Assessment of ESM Readiness Level for Exascale HPC

Interagency Council for Advancing Meteorological Services
Implementation Team - High Performance Computing

Jessie Carman, NOAA, jccarman0@gmail.com,
Oliver Elbert, Geophysical Fluid Dynamics Laboratory, oliver.elbert@noaa.gov,
Frank Giraldo, Naval Postgraduate School, fxgirald@nps.edu,
Mark Govett, NOAA, markgovett@gmail.com,
Lucas Harris, NOAA/Geophysical Fluid Dynamics Laboratory, lucas.harris@noaa.gov,
Thomas Hauser, National Center for Atmospheric Research, thauser@ucar.edu,
Dave McCarren, OPNAV/Navy, dmccarr@gmail.com,
Joseph Mouallem,
Cooperative Institute for Modeling the Earth System, Princeton University,
joseph.mouallem@noaa.gov,
Mark A. Olsen, NOAA Weather Program Office, mark.olsen@noaa.gov,
Todd Ringler, todd.ringler@mac.com,
Sarat Sreepathi, Oak Ridge National Laboratory, sarat@ornl.gov,
Mark Taylor, Sandia National Laboratories, mataylo@sandia.gov

April 15, 2025

Executive Summary:

- U.S. current and upcoming Exascale machines are GPU based.
- Exascale readiness requires porting ESMs to GPUs, involving large efforts: either new code or major refactoring of existing code.
- No dominant programming model to date. Approaches include rewriting in Domain Specific Languages, C++ with template based performance portability libraries and refactoring existing Fortran code to support GPU directives.
- Several atmosphere models are GPU-ready, with some progress on other components. No full ESM is GPU-ready as of 2024.
- Preliminary results from one of the first GCRMs to run on an exascale system show that modern GPU nodes can be ~6x faster than modern CPU nodes, or ~3.5x faster per Watt.
- We did not attempt to estimate GPU performance vs CPU performance on a per cost basis.

1. Introduction

Advancement of Earth System Models (ESMs) is becoming increasingly challenging due to a confluence of factors including increasing **model complexity** – to more fully represent the earth system, increasing **spatial resolution** - to achieve higher accuracy by resolving fine-scale dynamical to physical, biological, and chemical processes and their interaction, increasing **ensemble size** - to more accurately represent predictive uncertainty, and increased **computing requirements** – to enable more accurate and timely weather predictions and climate projections for societal benefit. The belief by many that computing will take care of itself is no longer valid given the disruptive changes in HPC that are driving up the cost of computing, increasing the difficulty of using emerging HPC effectively, and exposing limits in parallelism, portability and scalability of the ESM applications themselves.

This report is intended to assess the ability of current U.S. ESMs to use emerging exascale computing systems. ESMs broadly characterize the global models being developed for weather and climate prediction and projections. Current development efforts are pushing toward Global Cloud-Resolving Models (GCRM) for prediction out to decadal scales that more accurately resolve fine-scale processes needed to improve predictions at all time scales. Next-generation exascale computing is defined by an increasingly diverse, and rapidly changing landscape of processors (CPU, GPU, hybrid CPU-GPU, FPGA, AI), memory, file systems, interconnect, and other hardware that form the basis of very large, costly, ~hundreds of millions, systems with millions to hundreds of millions of processing cores. **Readiness** refers to the ability to run GCRMs (1 – 3km horizontal resolution) in production to support operational weather prediction (8 minutes per forecast day, or 180 times real time) and timely seasonal to decadal projections (1-5 simulated years per day, or 365 to 1,825 times real time) on future exascale class computers expected to be deployed worldwide. The goal of this report is to determine the current state of ESMs including performance, scalability, portability, and their ability to run at fine spatial scales being targeted by leading atmospheric and Earth system modeling centers.

