

Enabling performance portability for FV₃ / xSHIELD using a **Python-based** domain-specific language

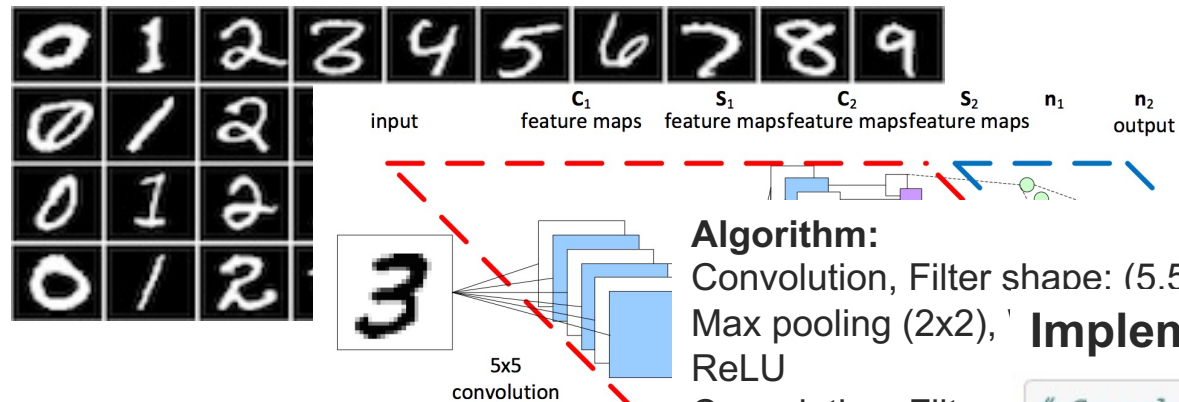
Oliver Fuhrer, Johann P. S. Dahm, Eddie C. Davis, Florian Deconinck, Oliver D. Elbert, Rhea C. George, Christopher Kung, Jeremy McGibbon, Andrew Pauling, Tobias F. Wicky, Elynn Wu

Partners: NOAA/GFDL and NASA/GMAO

September 3, 2021
ESCAPE-2 Dissemination Workshop



Machine Learning Scientist



Algorithm:

Convolution, Filter shape: (5.5.6). Stride=1. Padding='SAME'

Max pooling (2x2),

ReLU

Convolution, Filter shape: (5.5.6). Stride=1. Padding='SAME'

Max pooling (2x2),

ReLU

Fully Connected Layer

ReLU

Fully Connected Layer

Softmax

Implementation (Python + Tensorflow):

```
# Convolutional Layer 1
layer_conv1, weights_conv1 = new_conv_layer(input=x_image, num_input_channels=1, filters=6, name="conv1")

# Pooling Layer 1
layer_pool1 = new_pool_layer(layer_conv1, name="pool1")

# ReLU layer 1
layer_relu1 = new_relu_layer(layer_pool1, name="relu1")

# Convolutional Layer 2
layer_conv2, weights_conv2 = new_conv_layer(input=layer_relu1, num_input_channels=6, filters=16, name="conv2")

# Pooling Layer 2
layer_pool2 = new_pool_layer(layer_conv2, name="pool2")

# ReLU layer 2
layer_relu2 = new_relu_layer(layer_pool2, name="relu2")

# Flatten Layer
num_features = layer_relu2.get_shape()[1:4].num_elements()
```

Climate Scientist

$$\vec{u}(\vec{x}) = \sum_{i=1}^k \lambda_i \phi_i(\vec{x}) \vec{n}_i$$

Implementation (Fortran + OpenACC):

```
!$ACC PARALLEL &
!$ACC PRESENT( iqid_x_d, ..., ptr_vn_d, e_flx_avg_d, vn_d, vt_d, rbf_vec_coeff_e_d )
!$ACC LOOP GANG PRIVATE( i_startidx, i_endidx, jb )
  DO jb = i_startblk, i_endblk
    IF ( i_startblk == jb ) THEN; i_startidx = e_startidx; ELSE; i_startidx = 1; ENDIF
    IF ( i_endblk == jb ) THEN; i_endidx = e_endidx; ELSE; i_endidx = nproma; ENDIF
!$ACC LOOP VECTOR COLLAPSE(2)
    DO je = i_startidx, i_endidx
      DO jk = 1, nlev
        iqid_x_1 = iqid_x_d(je,jb,1)
        ! Average normal wind components
        ptr_vn_d(je,jk,jb) = e_flx_avg_d(je,1,jb)*vn_now_d(je,jk,jb) &
          + e_flx_avg_d(je,2,jb)*vn_now_d(iqid_x_1,jk,iqblk_1) &
          :
        ! RBF reconstruction of tangential wind component
        vt_now_d(je,jk,jb) = rbf_vec_coeff_e_d(1,je,jb) &
          * vn_now_d(iqid_x_1,jk,iqblk_1) &
          :
      ENDDO
    ENDDO
  ENDDO
!$ACC END PARALLEL
```

