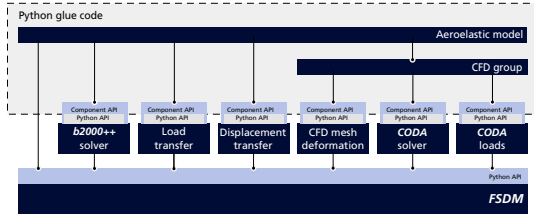
PYTEST-ISOLATE-MPI

Towards Modern Testing of MPI-Parallel HPC Applications

Motivation

Testing of toolchains for multi-disciplinary simulations:

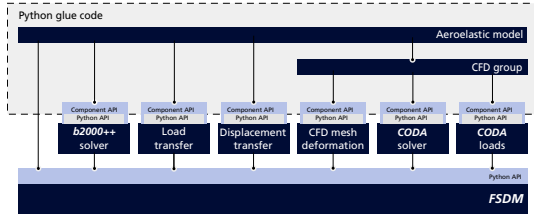
- each discipline (CFD, CSM, . . .) backed by dedicated solver
  - typically written in C++ or Fortran
  - tested with frameworks suited for these languages
- coupled with Python glue code
  - calls disciplinary solvers by their Python API
  - still MPI-parallel for efficient HPC coupling



Motivation

Testing of toolchains for multi-disciplinary simulations:

- each discipline (CFD, CSM, . . .) backed by dedicated solver
  - typically written in C++ or Fortran
  - tested with frameworks suited for these languages
- coupled with Python glue code
  - calls disciplinary solvers by their Python API
  - still MPI-parallel for efficient HPC coupling



Existing Python test frameworks lack provisions for MPI-parallel software, e. g.

- `MPI_Abort` or segfaults abort the test suite including the framework
- MPI deadlocks keep the from tests finishing at all
- no concept of per-process test results

Outline

1. Considered Software
   - Pytest
   - `pytest-mpi`
   - `pytest-forked` / `pytest-isolate`
   - `testflo`

2. The `pytest-isolate-mpi` Plug-in
   - Implementation
   - Usage
   - Limitations

CONSIDERED SOFTWARE

Pytest

State-of-the-art testing framework for Python:

```python
def inc(x):
    return x + 1

def test_answer():
    assert inc(3) == 5
```

Yields:

```
    def test_answer():
>       assert inc(3) == 5
E       assert 4 == 5
E        +  where 4 = inc(3)

test_sample.py:5: AssertionError
```

Pytest (II)

Powerful and idiomatic testing capabilities:

- `assert`-rewriting for failing test introspection
- flexible test parametrization and fixture management
- usable for unit, functional and integration tests

Sadly, no support for MPI

- best option: run Pytest in MPI
- prone to crashes and deadlocks
- one test report per MPI process

pytest-mpi

Plug-in for Pytest to assist with running under MPI:

- Automatic selection of MPI tests based on environment
- MPI-aware fixtures for temporary files

No provisions for handling `MPI_Abort` or deadlocks

```python
import pytest

@pytest.mark.mpi(min_size=2)
def test_size():
    from mpi4py import MPI
    comm = MPI.COMM_WORLD
    assert comm.size >= 2
```

pytest-forked / pytest-isolate

Pytest plug-ins to isolate tests via subprocesses:

- crashing tests can be handled gracefully
- deadlocks no longer stall the test suite via process timeouts

But: no direct MPI support

Very promising candidate:

- MPI support, process isolation, and timeouts
- very fast execution
  - `testflo` can launch as many MPI processes as the machine fits
  - tests are run concurrently to maximize throughput

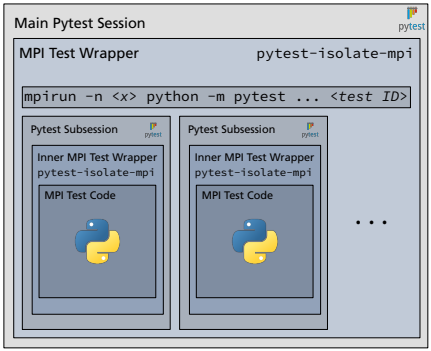
But: build on Python's `unittest` module

- cannot be considered *modern* anymore

```python
from unittest import TestCase

class MPITestCase(TestCase):

    N_PROCS = 2

    def test_comm_size(self):
        self.assertEqual(self.comm.size,
                         self.N_PROCS)
```

THE PYTEST-ISOLATE-MPI PLUG-IN

Implementation



1. Pytest collects tests as usual
2. MPI tests are intercepted by the plugin
   - Launch Pytest subsession within `mpirun` for test
   - Serialize test result to a per-process file
   - Main Pytest session collects result files
3. Pytest presents test results to the user

