National Aeronautics and Space Administration

Domain-Specific Language (DSL) Adoption into NASA's Goddard Earth Observing System (GEOS) Model

October 13, 2023

Purnendu Chakraborty, Florian Deconinck, Christopher Kung

Outline

- Motivation
- DSL Integration Strategy
- Validation : Held-Suarez & Aquaplanet
- Optimization
- Ported physics packages to GPU via OpenACC
- Next Steps

Outline

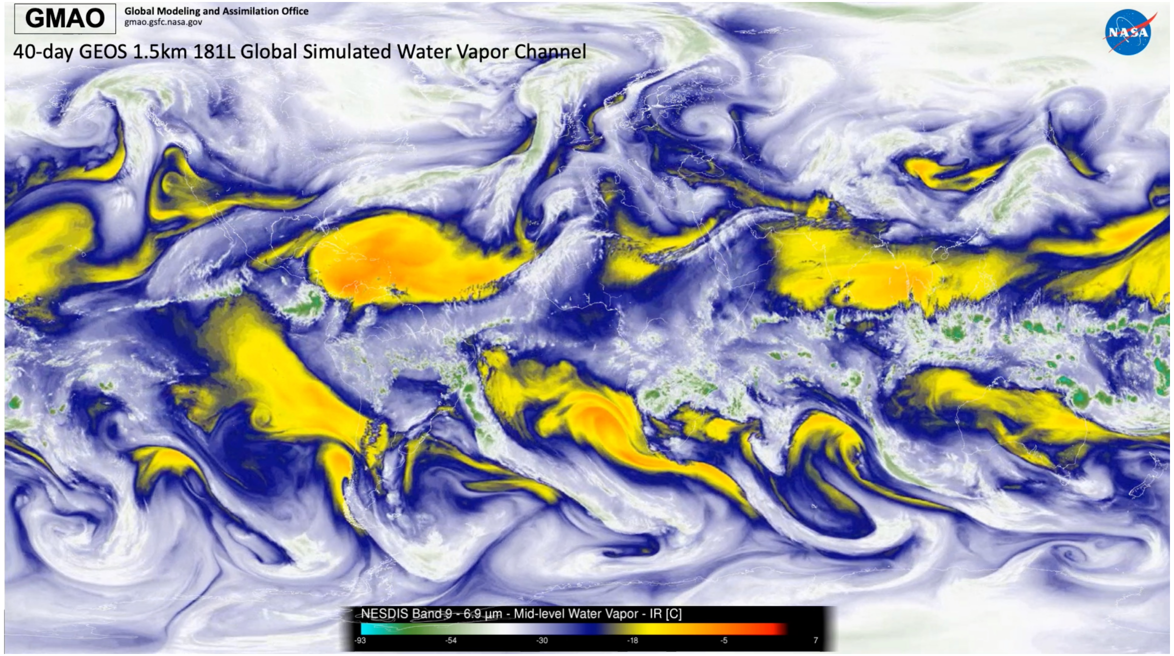
- **Motivation**
- DSL Integration Strategy
- Validation : Held-Suarez & Aquaplanet
- Optimization
- Ported physics packages to GPU via OpenACC
- Next Steps

Next-Generation Goddard Earth Observing System (GEOS) code

Goal : Develop the next generation GEOS model that enables an increase in resolution and scalability to meet future NASA Global Modeling and Assimilation Office (GMAO) requirements.

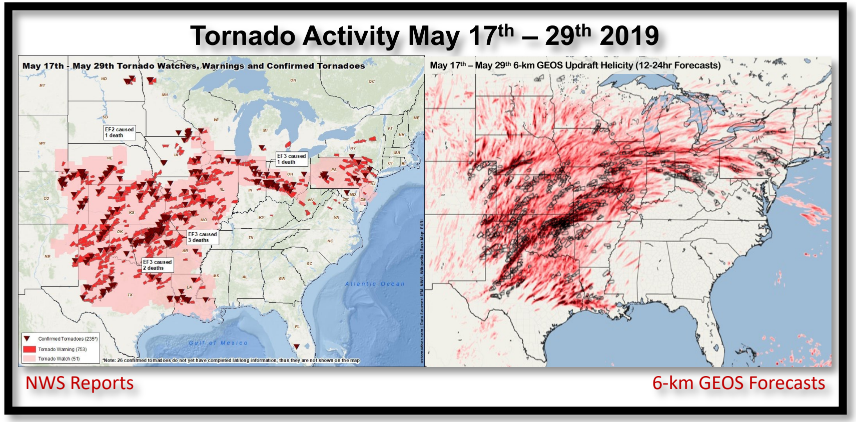
Team

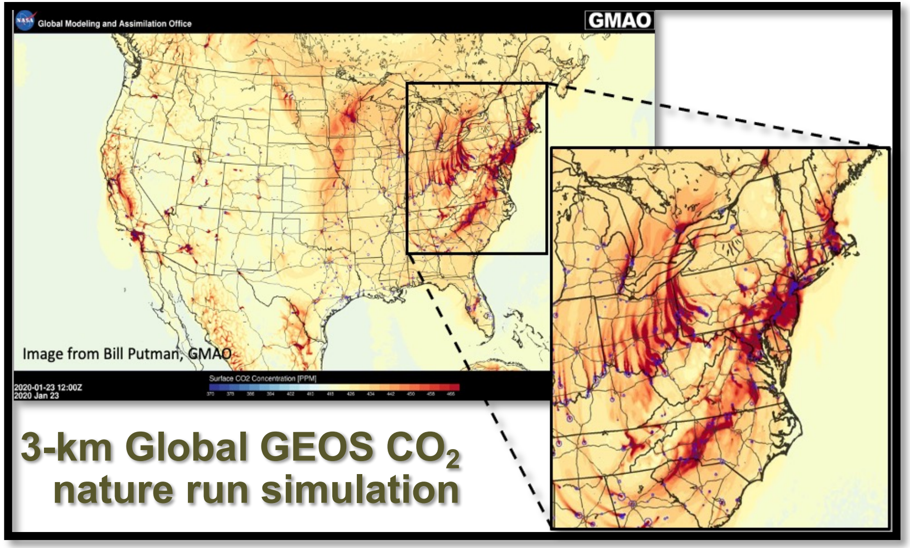
- Developers : Purnendu Chakraborty, Florian Deconinck, Chris Kung
- Support : GMAO Software Integration Team, NASA Center for Climate Simulation (NCCS)
- Management : Craig Pelissier, Bill Putnam, Dan Duffy, Tsengdar Lee
- External Collaborators : NOAA, NVIDIA, ETH Zurich’s SPCL and CSCS, AI2