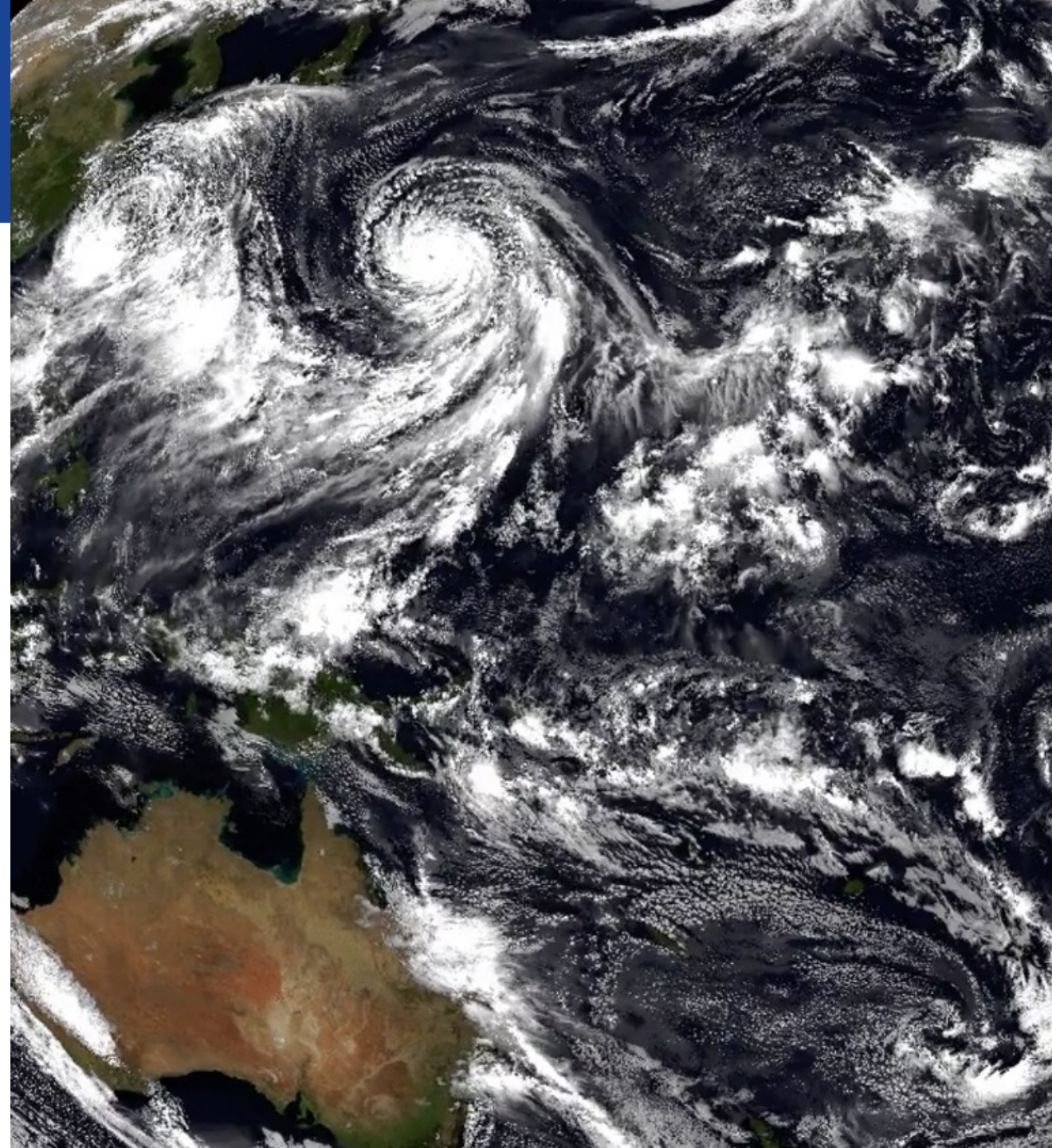
Goal

Build a global-storm resolving model in Python using an embedded domain-specific language that can run at scale on modern supercomputers

Started October 2019

Validation first, performance second

Focus on NVIDIA GPUs and x86 CPUs



But... Python is slow!

That depends on...