Usage – Writing MPI-Parallel Tests

Annotate MPI-parallel tests with MPI `mpi` marker:

```python
import pytest

@pytest.mark.mpi(ranks=2)
def test_failing_rank(mpi_ranks, comm):
    assert comm.rank != 0
```

```
============================= test session starts ==============================
collected 1 item

examples/test_one_failing_rank.py F.                                     [100%]

=================================== FAILURES ===================================
_____________________________ test_failing_rank[2][rank=0] _____________________________

mpi_ranks = 2, comm = <mpi4py.MPI.Intracomm object at 0x75c281685f50>

    @pytest.mark.mpi(ranks=2)
    def test_failing_rank(mpi_ranks, comm):
>       assert comm.rank != 0
E       assert 0 != 0
E        +  where 0 = <mpi4py.MPI.Intracomm object at 0x75c281685f50>.rank

examples/test_one_failing_rank.py:5: AssertionError
=========================== short test summary info ============================
FAILED examples/test_one_failing_rank.py::test_failing_rank[2][rank=0] - assert 0 != 0
============================== 1 failed, 1 passed in 3.46s ==============================
```

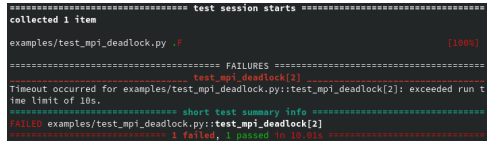
Usage – Crashing Tests

```python
import os
import pytest

@pytest.mark.mpi(ranks=2)
def test_aborting_rank(mpi_ranks, comm):
    if comm.rank == 0:
        os._exit(127)
```

```
============================= test session starts ==============================
collected 1 item

examples/test_one_aborting_rank.py .F                                    [100%]

=================================== FAILURES ===================================
_____________________________ test_aborting_rank[2] _____________________________
At least one MPI process has exited prematurely.
----------------------------- Captured stdout ------------------------------
============================= test session starts ==============================
platform linux -- Python 3.10.12, pytest-7.4.4, pluggy-1.5.0
============================= test session starts ==============================
platform linux -- Python 3.10.12, pytest-7.4.4, pluggy-1.5.0
rootdir: /home/gott_se/repos/pytest-isolate-mpi
plugins: cov-5.0.0, isolate-mpi-0.1.dev89+ga1f291a.d20240916
collecting ... rootdir: /home/gott_se/repos/pytest-isolate-mpi
plugins: cov-5.0.0, isolate-mpi-0.1.dev89+ga1f291a.d20240916
collecting ...
collected 1 item

collected 1 item

examples/test_one_aborting_rank.py
examples/test_one_aborting_rank.py .                                     [100%]

============================== 1 passed in 0.22s ===============================
----------------------------- Captured stderr ------------------------------
--------------------------------------------------------------------------
Primary job  terminated normally, but 1 process returned
a non-zero exit code. Per user-direction, the job has been aborted.
--------------------------------------------------------------------------
--------------------------------------------------------------------------
mpirun detected that one or more processes exited with non-zero status, thus causing
the job to be terminated. The first process to do so was:

  Process name: [[53018,1],0]
  Exit code:    127
--------------------------------------------------------------------------
=========================== short test summary info ============================
FAILED examples/test_one_aborting_rank.py::test_aborting_rank[2]
========================== 1 failed, 1 passed in 2.54s =========================
```

Usage – Tests with MPI Deadlocks

```python
import pytest


@pytest.mark.mpi(ranks=2, timeout=10,
                 unit="s")
def test_mpi_deadlock(mpi_ranks, comm):
    if comm.rank == 0:
        comm.Barrier()
```



Limitations

Reports for crashed MPI tests

- crashed processes will not write per-process test report
- only output of `mpirun` available

Limitations

Reports for crashed MPI tests

- crashed processes will not write per-process test report
- only output of `mpirun` available

Limited support for reusing fixtures between tests with fixture scopes

- all tests run in a dedicated Pytest session
- so all scopes (`session`, `package`, `class`, `function`) behave the same
- `pytest-isolate-mpi` can share `session` fixtures between MPI tests

Future Plans

Employ in DLR simulation software stacks

- e. g. for the system tests for CFD solver CODA

Future Plans

Employ in DLR simulation software stacks

- e. g. for the system tests for CFD solver CODA

Integrate with HPC Environments

- Allow to swap out `mpirun` with `sbatch`

Conclusion

`pytest-isolate-mpi` enables modern testing for MPI-parallel Python software

Try it:

```
pip install pytest-isolate-mpi
```

More information at:

https://pytest-isolate-mpi.readthedocs.io/