GEOS Future Requirements

- **Production support for Coupled Data Assimilation**
  - Numerical Weather Prediction: 6-km Atmosphere Model (ATM) coupled to 6-km Ocean Model (OCN)
  - Sub-Seasonal to Seasonal Predictions: 25-km ATM coupled to 25-km OCN (Ensemble members creating 30- to 90-day predictions)
  - Reanalysis: 12-km ATM coupled to 12-km OCN
- **Ultra-high resolution global Nature Runs for Climate Observing System Simulation Experiments**
  - Coupled 3-km ATM and 3-km OCN
  - Carbon and chemistry: 3- to 1-km ATM
  - Global convection resolving weather: < 1 km ATM





Outline

- Motivation
- **DSL Integration Strategy**
- Validation : Held-Suarez & Aquaplanet
- Optimization
- Ported physics packages to GPU via OpenACC
- Next Steps

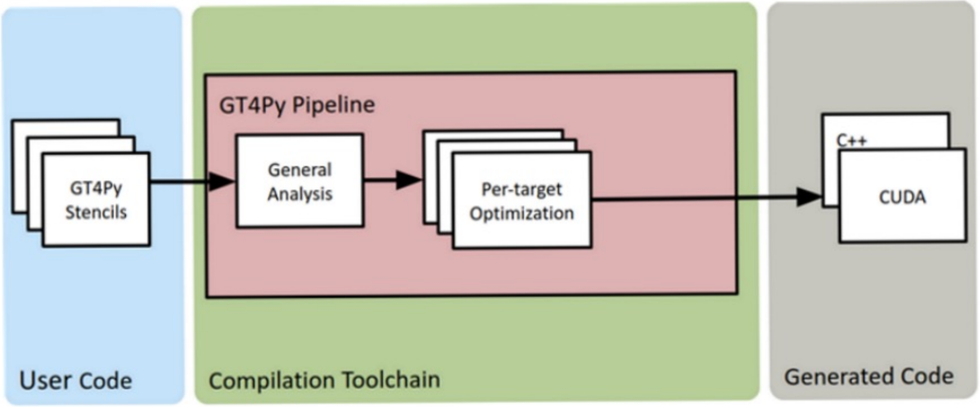
Approach / Strategy

- **Leverage heterogeneous architectures via Domain Specific Language (DSL) and OpenACC**
  - Recognize that accelerator-based heterogeneous systems provide a promising platform to meet future GEOS requirements.
- **DSL : Programming language specialized to a particular domain**
  - DSL adoption creates an opportunity to create portable and scalable code across multiple platforms, including traditional CPU systems and accelerator-based systems, by abstracting away details of the computing architecture
  - For GEOS, we utilize the GridTools for Python (GT4Py) DSL
- **Incorporate a GT4Py port of Geophysical Fluid Dynamics Laboratory's (GFDL) dynamical core (FV3) created by Vulcan / Allen Institute for AI (AI2) into GEOS**
  - GT4Py-ported FV3 (gtFV3) is a non-hydrostatic code used for global storm-resolving modeling
  - Leverage DSL without extensive investment in code rewriting / refactoring

Some definitions

GT4Py (GridTools for Python)

- GridTools
  - C++ DSL
  - Express "stencil"-like patterns concisely
  - Multiple backends - OpenMP(CPUs)/CUDA/HIP
  - Easy switching between backends
- Frontend code in Python