- Granularity
- Copy-free data-views

Example: Fully Python-wrapped version of NOAA's fv3gfs model is overhead-free.

```
import fv3core
import fv3gfs.wrapper as wrapper
from fv3gfs.util import io
from mpi4py import MPI

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

wrapper.initialize()

for step in range(wrapper.get_step_count()):
    wrapper.step_dynamics()
    wrapper.step_physics()
    if (io):
        state = wrapper.get_state()
        io.write_state(state, f"out_state_{rank}.nc")

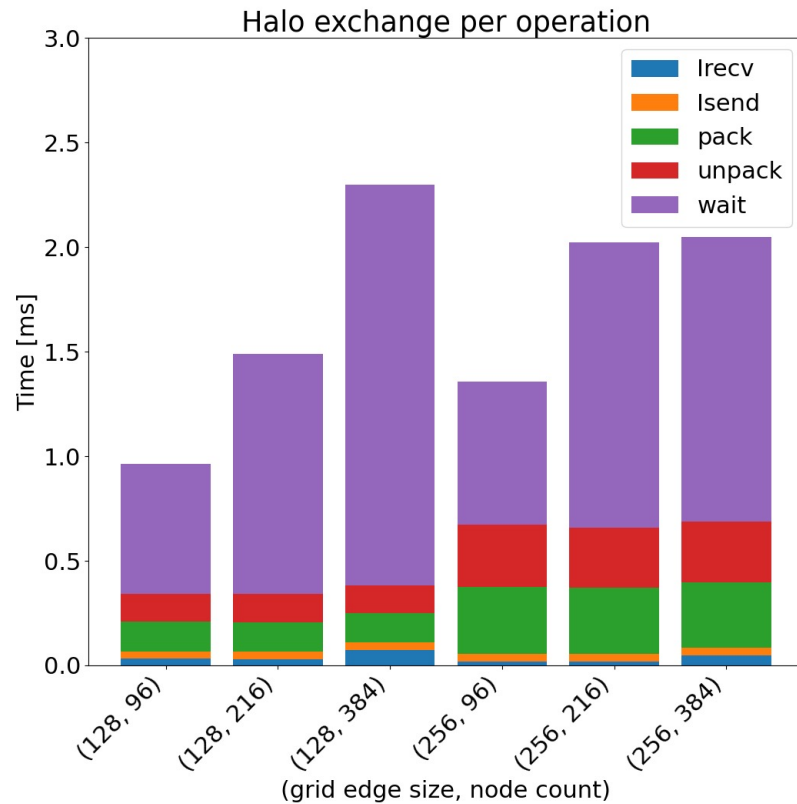
wrapper.cleanup()
```

McGibbon et al. 2020, GMD

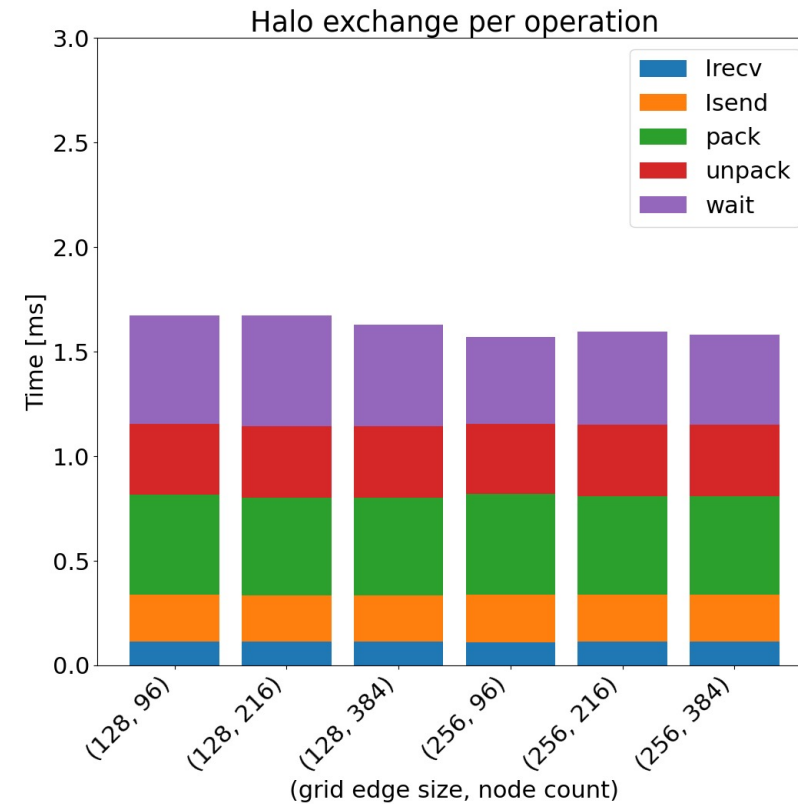
<https://doi.org/10.5194/gmd-14-4401-2021>

But... Python is not parallel!

Example: Halo-exchange of FV₃ dynamical core on Piz Daint



Fortran (FMS/MPI)



Python (CuPy/mmpi4pi)

But... Python is not HPC!