During the writing of this report, it became clear that exascale readiness can be thought of as equivalent to GPU readiness. This is because current exascale HPC systems, as well as most of the near-exascale HPC systems obtain most of their floating point capability from Graphics Processing Units (GPUs). This is especially true in the U.S., where the Department of Energy has deployed three exascale systems, Frontier at the Oak Ridge Leadership Computing Facility (OLCF), Aurora, at the Argonne Leadership Computing Facility (ALCF), and El Capitan at the Lawrence Livermore National Laboratory. Here we define exascale via the matrix algebra benchmark used by the TOP500 list of the world's fastest computers (http://top500.org). Frontier and Aurora were the first exascale systems to make it into the TOP500 list, and as of November 2024, El Capitan, Frontier and Aurora are ranked #1, #2, and #3 on this list, respectively. The U.S. has a leadership role in HPC, as shown by the fact that 5 of the top 10 systems in the TOP500 are in the U.S. These 5 systems are all GPU based, and in fact 9 out of the 10 top systems are GPU based.

Due to the dominance of GPU based exascale systems, in this report we focus on the GPU readiness of the major U.S. ESMs. The primary audience is intended to be scientists, managers and leaders who may gain increased understanding of the approaches being taken by major

modeling centers to make use of GPU based systems in ESMs, and the potential capabilities of GPU based exascale systems for Earth system modeling. We hope the report will lead to both increased awareness and strengthened collaborations between agencies (under ICAMS), academia, and industry in order to overcome challenges in computing, model development and data handling that threaten continued improvements in U.S. Earth system prediction capabilities.

2. GPU Computing

The programming model is the defining characteristic of ESM readiness for GPU systems. Running today's ESMs on GPU systems requires substantial effort in software development and engineering, requiring new code or extensive refactoring and other code changes. This situation is similar to the transition from scalar vector machines to distributed memory parallel computers in the 1990s. This transition also required a fundamental change in the underlying programming model. The transition to GPUs has proven even more challenging since modern ESMs are far more sophisticated and involve much more code than climate and weather prediction models used in the 1990s. The transition is also difficult due to the different hardware instruction sets used by the major GPU vendors (NVIDIA, AMD and Intel), with each vendor providing a different low level programming model. A goal of many modeling centers is **performance portability**, where a single code base can run effectively on traditional CPU systems and support a range of GPU systems.

Nearly all major ESMs are written in Fortran with the MPI message passing interface. Transitioning these codes to GPUs largely preserves the MPI approach, but requires major changes to the Fortran code, representing the execution model on each compute node. Programming models being considered by U.S. modeling centers include Fortran+Directives (OpenACC and OpenMP, R. Usha et al., IEEE High Performance Extreme Computing Conference (HPEC), 2020), Domain Specific Languages (DSL), and C++/Kokkos (C.R. Trott et al., IEEE Transactions on Parallel and Distributed Systems, 2022). In the Fortran+Directives approach, existing Fortran code can be modified to make use of GPUs, making this an attractive approach that would appear to be the least disruptive. However, obtaining good GPU performance for this approach often requires substantial code refactoring. This approach also relies on vendor support with their Fortran compilers, with widely different levels of support across OpenACC, OpenMP, and GPUs from different vendors. For the DSL approach, the model components are rewritten in a higher level language (as compared to Fortran) with language constructs designed to support the type of operations found in ESMs, and the DSL has the capability to translate these constructs into the various low level GPU programming languages (J. Dahm et al., Geosci. Model Dev. 2023). In the C++/Kokkos approach, the model components have to be rewritten in C++, using the Kokkos programming model to abstract the on-node parallel execution model. Kokkos is not domain specific, but it is similar to a DSL in that it includes abstractions for frequently used parallel computing patterns and policies that provide details for how these patterns are to be applied. The C++/Kokkos approach is similar to the Fortran/Directives approach, but has the advantage that it avoids relying on vendor support and instead uses standard C++ template metaprogramming techniques to translate Kokkos operations into

vendor specific GPU code. As with OpenACC and OpenMP, there are several C++ programming models similar to Kokkos, including SYCL (Howes and Rovatsou, https://www.khronos.org/registry/sycl/specs/sycl-I.2.pdf) and YAKL (M. Norman et al., International Journal of Parallel Programming, 2022).