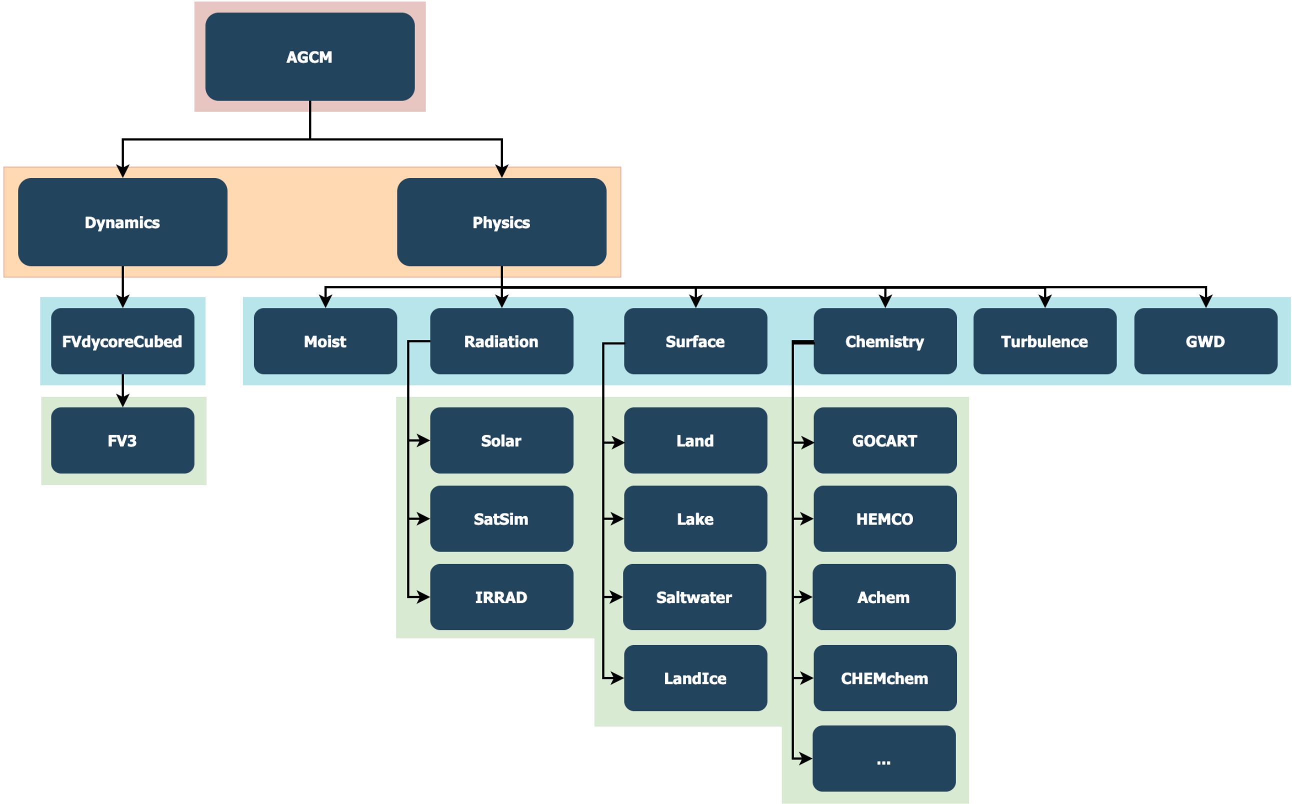


gtFV3[1]

- GT4Py port of (Fortran) FV3

[1] A Python-based Performance-Portable Implementation of the FV3 Dynamical Core. Dahm et al., EGUsphere [preprint: *https://doi.org/10.5194/egusphere-2022-943, 2022.*]

GEOS



AGCM
Dynamics
Physics
FVdycoreCubed
FV3
Moist
Radiation
Surface
Chemistry
Turbulence
GWD
Solar
SatSim
IRRAD
Land
Lake
Saltwater
LandIce
GOCART
HEMCO
Achem
CHEMchem
...

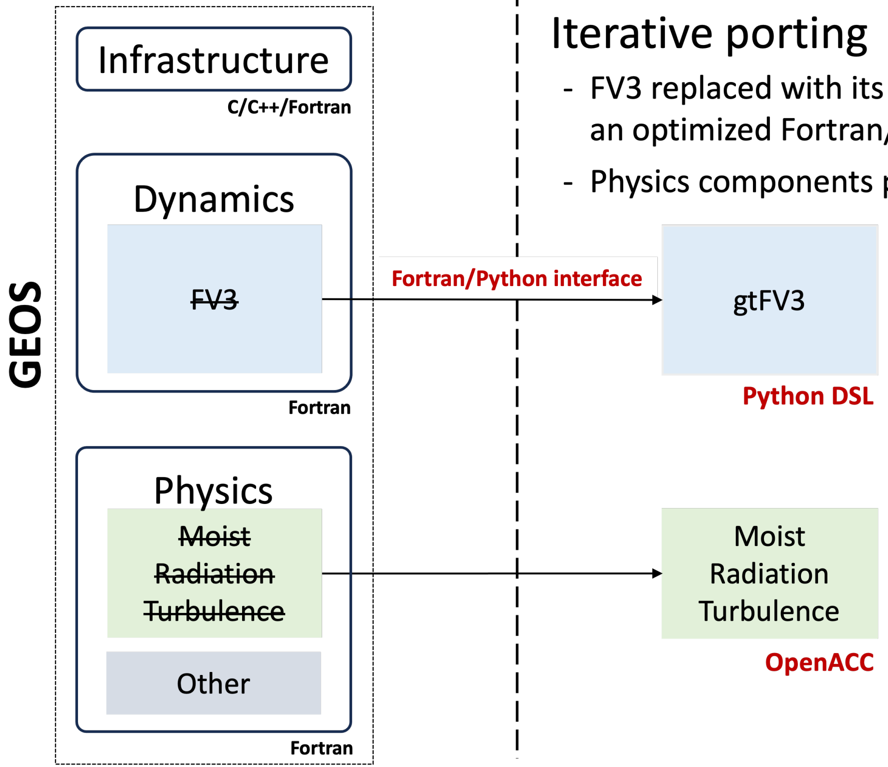
GPU Porting Approach

CPU

GPU

Iterative porting

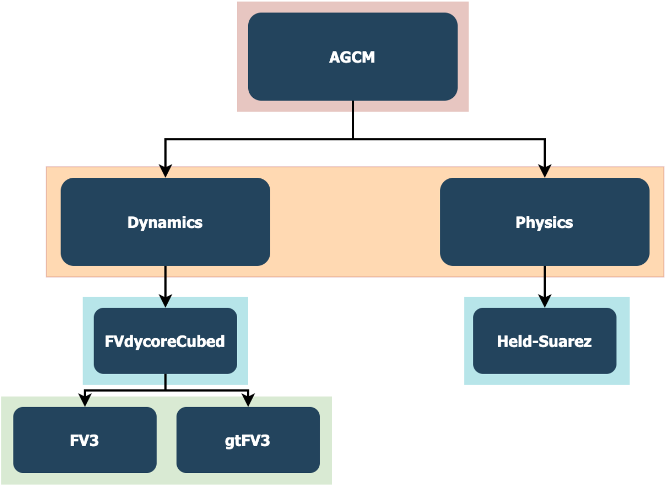
- FV3 replaced with its Python DSL port via an optimized Fortran/Python interface
- Physics components ported via OpenACC



