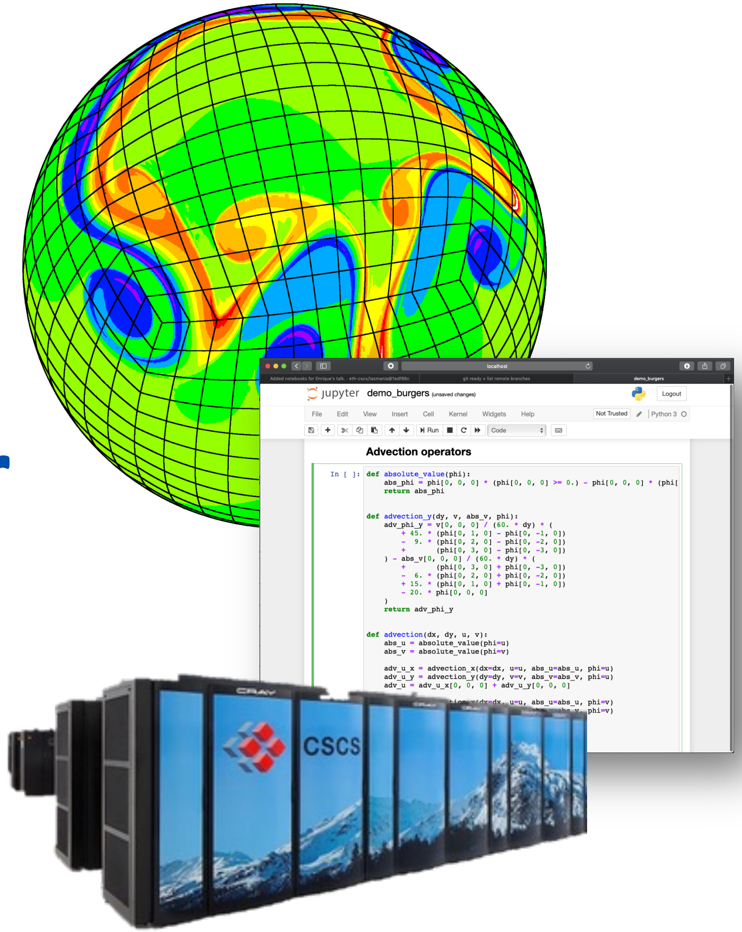
High Performance Computing for Weather and Climate (HPC4WC)

Content: Pace: A model in Python
Lecturer: Oliver Fuhrer
Block course 701-1270-00L
Summer 2024



Team

Oliver
Fuhrer

Johann
Dahm

Florian
Deconinck

Oliver
Elbert

Jeremy
McGibbon

Tobias
Wicky

Elynn Wu

Collaborators

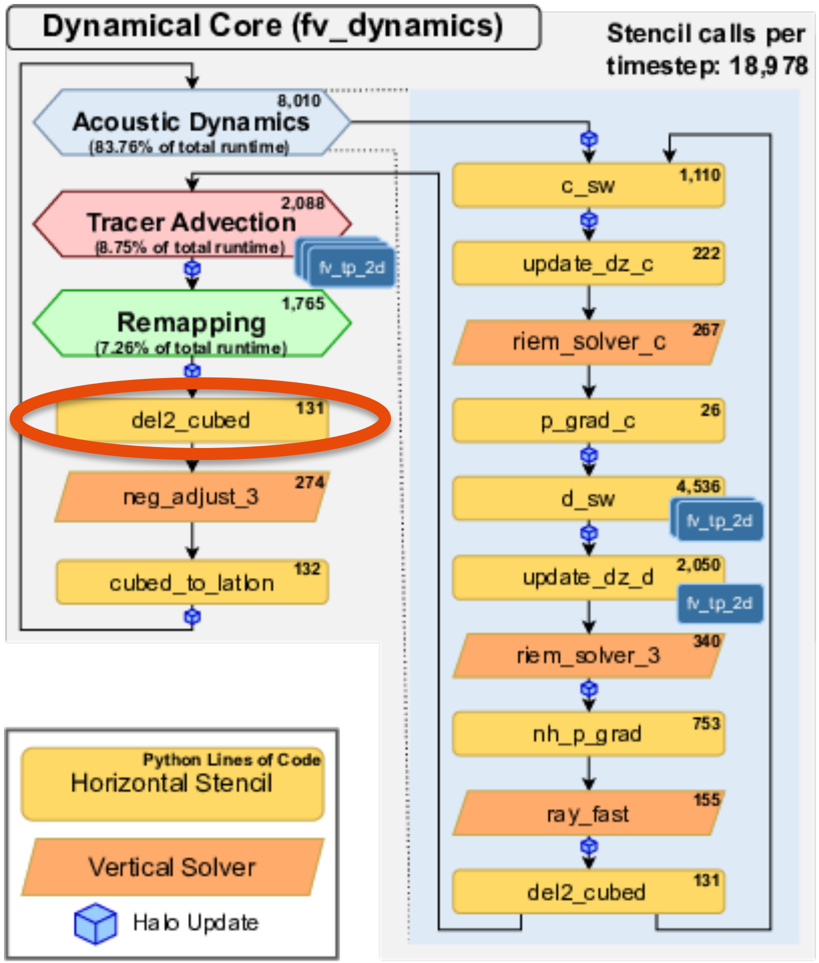
GFDL
NASA
CSCS
MeteoSwiss
ETH zürich
UNIVERSITY of WASHINGTON