3. Earth System Prediction Models

Earth System Models couple together global models of the Earth's atmosphere, oceans, land surface and ice, in order to simulate the Earth's weather as well as past, present and future climate. ESMs apply physical laws to model the general circulation of the atmosphere, ocean and sea ice as well as land processes and include numerous subgrid parameterizations representing unresolved processes such as turbulence, cloud physics, radiation, aerosols and chemistry. Each ESM component model (atmosphere, land, ocean and ice) represents numerous multi-scale processes and is a parallel application in its own right, with its own development history. Components often run on different grids, coupled together at their shared boundaries.

State-of-the art ESMs have long been used for climate modeling, both for increasing our understanding of the Earth system and for future climate projections. They are a key tool for understanding how the availability of natural resources will change over time as well as respond to external factors like greenhouse gas emissions and other human activity. They are increasingly used for numerical weather prediction, where the feedbacks between atmosphere, ocean, land and ice processes can improve short range forecasts and are critical for sub-seasonal and seasonal forecasts. ESMs are also considered critical for the development of Earth system digital twins, where they may play the role of a digital twin, or serve as the backbone of a digital twin system, generating the massive data sets that can be made accessible via machine learning and artificial intelligence approaches (P. Bauer et al., Nature Comput. Sci., 2024, The Digital Twin Landscape, National Academies of Sciences, Engineering, and Medicine, 2024, doi: 10.17226/26894).

As mentioned in the introduction, this working group has been focused on the GPU readiness of Global Cloud Resolving Models (GCRMs). GCRMs are a key component of any km-scale ESM. They typically represent the most expensive model component within an ESM, in terms of both resources required and time-to-solution. They are the most complex component, due to the combination of dynamical motions requiring small time-steps to resolve, requirement to transport large numbers of atmospheric constituents, and the extensive suite of unresolved physical processes that must be parameterized. The multiphysics nature of GCRMs results in a large number of numerical motifs, with the computational cost spread relatively evenly over hundreds of different computational kernels, combined with the need for frequent internode communication. As such, GCRM performance on GPU systems is a good indicator of the full ESM performance, and the GPU acceleration obtained by a GCRM can be taken as a rough estimate of the minimum speedups that can be attained by other ESM components.

3.1 Survey of U.S. Developments

This subsection contains summaries contributed by several modeling centers, listed alphabetically.

CESM Model The Community Earth System Model (CESM) is a hydrostatic global model that contains a suite of horizontal discretization methods including spherical harmonics, finite volumes, and spectral elements with a 2nd order finite difference Lorenz grid in the vertical. The time-integration is based on the classical semi-implicit method which, for spherical harmonics hydrostatic models, can be performed algebraically; for the FV variant a semi-Lagrangian method is used whereas for the SE variant explicit time-integration is used. CESM uses MPI to scale on CPU hardware. The default setting for IPCC-type simulations is the spectral element horizontal discretization with a CSLAM semi-Lagrangian option for tracers. There is on-going work in using both MPAS and FV3 as the dynamical core; the MPAS option offers a nonhydrostatic model while the FV3 option would remain hydrostatic.

CLIMA Model CliMA is developing a new Earth system model that is performance portable across hardware platforms with either CPUs or GPUs. CliMA has developed a nonhydrostatic atmosphere model that uses spectral elements in the horizontal and finite differences in the vertical (similar to E3SM). It incorporates new parameterizations for turbulence, cloud, and convection that are based on physical laws but also contain machine-learned elements. CliMA has also developed a new ocean model that seamlessly spans scales from large-eddy simulations to global simulations and runs on GPUs; the results for the ocean component were submitted to SC 23 for consideration for the Gordon Bell prize. A new land model, likewise performance portable and sharing software infrastructure with the atmosphere, is also under development. The Clima model will leverage exascale capabilities in multiple ways, such as rapid calibration and uncertainty quantification using ensemble Kalman inversion, and using global simulations coupled with localized high-resolution simulations to reduce uncertainty in climate predictions.