Outline

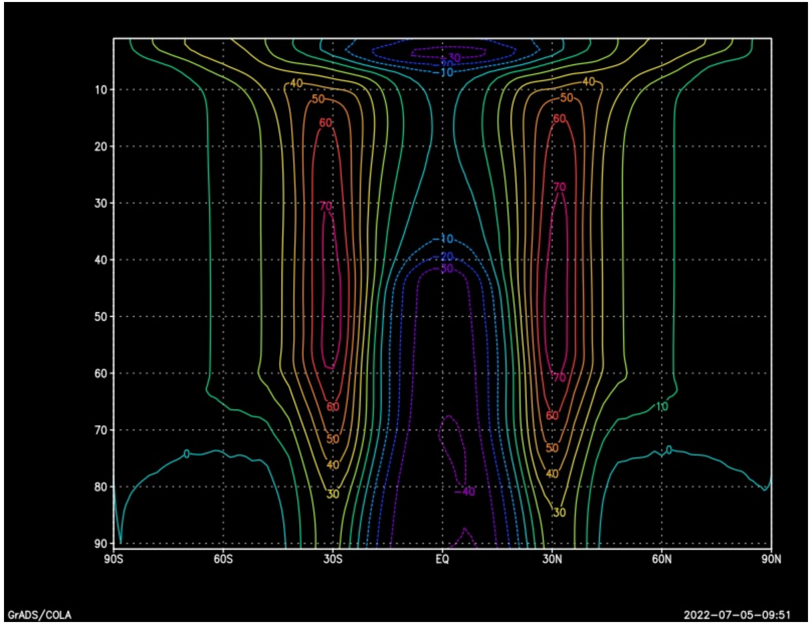
- Motivation
- DSL Integration Strategy
- **Validation : Held-Suarez & Aquaplanet**
- Optimization
- Ported physics packages to GPU via OpenACC
- Next Steps

Validation Case – Held-Suarez

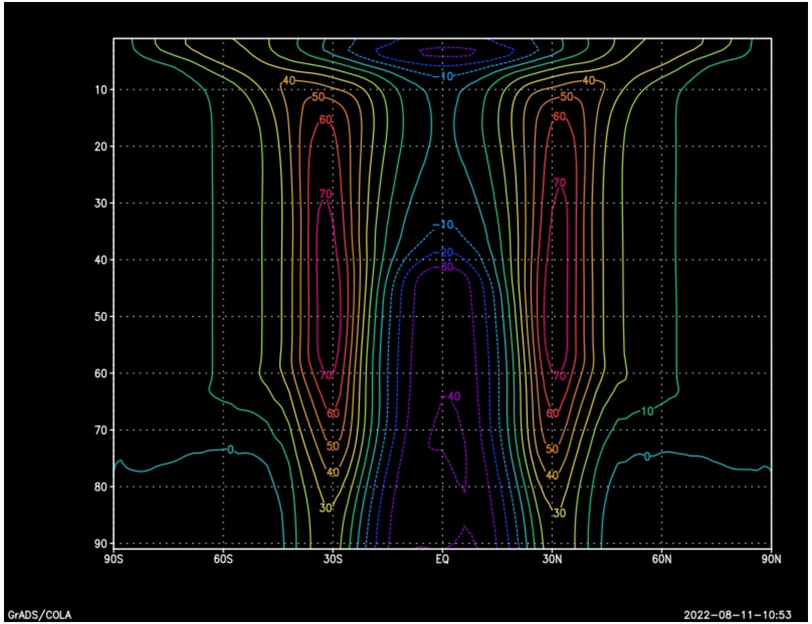


FV3 – HS vs gtFV3 – HS (OpenACC)

FV3 (CPU), HS (CPU)