Pace: Python based FV3

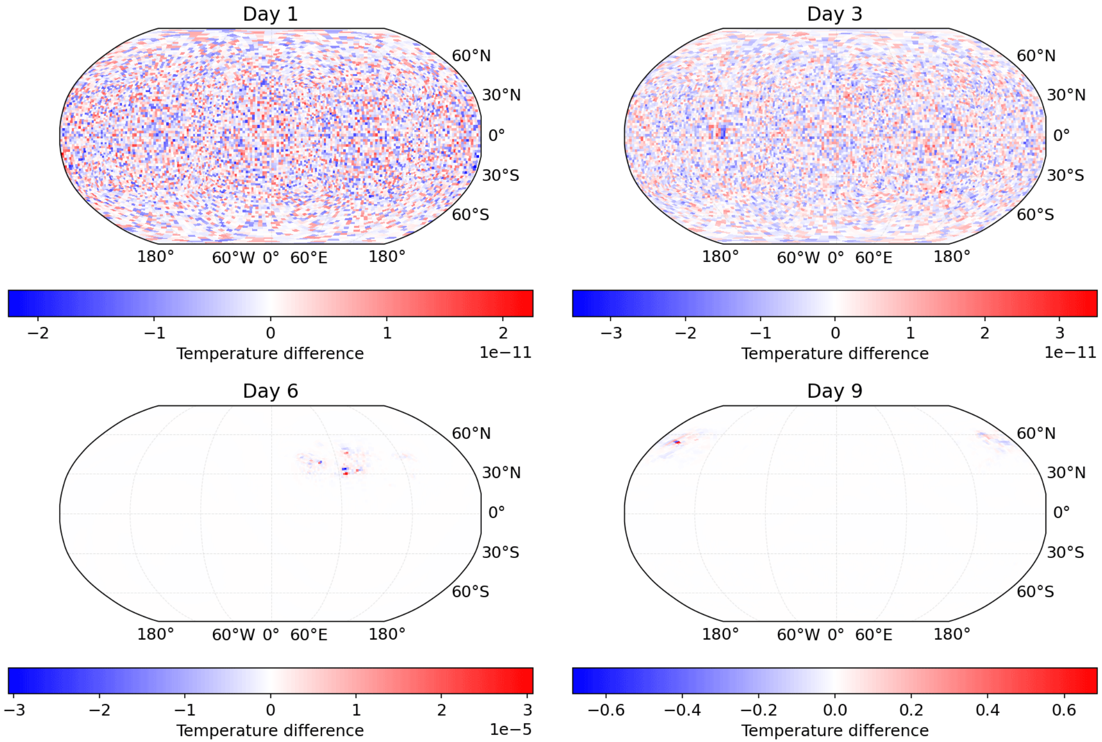
GFDL Finite-Volume Cubed-Sphere Dynamical Core (FV3)

