - uses: actions/checkout@v2
    - uses: actions/setup-python@v1
      with:
        python-version: ${{ matrix.python-version }}
    - uses: mpi4py/setup-mpi@v1
      with:
        mpi: ${{ matrix.mpi }}
    - run: pip install -e .
    - run: pip install pytest
    - run: python -We -c "import mpi4py"
    - run: python -We -c "import numba_mpi"
    - run: mpiexec -n 2 pytest -p no:unraisableexception -We
```

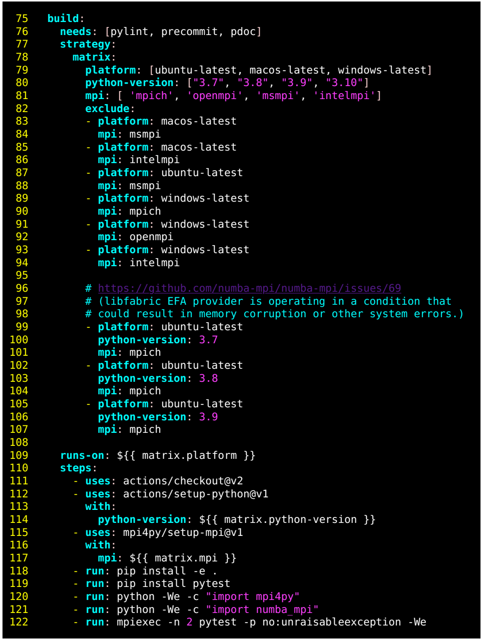
kudos to mpi4py team

for providing setup-mpi GitHub Action

this has saved us a lot of time!

OSes and MPI implementations tested

	Linux	macOS	Windows
OpenMPI	+	+	
MPICH	+	+	
IntelMPI	+		
MSMPI			+



```yaml
  build:
    needs: [pylint, precommit, pdoc]
    strategy:
      matrix:
        platform: [ubuntu-latest, macos-latest, windows-latest]
        python-version: ["3.7", "3.8", "3.9", "3.10"]
        mpi: [ 'mpich', 'openmpi', 'msmpi', 'intelmpi']
        exclude:
        - platform: macos-latest
          mpi: msmpi
        - platform: macos-latest
          mpi: intelmpi
        - platform: ubuntu-latest
          mpi: msmpi
        - platform: windows-latest
          mpi: mpich
        - platform: windows-latest
          mpi: openmpi
        - platform: windows-latest
          mpi: intelmpi

        # https://github.com/numba-mpi/numba-mpi/issues/69
        # (libfabric EFA provider is operating in a condition that
        # could result in memory corruption or other system errors.)
        - platform: ubuntu-latest
          python-version: 3.7
          mpi: mpich
        - platform: ubuntu-latest
          python-version: 3.8
          mpi: mpich
        - platform: ubuntu-latest
          python-version: 3.9
          mpi: mpich