Driven by AI/ML, the fraction of HPC workloads written and tooling available in Python is rapidly increasing.

Train GPT-3

175 billion parameters

PyTorch

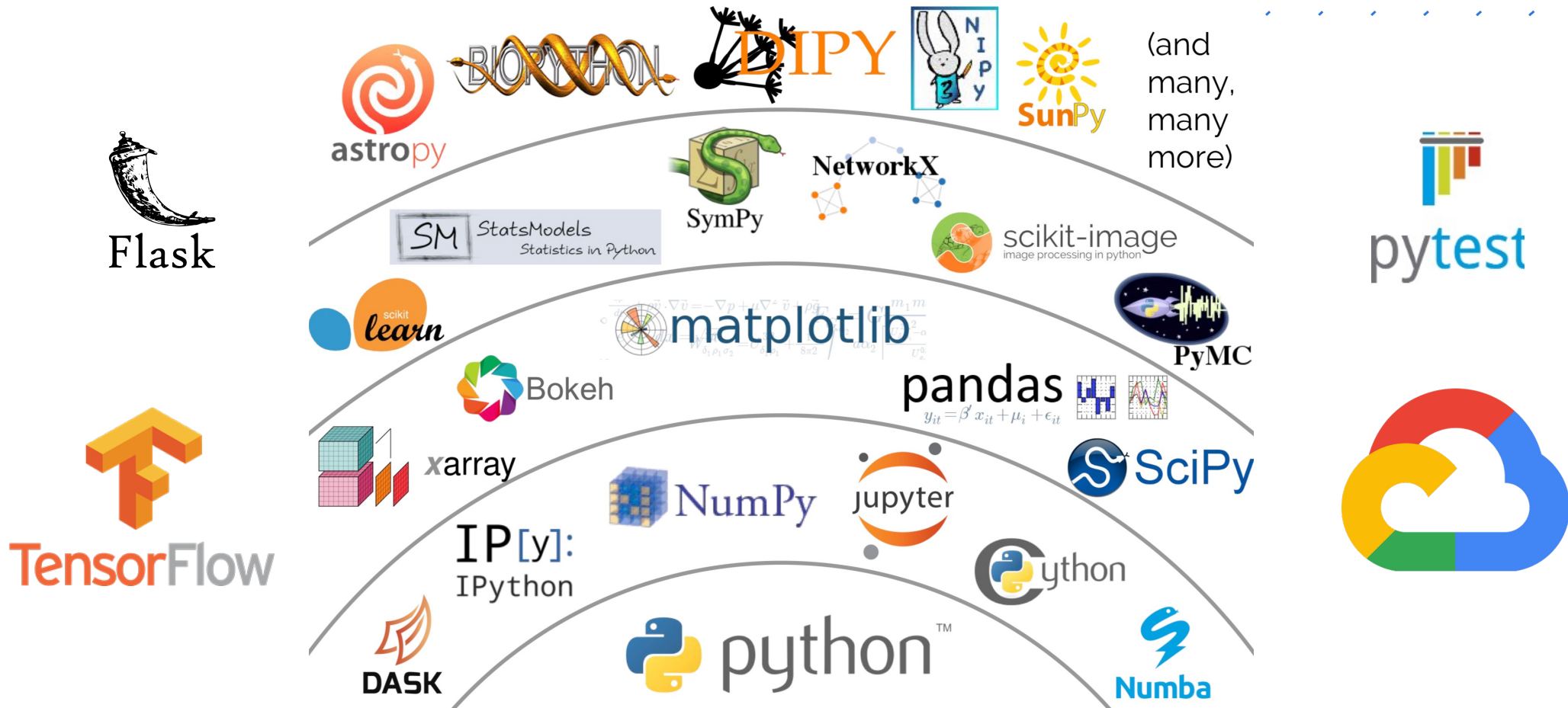


1km Global Storm Resolving Model

5 year integration



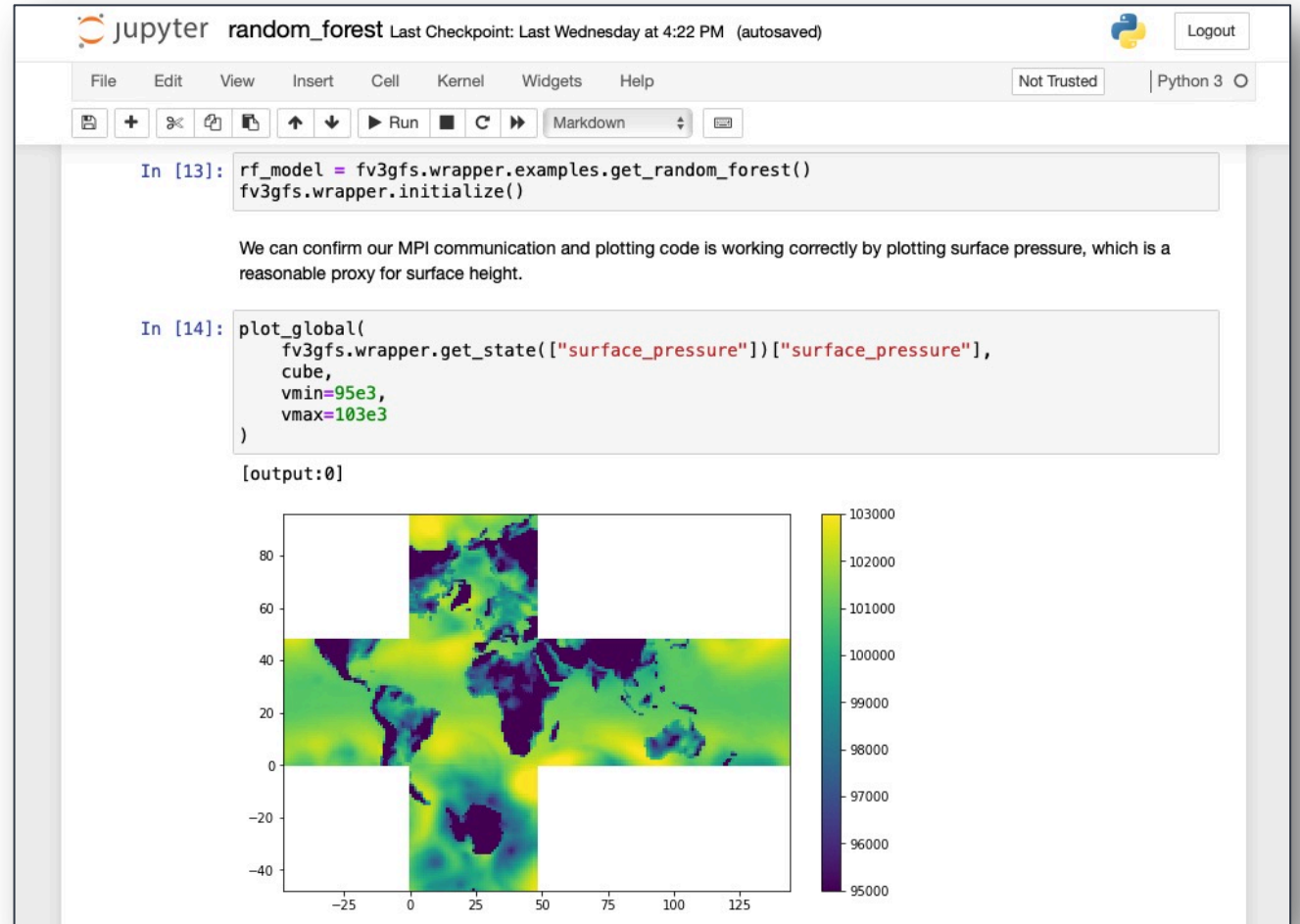
And... Python Ecosystem



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

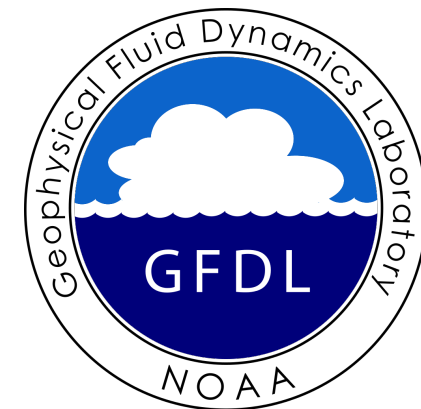
And... Developing in Python

- Python enables interactive development
- Easy to integrate with machine learning, analysis code, online diagnostics
- Same code runs at scale or on a laptop
- Easy to maintain and use extensive unit tests

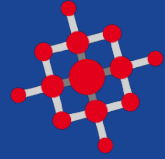


The FV3GFS / xSHIELD Model

- FV3 dynamical core integrated into UFS, GEOS, CESM, GFDL models
- xSHIELD = FV3 + GFS physics for global storm-resolving simulations (DYAMOND)
- Cubed-sphere grid balances uniform resolution and simple code
- Horizontal finite volume dynamics
- Vertical Lagrangian dynamics with remapping
- Highly optimized for x86 CPU architectures



GT4Py



- Open-source, open-development project <https://github.com/GridTools/gt4py>
- Joint development with ETH Zurich/CSCS and MeteoSwiss
- Part of the GridTools ecosystem of tools and libraries for weather and climate
- GT4Py is embedded in Python
- Emphasis on tight integration with scientific Python stack
- Multiple backends: **Python** (NumPy), **CPU** (C++/OpenMP), **GPU** (CUDA), ...