Finite volume transport on a cubed sphere grid

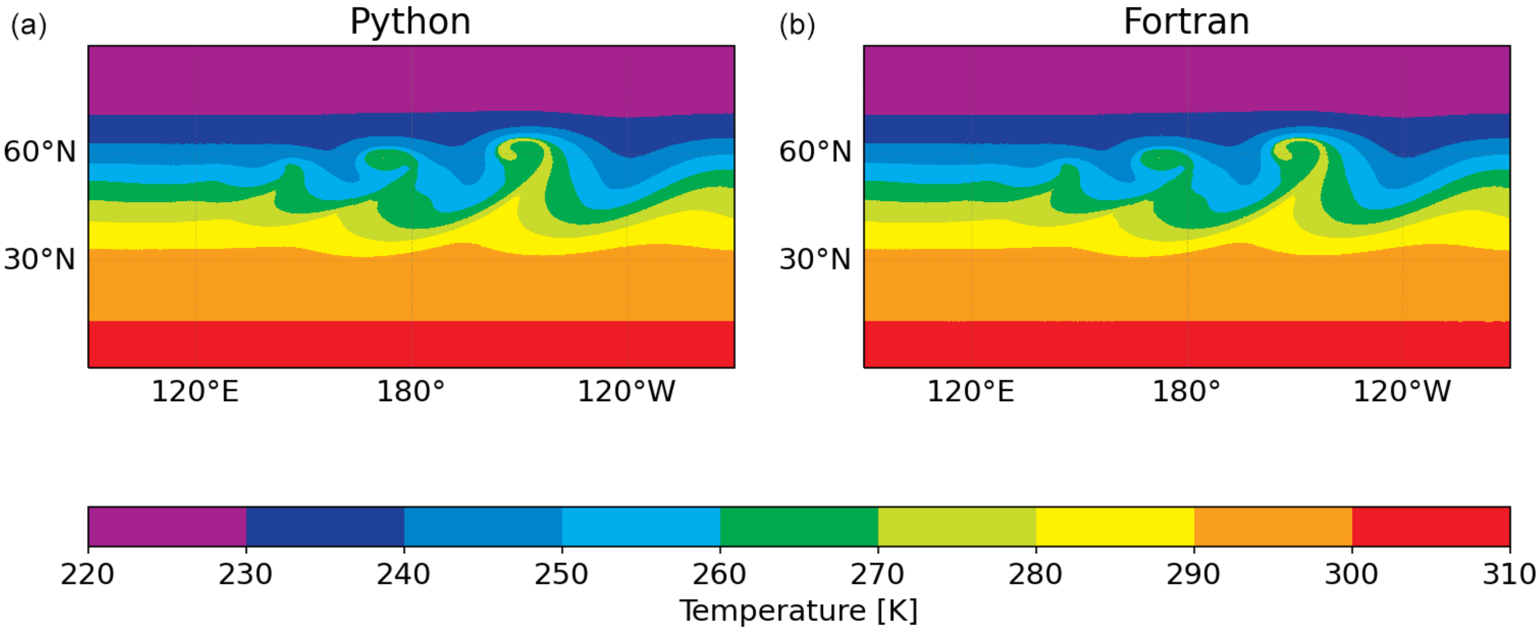
- Integrated into several models, including
  - Operational weather models (Global Forecast System)
  - Next Generation Global Prediction System



Validation


Day 1
Day 3
Day 6
Day 9
60°N
30°N
0°
30°S
60°S
180°
60°W 0° 60°E
180°
−2 −1 0 1 2
Temperature difference
1e−11
−3 −2 −1 0 1 2 3
Temperature difference
1e−11
−3 −2 −1 0 1 2 3
Temperature difference
1e−5
−0.6 −0.4 −0.2 0.0 0.2 0.4 0.6
Temperature difference

Validation



(a)
Python
60°N
30°N
120°E
180°
120°W
(b)
Fortran
60°N
30°N
120°E
180°
120°W
220
230
240
250
260
270
280
290
300
310
Temperature [K]