    runs-on: ${{ matrix.platform }}
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-python@v1
        with:
          python-version: ${{ matrix.python-version }}
      - uses: mpi4py/setup-mpi@v1
        with:
          mpi: ${{ matrix.mpi }}
      - run: pip install -e .
      - run: pip install pytest
      - run: python -We -c "import mpi4py"
      - run: python -We -c "import numba_mpi"
      - run: mpiexec -n 2 pytest -p no:unraisableexception -We
```

kudos to mpi4py team

for providing setup-mpi GitHub Action

this has saved us a lot of time!

OSes and MPI implementations tested

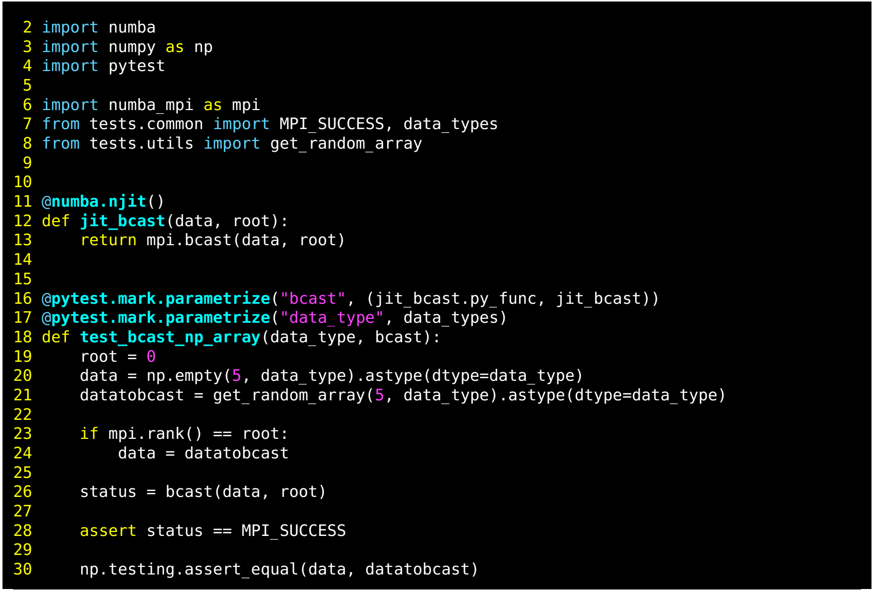
	Linux	macOS	Windows
OpenMPI	+	+	
MPICH	+	+	
IntelMPI	+		
MSMPI			+

caveat

MPICH v4 fails on Ubuntu for Python <3.10

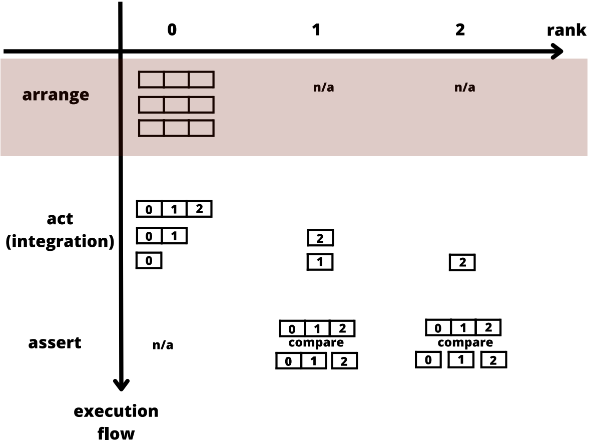
”*libfabric EFA provider is operating in a condition that could result in memory corruption*” ⇝ SIGABRT

numba-mpi: sample unit test

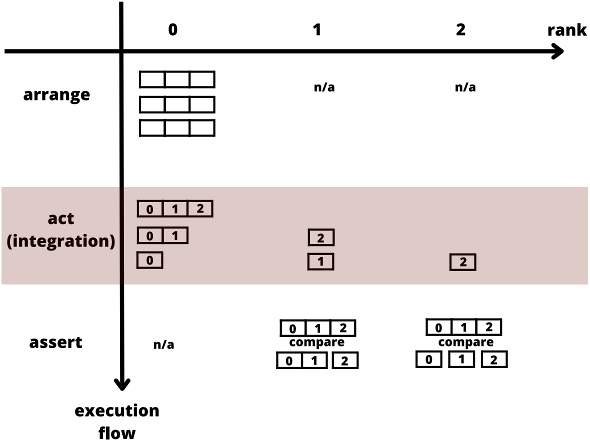


```python
import numba
import numpy as np
import pytest

import numba_mpi as mpi
from tests.common import MPI_SUCCESS, data_types
from tests.utils import get_random_array


@numba.njit()
def jit_bcast(data, root):
    return mpi.bcast(data, root)


@pytest.mark.parametrize("bcast", (jit_bcast.py_func, jit_bcast))
@pytest.mark.parametrize("data_type", data_types)
def test_bcast_np_array(data_type, bcast):
    root = 0
    data = np.empty(5, data_type).astype(dtype=data_type)
    datatobcast = get_random_array(5, data_type).astype(dtype=data_type)

    if mpi.rank() == root:
        data = datatobcast

    status = bcast(data, root)