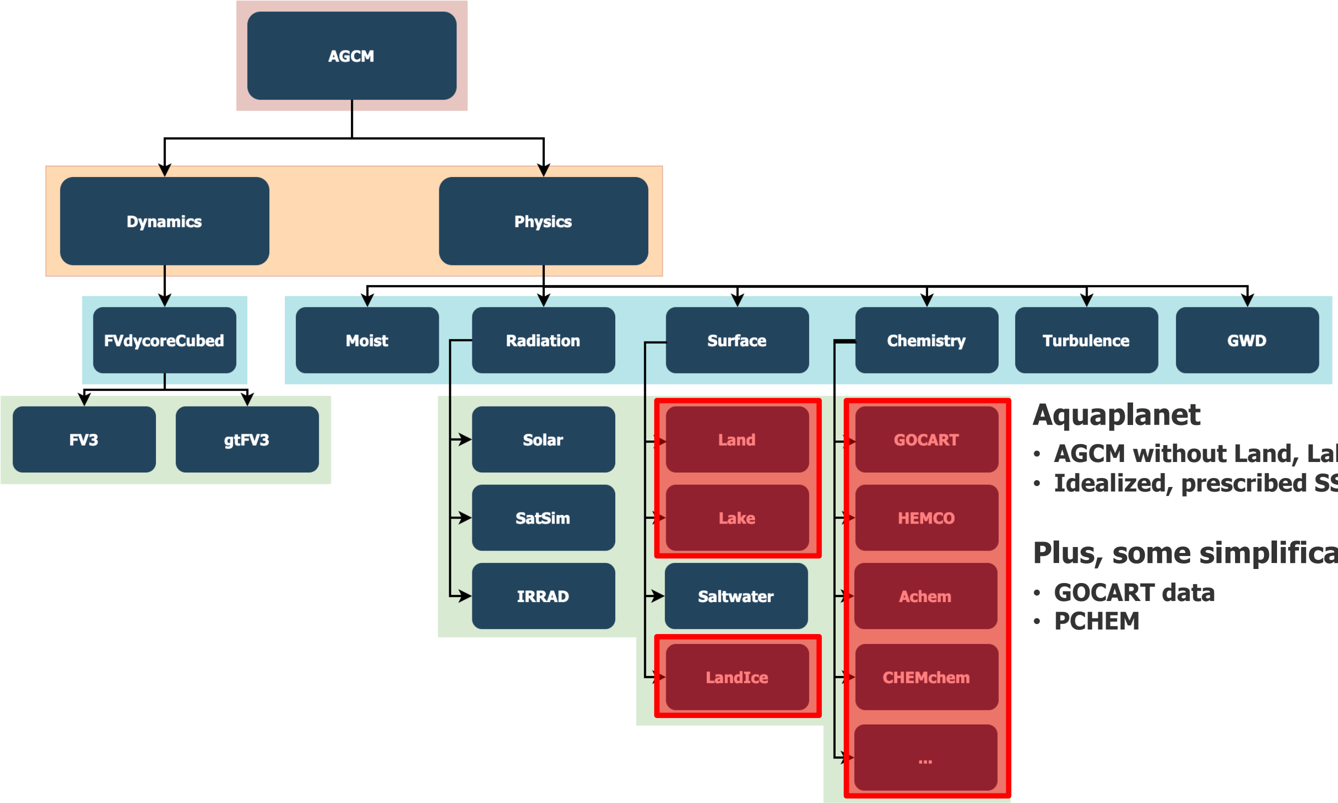


gtFV3 (GPU), HS (GPU)



Zonal mean of U-winds averaged over 30 days

Validation Case - Aquaplanet



Aquaplanet

- AGCM without Land, Lake and LandIce
- Idealized, prescribed SST

Plus, some simplification

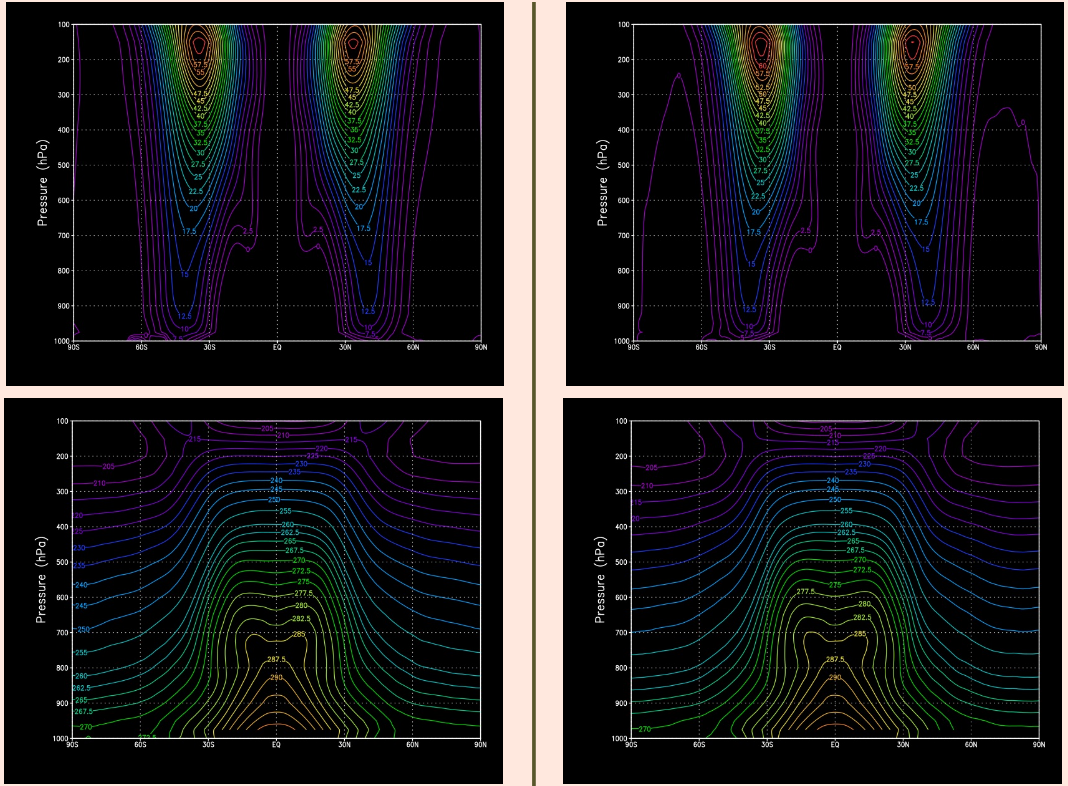
- GOCART data
- PCHEM

Aquaplanet-FV3 Vs Aquaplanet-gtFV3 – comparing time averages

- Assembled application (one codebase, one build, choose FV3/gtFV3 at runtime), resolved significant differences between old/new FV3 and gtFV3
- From data dump (C180 - 50km), ran 18 months simulation (6 months spin-up + 12 months **analysis**) for both cases
- Compared time-averaged zonal means over the **analysis** period