Physical Parametrizations

	Microphysics	PBL & Turbulence	Sea-Ice	Shallow Convection	LSM	Radiation
Authors	 Mikael	 Chris Kung (NASA)	 Vera  Nadja  Nicolai	 Mikael  Chenwei  Langwen	 Safira	 Andrew
Scheme	**GFDL Cloud Microphysics Scheme**	GFS scale-aware EDMF PBL and Free Atmospheric Turbulence Scheme	GFS Sea Ice Scheme	**GFS SAS-based Mass-Flux Scheme for Shallow convection**	GFS Noah Land Surface Model	GFS RRTMG

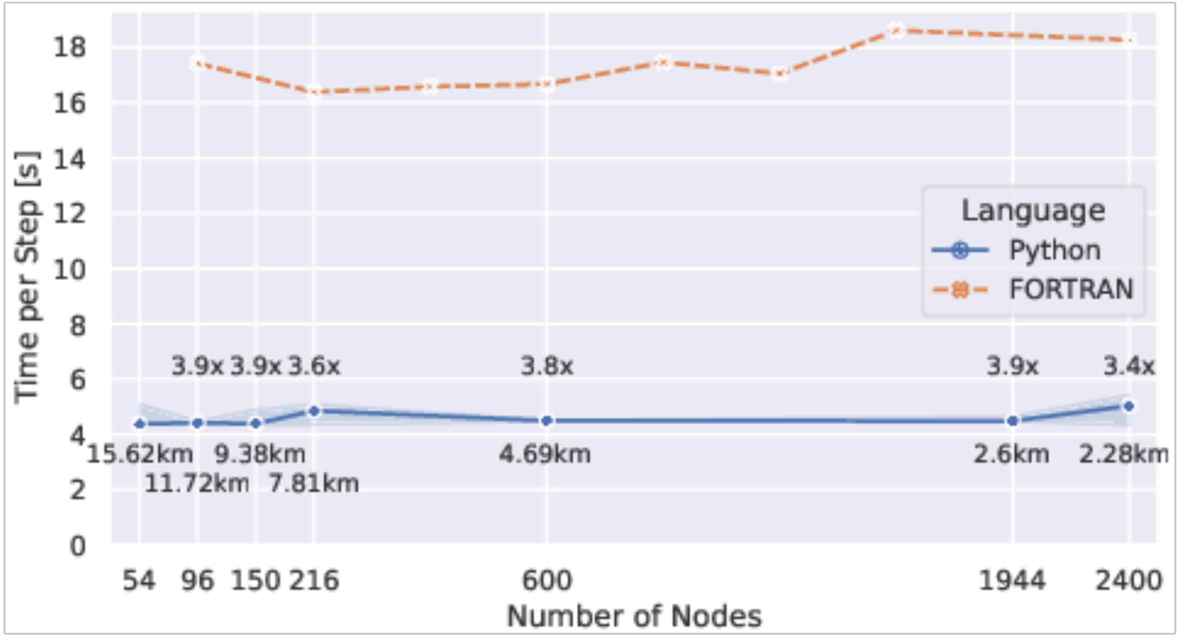
The Pace Model

Full program optimization

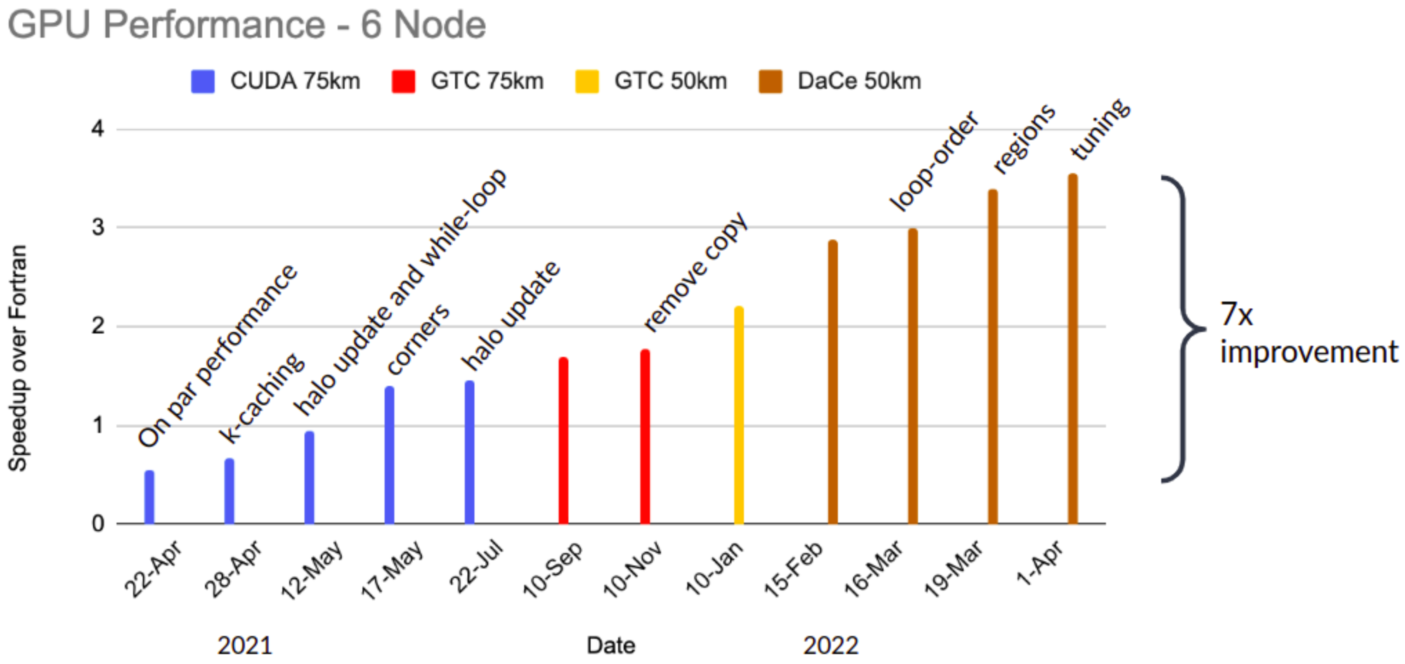
DSL coverage of all main numerical computation

Custom code for halo updates

New DSL concepts for FV3-specific motifs



The Pace Model



Why did you torture us for 4 days?

DSL still uses these concepts under the hood

```
def visit_Stencil(self, node: oir.Stencil, **kwargs: Any) -> oir.Stencil:
    write_before_read_tmps = {
        symbol
        for symbol, value in kwargs["symtable"].items()
        if isinstance(value, oir.Temporary)
    }
    horizontal_executions = node.iter_tree().if_isinstance(oir.HorizontalExecution)

    for horizontal_execution in horizontal_executions:
        accesses = AccessCollector.apply(horizontal_execution)
        offsets = accesses.offsets()
        ordered_accesses = accesses.ordered_accesses()

        def write_before_read(tmp: str) -> bool:
            if tmp not in offsets:
                return True
            if offsets[tmp] != {(0, 0, 0)}:
                return False
            return next(
                o.is_write and o.horizontal_mask is None
                for o in ordered_accesses
                if o.field == tmp
            )

        write_before_read_tmps = {
            tmp for tmp in write_before_read_tmps if write_before_read(tmp)
        }

    return super().visit_Stencil(node, tmps_to_replace=write_before_read_tmps, **kwargs)
```

Why did you torture us for 4 days?

There are things that we still need to do manually:

E.g. Halo updates

```
    )

# Post recv MPI order
with self._timer.clock("Irecv"):
    self._recv_requests = []
    for to_rank, transformer in self._transformers.items():
        self._recv_requests.append(
            self._comm.comm.Irecv(
                transformer.get_unpack_buffer().array,
                source=to_rank,
                tag=self._tag,
            )
        )

# Pack quantities halo points data into buffers      FlorianDeco
with self._timer.clock("pack"):
    for transformer in self._transformers.values():
        transformer.async_pack(quantities_x, quantities_y)
```

What does Python bring us?

```
class Physics:

...
prepare_microphysics(physics_state)
microph_state = physics_state.microphysics
microphysics(microph_state)
```