    assert status == MPI_SUCCESS

    np.testing.assert_equal(data, datatobcast)
```

sample integration test scheme



(https://github.com/atmos-cloud-sim-uj/PySuperDropletLES/blob/main/tests/test_2d.py)



0
1
2
rank

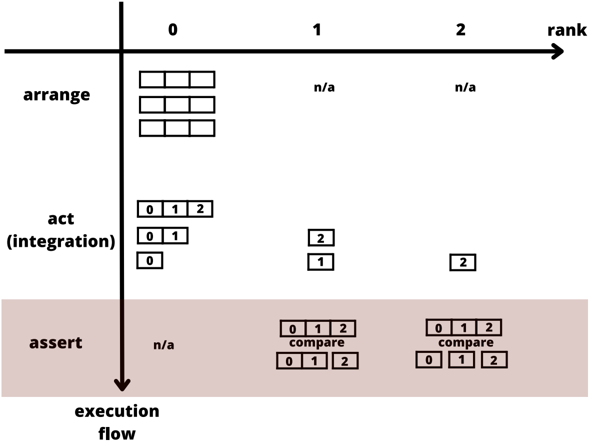
arrange
n/a
n/a

act
(integration)
0 1 2
0 1
0
2
1
2

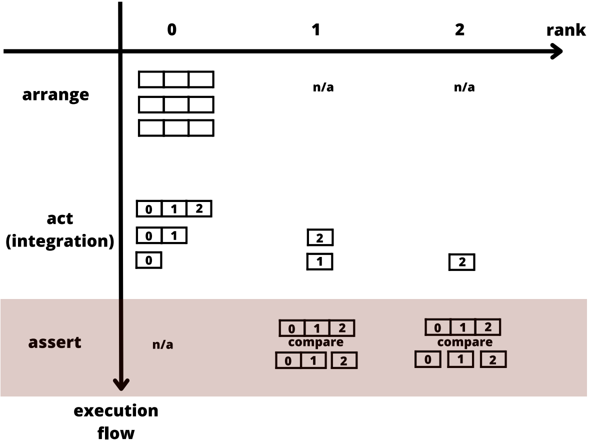
assert
n/a
0 1 2
compare
0 1 2
0 1 2
compare
0 1 2

execution
flow

(https://github.com/atmos-cloud-sim-uj/PySuperDropletLES/blob/main/tests/test_2d.py)



(https://github.com/atmos-cloud-sim-uj/PySuperDropletLES/blob/main/tests/test_2d.py)



(https://github.com/atmos-cloud-sim-uj/PySuperDropletLES/blob/main/tests/test_2d.py)

caveat

using HDF5/MPI-IO (h5py) for concurrent file access from different MPI ranks

... implies insurmountable trouble setting up CI test env on Windows (help welcome!)

py-pde: independent use case

py-pde is a Python package for solving partial differential equations (PDEs).

https://py-pde.readthedocs.io

py-pde is a Python package for solving partial differential equations (PDEs).

Focus:

- Finite differencing and simple grids
- PDEs defined by mathematical expressions (supplied as strings)

py-pde

https://py-pde.readthedocs.io

py-pde is a Python package for solving partial differential equations (PDEs).

Focus:

- Finite differencing and simple grids
- PDEs defined by mathematical expressions (supplied as strings)

https://py-pde.readthedocs.io

Solution strategy:

- Partition the grid onto different nodes using numba-mpi
- On each node, parse expressions using sympy and compile the result using numba
- Iterate the PDE, exchanging boundary information between nodes using numba-mpi

take-home messages

Python:

- common mismatch: **language vs. ecosystem** (e.g., arrays, number-crunching)
- has a range of **gluable HPC solutions** (JIT, GPU, multi-threading, MPI, ...)

Python:

- common mismatch: **language vs. ecosystem** (e.g., arrays, number-crunching)
- has a range of **gluable HPC solutions** (JIT, GPU, multi-threading, MPI, ...)

python™ + Numba + `MPI` = 👍

numba-mpi:

- enables one to **glue** `MPI` **with** `LLVM` **JIT-compiled Python** code
- **CI-tested** on Linux, macOS, Windows; `MPICH`, `OpenMPI`, `Intel MPI`, & `MS MPI`
- developed aiming for **100% unit test coverage** (of the wrapping logic)
- already a dependency of two PDE-solver projects: `py-pde` & `PySuperDropletLES`

numba-mpi sites:

- github.com/numba-mpi (contributors: slayoo, xann16, david-zwicker, Delcior, abulenok)
- pypi.org/p/numba-mpi
- anaconda.org/conda-forge/numba-mpi

numba-mpi sites:

- github.com/numba-mpi (contributors: slayoo, xann16, david-zwicker, Delcior, abulenok)
- pypi.org/p/numba-mpi
- anaconda.org/conda-forge/numba-mpi

contributions welcome:

- packaging `h5py` **for Windows** (`HDF5`) with support for `MPI-IO`
- `MPICH` $\geqslant$ 4.0 with Python $<$ 3.10 on Linux (`libfabric` **EFA provider issue**)
- numba-mpi contribs:
  - **logo**
  - adding support for other functions from the MPI API
  - **droping dependency on mpi4py**
  - benchmarking performance

numba-mpi sites:

- github.com/numba-mpi (contributors: slayoo, xann16, david-zwicker, Delcior, abulenok)
- pypi.org/p/numba-mpi
- anaconda.org/conda-forge/numba-mpi

contributions welcome:

- packaging `h5py` **for Windows** (`HDF5`) with support for `MPI-IO`
- `MPICH` $\geqslant$ 4.0 with Python $<$ 3.10 on Linux (`libfabric` **EFA provider issue**)
- numba-mpi contribs:
  - **logo**
  - adding support for other functions from the MPI API
  - **droping dependency on mpi4py**
  - benchmarking performance

404

logo not found

funding:

NATIONAL SCIENCE CENTRE
POLAND

(grant no. 2020/39/D/ST10/01220)