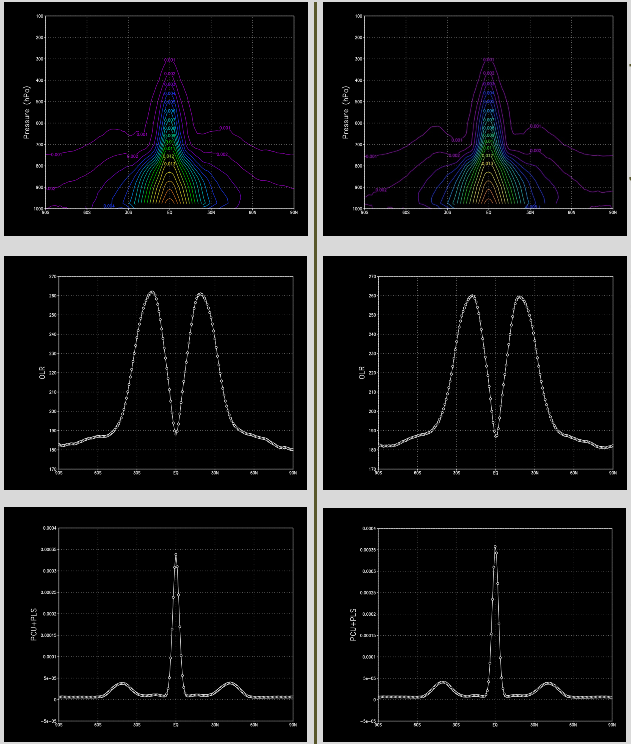
Zonal wind $(m\,s^{-1})$

Temperature (K)



Aquaplanet (FV3)

Aquaplanet (gtFV3)



Specific humidity (kg/kg)

Outgoing Longwave Radiation (W/m^2)

Precipitation $(kg/m^2/s)$

Aquaplanet (FV3)

Aquaplanet (gtFV3)

Outline

- Motivation
- DSL Integration Strategy
- Validation : Held-Suarez & Aquaplanet
- **Optimization**
- Ported physics packages to GPU via OpenACC
- Next Steps

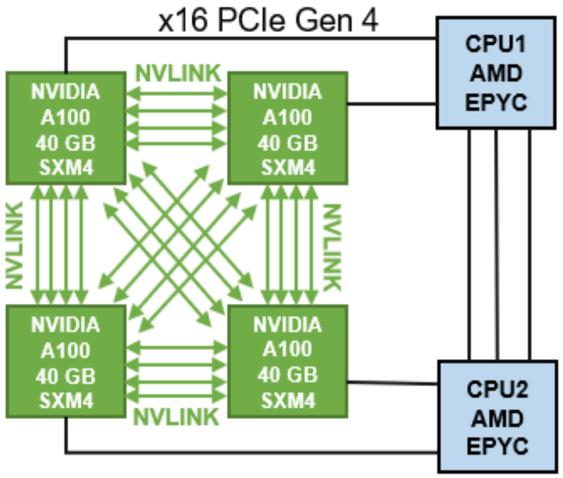
GEOS-gtFV3 Performance Strategy

- **Many-processes vs Many-threads**
  - GEOS currently uses a many-processes paradigm: one CPU core per task, with problems divided in smaller problems.
  - GPUs are built for many-threads paradigm: one GPU per problem computed across many-threads, with the bigger problem size that fits memory.

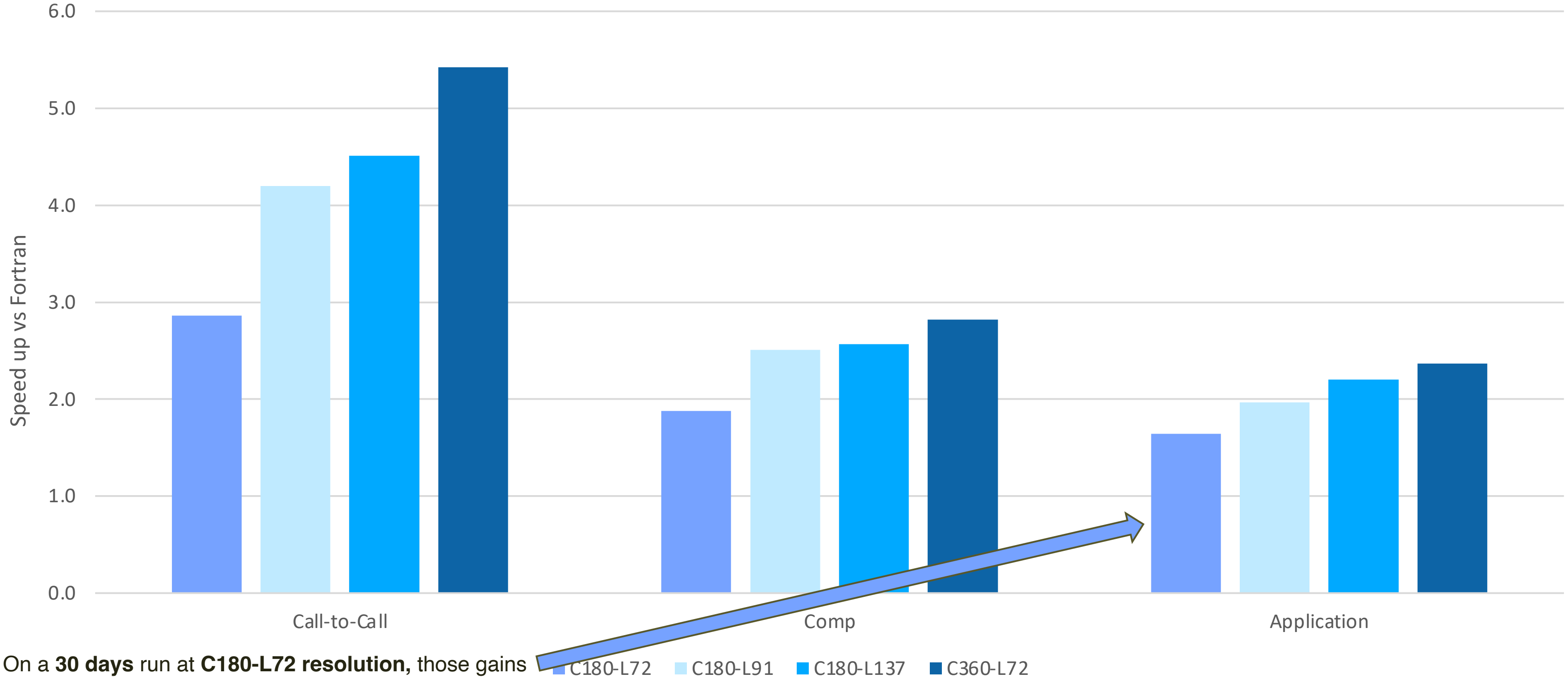
- **Due to the hybrid nature of our work, we need solutions that can do *both***
  - **Solution 1** : Map many-processes onto GPUs via NVIDIA's Multi-Process Service (MPS).
  - **Solution 2** : Multi-level domain decomposition for physics and dynamics
    - GMAO SI Team is adding GEOS framework capabilities that will create multi-level domains. Development is ongoing.
    - Enables optimal hardware usage: GPU is given the entire problem at once, CPU is given a smaller chunk of the problem.