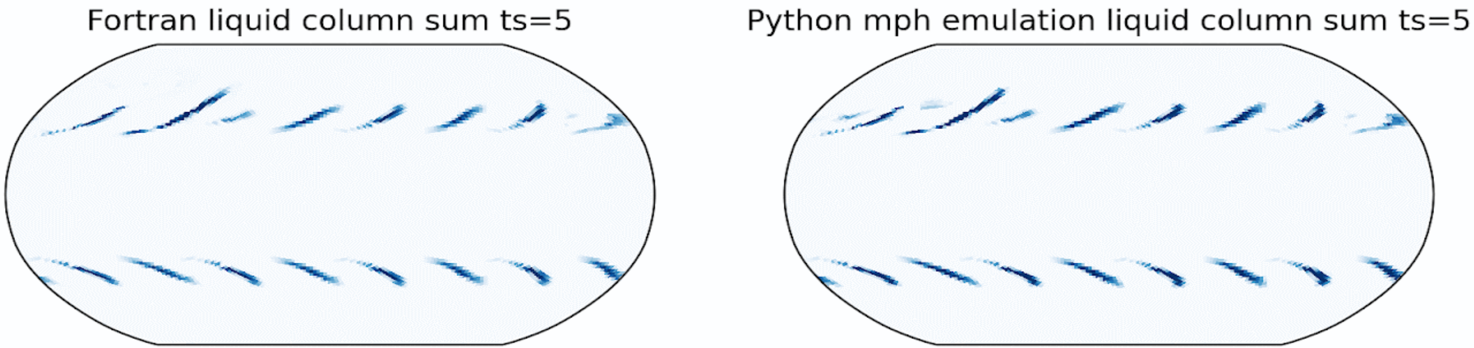
GT4Py stencil-based

```
emulation_model = tf.keras.models.load_model("model.tf")

emulation_dict = prepare_emulation_data(physics_state.microphysics)
predictions = emulation_model(emulation_dict)
model_outputs = unpack_predictions(predictions, emulation_model.output_names, …)
```

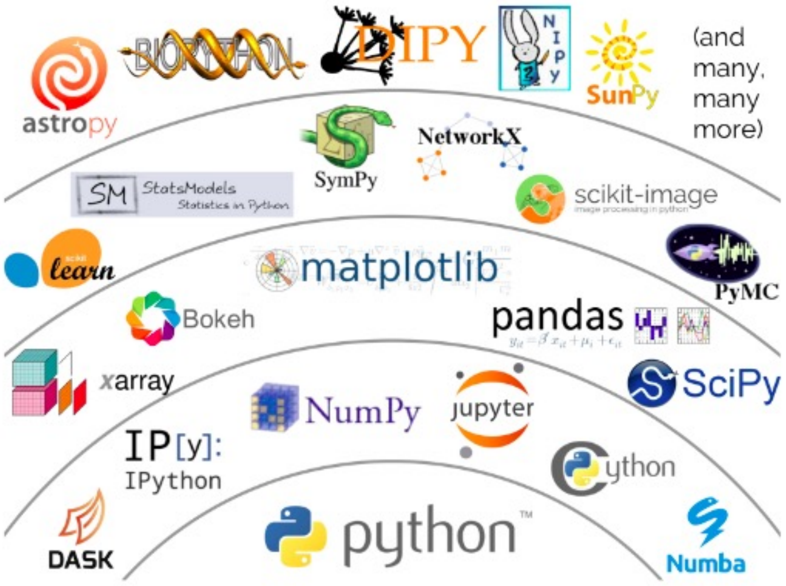
ML-based microphysics

ML-based Microphysics



Why else is Python awesome?

The rich python ecosystem is valuable – new options for development



Credit: Jake VanderPlas, "The Unexpected Effectiveness of Python in Science", PyCon 2017

Why else is Python awesome?

Testing is WAY easier!

```python
def test_temporaries_are_deterministic():
    """
    This is a precursor test to the next one, ensuring that two
    identically-initialized dycores called on identically-initialized
    states produce identical temporaries.

    This will fail if there is non-determinism in the initialization,
    for example from using `empty` instead of `zeros` to initialize data.
    """
    dycore1, state1, timer1 = setup_dycore()
    dycore2, state2, timer2 = setup_dycore()

    dycore1.step_dynamics(state1, timer1)
    first_temporaries = copy_temporaries(dycore1, max_depth=10)
    assert len(first_temporaries) > 0
    dycore2.step_dynamics(state2, timer2)
    second_temporaries = copy_temporaries(dycore2, max_depth=10)
    assert_same_temporaries(second_temporaries, first_temporaries)
```

Why else is Python awesome?

You!