The screenshot shows the GitHub repository page for GridTools/gt4py. The repository is in the 'master' branch and has 10 pull requests, 73 issues, and 15 forks. The file list includes:

File	Description	Last Commit
.github	Upped sphinx version requirement to fix br...	8 days ago
ci	CSCS-CI improvements (#191)	last month
docs	cli can generate stencil code from gtscrip...	2 months ago
examples	removed demo_ir example, added demo_ho...	9 months ago
src/gt4py	Correct documentation in infos.py (#223)	2 days ago
tests	Allow temporary variables (and function cal...	8 days ago
.gitignore	Prepare integration with Dawn toolchain (re...	4 months ago
.gitlab-ci.yml	CSCS-CI improvements (#191)	last month
.gitmodules	Initial commit. Basic functionality working.	12 months ago
.pre-commit-config....	Additional code checkers (#180)	last month
AUTHORS.rst	Fix bug that deletes arg annotations in gts...	2 months ago
CONTRIBUTING.rst	add documentation about new pre-commit ...	4 months ago
Dockerfile	Complete minimum working GitLab CI confi...	last month
LICENSE.txt	Initial commit. Basic functionality working.	12 months ago
LICENSE_HEADER.txt	Update information for externa	Initial commit. Basic functionality working.
MANIFEST.in	Minor fixes in docs and configuration settin...	9 months ago
README.rst	Fix formatting and add "Formatting and co...	2 months ago
bors.toml	[WIP] Initial GitLab-CI support (#183)	last month
pyproject.toml	rename stages, add more checkers, add py...	4 months ago
requirements-dev.txt	Upped sphinx version requirement to fix br...	8 days ago

The right sidebar shows the repository's 'About' section, which describes it as a 'Python API to develop performance portable applications for weather and climate.' It also shows the 'Releases' section (No releases published), 'Packages' section (No packages published), and 'Contributors' section (9 contributors). The 'Languages' section shows the following distribution: Python 98.5%, C++ 1.2%, and Other 0.3%.

GT4Py Pipeline

Application (DSL Code)

```
import numpy as np
from gt4py.gtscript import Field, PARALLEL,
computation, interval

def laplacian(field):
    return - 4. * field[ 0, 0, 0] + field[-1, 0, 0]
        + field[1, 0, 0] + field[0, -1, 0]
        + field[0, 1, 0]

def laplap_stencil(in_field: Field[np.float64],
out_field: Field[np.float64]):
    with computation(PARALLEL), interval(...):
        tmp = laplacian(in_field)
        out_field = laplacian(tmp)
```

Frontend

Toolchain

Backend

Python AST

Definition IR

Analysis, Parallelization

Optimizable IR

Optimizations

Code generation (e.g. C++)

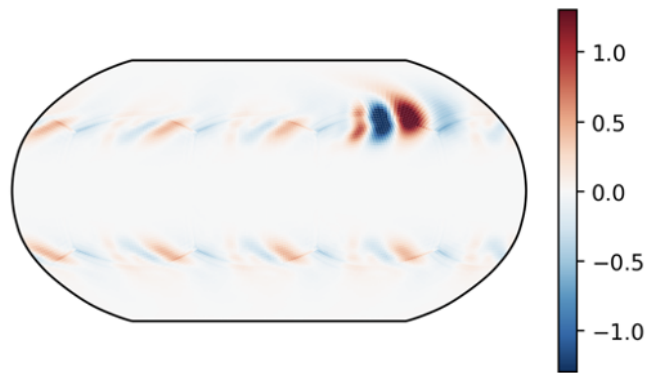
Python module

FV3core: DSL port of FV3

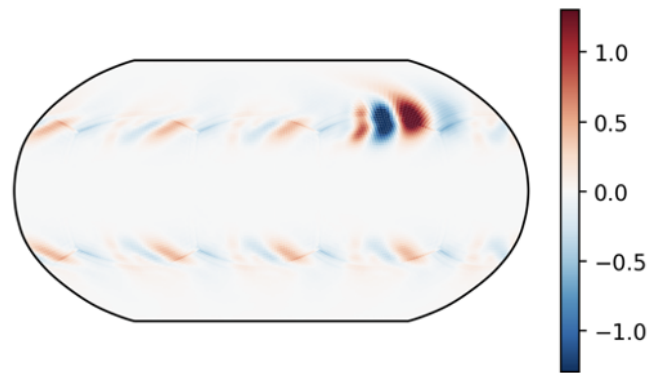
Check it out on GitHub!
<https://github.com/VulcanClimateModeling/fv3core>