GEOS-gtFV3 Benchmarks: GPU vs CPU node

- The benchmark setup compares a CPU-node with a GPU-node as available on Discover.
- **Discover Hardware Setup**
  - CPU: AMD EPYC 7402, 24 cores
  - GPU: NVIDIA A100, 40 GB VRAM
  - Node: 2x CPU, 4x GPU, 2x InfiniBand
  - Using **MPS**: 12 CPU cores per GPU (node to node)
- **Experiment**
  - Held-Suarez configuration
  - Mainly focus on the dynamical core performance
  - Performance comparison based on 24-hour simulation
  - 30-day simulation to check for stability



GEOS-gtFV3 Held-Suarez Performance Comparison



- On a **30 days** run at **C180-L72 resolution,** those gains translate as follows:
  - Fortran: 3h 49m
  - Hybrid GPU/CPU: 2h 21m (**1.62x** speedup consistent with 24-hour benchmark)

Outline

- Motivation
- DSL Integration Strategy
- Validation : Held-Suarez & Aquaplanet
- Optimization
- **Ported physics packages to GPU via OpenACC**
- Next Steps

OpenACC for Physics

- **Why OpenACC?**
  - Development remains in Fortran
  - Flexibility : GT4Py structures code based on stencil computations, making it less flexible if an algorithm doesn't easily map into a stencil.
- **What about "proprietary-ness"?**
  - OpenACC may not be the final solution since it's (mostly) NVIDIA-centric and OpenACC compilers (generally) are not available or less optimal for non-NVIDIA GPUs.
  - The OpenACC implementation can guide development of an OpenMP-offloading implementation
- **General OpenACC Porting Methodology**
  - Create a CPU-based standalone code containing physics schemes that verifies with input and output data generated from GEOS
  - Apply OpenACC to parallelizable code sections within the CPU-based standalone code
  - Verify that OpenACC standalone code computes data that closely matches GEOS generated data

Status of OpenACC Physics Standalones

Moist	OpenACC
fillq2zero	✓
buoyancy	✓
buoyancy2	✓
aer_activation	✓
cup_gf (v GF2020)	In progress
gfdl_cloud_microphys_driver	✓
evap_subl_pdf_loop	✓
radcoup_loop	✓
uw_shallow_convection	✓

Turbulence	OpenACC
run_edmf	✓
vtrisolvesurf	✓
vtrilu	✓
update_moments	✓

Gravity Wave	OpenACC
ncar_gwd	✓

Radiation	OpenACC
RRTMGP	✓

OpenACC Physics Integration Approach

--Run	1	604.914	71.76	0.583	0.07
----EXTDATA	192	0.047	0.01	0.047	0.01
----GCM	384	585.390	69.45	23.405	2.78
------AGCM	384	561.199	66.58	32.470	3.85
--------SUPERDYNAMICS	576	227.025	26.93	0.512	0.06
----------DYN	576	226.513	26.87	226.513	26.87
--------PHYSICS	384	301.680	35.79	4.721	0.56
----------GWD	384	8.924	1.06	8.924	1.06
----------MOIST	384	156.577	18.58	156.577	18.58
----------TURBULENCE	576	20.380	2.42	20.380	2.42
----------CHEMISTRY	576	20.937	2.48	0.121	0.01
------------CHEMENV	576	10.521	1.25	10.521	1.25
------------PCHEM	384	8.298	0.98	8.298	0.98
------------GOCART.data	576	1.998	0.24	1.998	0.24
----------SURFACE	576	5.523	0.66	1.648	0.20
------------SALTWATER	576	3.875	0.46	0.268	0.03
--------------SEAICETHERMO	576	1.130	0.13	1.130	0.13
--------------OPENWATER	576	2.477	0.29	2.477	0.29
----------RADIATION	384	84.619	10.04	0.314	0.04
------------SOLAR	384	53.611	6.36	53.611	6.36
------------IRRAD	384	21.464	2.55	21.464	2.55
------------SATSIM	384	9.229	1.09	9.229	1.09
--------ORBIT	384	0.024	0.00	0.024	0.00
------AIAU	192	0.010	0.00	0.010	0.00
------ADFI	384	0.044	0.01	0.044	0.01
------OGCM	384	0.732	0.09	0.410	0.05
--------ORAD	384	0.052	0.01	0.052	0.01
--------SEAICE	384	0.146	0.02	0.058	0.01
----------DATASEAICE	384	0.088	0.01	0.088	0.01
--------OCEAN	384	0.125	0.01	0.069	0.01
----------DATASEA	384	0.055	0.01	0.055	0.01
----HIST	384	18.893	2.24	18.893	2.24