Baroclinic instability testcase (Jablonowski and Williamson 2006)
6 day surface temperature anomaly [K]

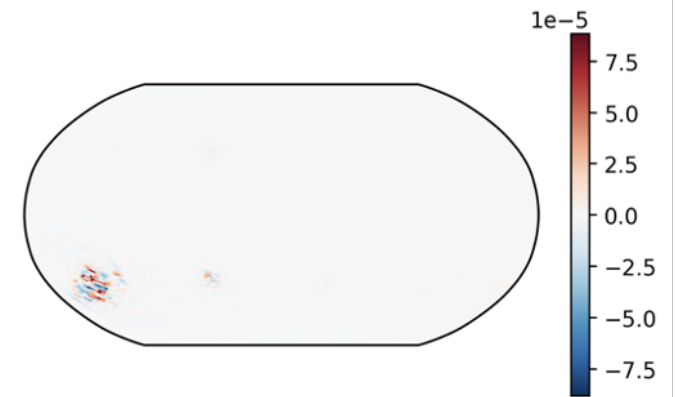
Reference
(Fortran, x86 CPU)



DSL Port
(Python, NVIDIA GPU)



Difference



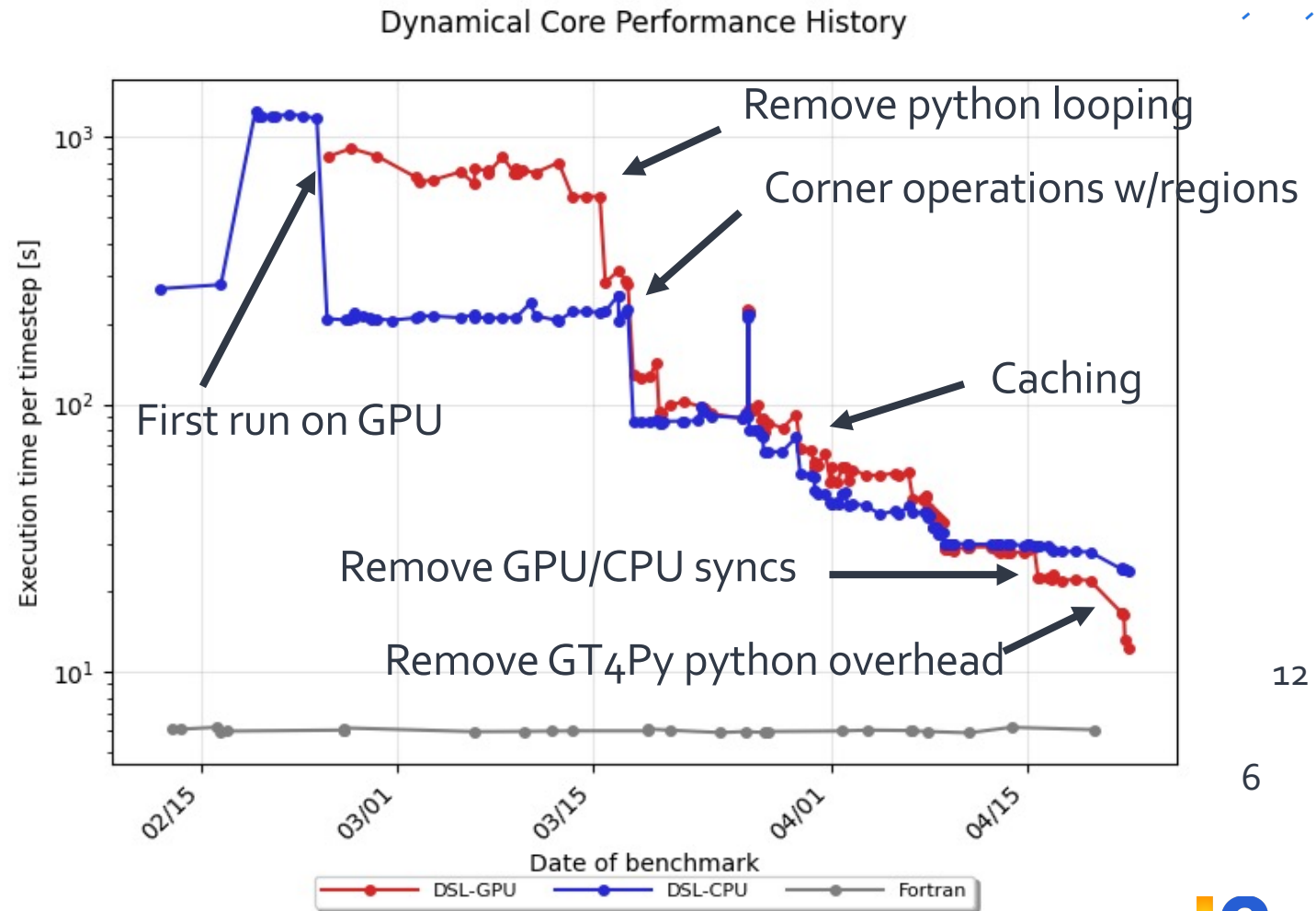
Performance: FV3core

Making it fast...

10 weeks of improvements

- Remove python code
- Use new GT4py features to merge stencils
- Cache temporary objects
- Asynchronous execution model











Currently 2.8x slower on CPU and 1.6x faster on GPU as compared to Fortran reference



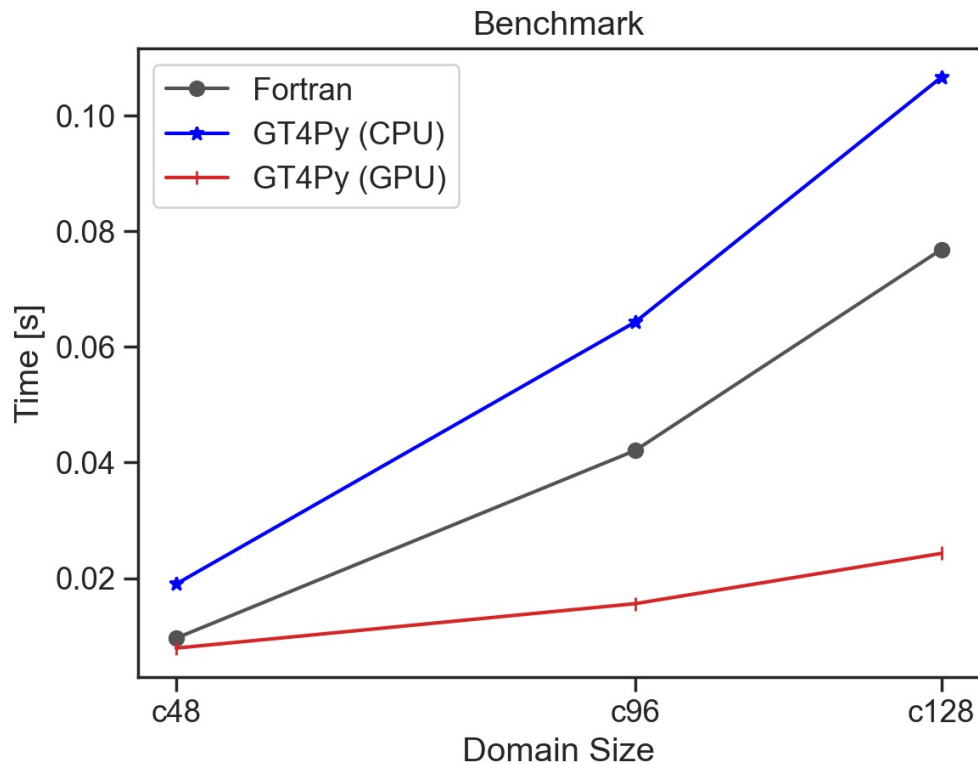
12

6

Physical Parameterizations

	Microphysics	PBL & Turbulence	Sea-Ice	Shallow Convection	LSM	Radiation
Authors	 Mikael	 Chris	   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 (sa-MF)	GFS Noah Land Surface Model	GFS RRTMG
Status	Ported	Ported	Ported	Ported	Ported	In progress

Performance: Microphysics



Domain size: 128 x 128 x 79

	Runtime [s]	Speedup
Fortran	0.077	REF
GT4Py (CPU)	0.107	0.72
GT4Py (GPU)	0.024	3.2

System: Piz Daint, CSCS
CPU: Intel Xeon E2690 v3 @ 2.6 GHz, 12 core
GPU: NVIDIA Tesla P100

Further optimization of DSL toolchain needed to speed up runtime

Next Steps

Model

- Finish port of radiation scheme
- Integrate physical parameterizations with dynamical core
- Refactor code to make use of new DSL features

DSL Toolchain

- Finish migration to new backends
- Extend with new features to eliminate more Python code
- Improve performance of x86 CPU and NVIDIA GPU backends

Summary & Outlook

Python/DSL-based weather and climate models can be fast and portable.
Huge push from AI/ML continually improving ecosystem for HPC applications.

They enable exciting new possibilities

- Optimizations that would not be desirable / feasible without the high level of abstraction (collaboration with CSCS/ETH)
- Integration with ML learning and emulation workflows (in house)
- Automatic differentiation (on-going collaboration with Google Research)
- Integration with other toolchains (collaboration with SPCL/ETH and DaCe)

Thank you!