Measured wallclock time

- Statistics from "RUN" portion of GEOS Aquaplanet setup
  - 96 ranks / 8 GPUS
- Overall Physics : ~50% of "RUN" execution time
  - Moist : ~25% of "RUN" execution time
  - Radiation : ~14% of "RUN" execution time
- Plan : Integrate OpenACC-ported Moist physics for greatest initial potential to reduce execution time

OpenACC Physics Integration: Moist

Times for component <MOIST>

Name	Min %	Min inclusive	Min exclusive	Mean %	Mean inclusive	Mean exclusive	Max %	Max inclusive	Max exclusive	PE max	PE min	# cycles
MOIST	0.00	135.47	0.00	0.00	160.18	0.00	0.00	184.86	0.00	00003	00022	387
--SetService	0.01	0.02	0.02	0.03	0.05	0.05	0.04	0.08	0.08	00024	00090	1
----generic	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	00034	00053	1
--Initialize	0.04	1.72	0.05	0.09	1.76	0.14	0.10	1.85	0.20	00003	00092	1
----generic	1.27	1.58	1.58	1.01	1.63	1.63	0.86	1.73	1.73	00000	00029	1
--Record	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	00088	00029	192
----generic	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	00079	00055	192
--Run	0.00	126.44	0.00	0.00	151.20	0.00	0.00	175.83	0.00	00032	00068	192
----GenRunMine	13.11	126.44	16.28	13.30	151.20	21.30	12.81	175.83	25.68	00009	00084	192
------CONV_TRACERS	0.07	0.09	0.09	0.06	0.09	0.09	0.05	0.10	0.10	00043	00085	192
------AERO_ACTIVATE	0.09	0.11	0.11	0.07	0.11	0.11	0.07	0.14	0.14	00040	00038	192
------GF	30.66	38.06	38.06	28.76	46.07	46.07	29.22	58.59	58.59	00089	00045	192
------UW	20.88	25.93	25.93	25.20	40.36	40.36	27.25	54.64	54.64	00041	00002	192
------GFDL_1M	2.88	35.28	3.58	2.26	43.27	3.62	1.84	51.22	3.68	00047	00024	192
--------CLDMACRO	7.16	8.89	8.89	6.15	9.86	9.86	5.63	11.29	11.29	00039	00005	192
--------CLDMICRO	18.05	22.41	22.41	18.60	29.80	29.80	18.56	37.21	37.21	00044	00028	192
--Finalize	0.00	7.17	0.00	0.00	7.17	0.00	0.00	7.17	0.00	00042	00094	1
----generic	5.77	7.17	7.17	4.47	7.17	7.17	3.57	7.17	7.17	00005	00000	1

- GF: Contains cup_gf (v GF2020)
- UW : Contains uw_shallow_convection
- GFDL_1M : Contains gfdl_cloud_microphys, evap_subl_pdf_loop, radcoup_loop

Moist Standalone Runtimes

Standalone	CPU Runtime (sec)	GPU Runtime (sec)	GPU Speedup
aer_activation	6.24×10^{-1}	8.70×10^{-3}	71.8
buoyancy	5.38×10^{-3}	1.99×10^{-4}	27.1
evap_subl_pdf_loop	7.50×10^{-1}	7.18×10^{-3}	104.4
fillq2zero	7.86×10^{-3}	1.10×10^{-4}	71.5
gfdl_cloud_microphys_driver	1.64×10^{0}	1.04×10^{0}	1.58
radcoup_loop	1.72×10^{-1}	1.20×10^{-3}	143.4
cup_gf (v GF2020)	1.43×10^{-1}	TBD	TBD
buoyancy2	1.20×10^{0}	2.64×10^{-2}	45.5

- Resolution : C180 (180 x 180 x 72)
- Compiler : nvfortran v22.3
- CPU : AMD EPYC 7402
- GPU : NVIDIA A100-SMX4 40 GB

- CPU runtimes are single core execution times
- Listed runtimes are the average of standalone runtimes from 6 different input sets
- Timings do not measure data transfer time between host and device

OpenACC Physics Integration Status

Timer	Routines	Porting status[1]	Integration status[2]
Total	buoyancy	Ported	Builds and runs
	buoyancy2	Ported	
--CONV_TRACERS			
--AERO_ACTIVATE	aer_activation	Ported	Builds
--GF	cup_gf	Porting	
--UW	uwshcu_inv	Porting	
--GFDL_1M			
----CLDMACRO	evap/subl/pdf (loop)	Ported	Builds
----CLDMICRO	gfdl_cloud_microphys	Ported	Builds
	radiation coupling (loop)	Ported	Builds
	fillq2zero	Ported	Builds and runs
In-between code		✗	✗

[1] Standalone, NVIDIA Fortran
[2] Integration into GEOS, GNU Fortran

Outline

- Motivation
- DSL Integration Strategy
- Validation : Held-Suarez & Aquaplanet
- Optimization
- Ported physics packages to GPU via OpenACC
- Next Steps

April 2021
Project start

August 2022
- Completed gtFV3-Held-Suarez (OpenACC) integration, validation
- Completed Turbulence porting to OpenACC

May 2023
- Completed Aquaplanet integration, started validation
- Porting Moist physics to OpenACC
- Started performance engineering of gtFV3

September 2023
- Complete Aquaplanet validation and benchmarking with gtFV3
- Porting Moist and Turbulence Physics to OpenACC
- Further optimization of gtFV3

Q1 2024
- Complete porting, integrating, validating and benchmarking of Aquaplanet-gtFV3 with OpenACC physics components

Thank you!

purnendu.chakraborty@nasa.gov
florian.g.deconinck@nasa.gov
christopher.w.kung@nasa.gov