References:
[1] https://clima.caltech.edu

Earthworks Earthworks is a US NSF-funded collaboration between NCAR and Colorado State University (CSU) to build a global high-resolution Earth System Model with all components using the same mesh. It is assembled from all MPAS components (MPAS-atmosphere from NCAR and MPAS ocean and cryosphere from the E3SM effort) with the unstructured SCVT meshes and coupled within the CESM framework. The individual components have been described above and the uniqueness is developing a high-resolution model in which the model coupling is simplified by utilizing the same mesh for all components. *Readiness:* While this effort is leveraging the efforts at NCAR and in E3SM for exascale, there is additional collaboration with Nvidia for GPU porting and optimization, typically using OpenACC.

E3SM Model The Energy Exascale Earth System Model (E3SM) is a fully coupled variable-resolution Earth System Model developed for the U. S. Department of Energy (DOE) mission needs. The atmosphere is a non-hydrostatic global model that uses continuous Galerkin (or spectral element) methods in the horizontal (3rd order polynomials within an A grid on quadrilateral elements) and staggered (Lorenz grid) finite differences in the vertical with Lagrangian vertical levels. It supports variable resolution meshes based on local refinements of a cubed-sphere mesh. E3SM contains a suite of time-integrators but the methods of choice are 1D nonlinear HEVI methods. The governing equations are written in advective form and use potential temperature as the thermodynamic variable. The ocean, sea-ice and land-ice components are built on the Model for Prediction Across Scales (MPAS) and utilize variable-resolution unstructured Spherical Centroidal Voronoi Tessellations (SCVT). A comprehensive land model and river runoff scheme are also included and can use a finer scale mesh than the atmosphere. To take advantage of newer architectures, the E3SM also supports a Multiscale Modeling Framework (MMF) in which subgrid cloud-resolving models are used to capture finer scale processes (sometimes referred to as superparameterization) *Readiness:* Many parts of E3SM can already utilize GPUs through a combination of C++ frameworks like Kokkos and YAKL in the atmosphere and directive-based approaches (OpenACC, OpenMP) in the land, ocean, and sea-ice components. A significant effort continues to completely rewrite the model in C++ using performance-portable frameworks like Kokkos that provide abstractions with backend implementations supporting a variety of underlying hardware.

GALWEM The Global Air Land Weather Exploitation Model is the operational weather prediction system of the U.S. Air Force. GALWEM is run 4 times per day out to 10 days using the UKMO Unified Model (UM) [2] version 10.9 at 17 km grid resolution that is initialized with conditions provided by the UKMO 4x daily; the UKMO uses hybrid incremental 4dvar for data assimilation.
The UM atmospheric model is a unified regional and global model that uses the compressible (nonhydrostatic) equations and solves them using semi-Lagrangian semi-implicit time-integration with gridpoint numerics on a latitude-longitude grid (with rotated poles). GALWEM has exploited this configuration to run regional windows of both 4 km and 1.5 km horizontal resolutions for specific geographic areas. GALWEM is not exascale capable; however, the U.K. Met Office's next model (LFRic) [3] will be and is expected to replace GALWEM within the next 3-4 years and is expected to be adopted by the USAF. LFRic relies on PSyclone to produce the backend code for various parallel programming models [4]; however, USAF does not have a projected need for exascale modeling at this time.

References:
[1] https://cpaess.ucar.edu/sites/default/files/meetings/2019/documents/McMillen.pdf
[2] https://www.metoffice.gov.uk/research/approach/modelling-systems/unified-model
[3] https://www.metoffice.gov.uk/research/approach/modelling-systems/lfric
[4] https://zenodo.org/record/6078561#.ZDq0aS-B1pQ

GEOS Model System Developed by the Global Modeling and Assimilation Office (GMAO) [1] at NASA Goddard Space Flight Center (GSFC), the Global Earth Observing System (GEOS) [2] provides a seamless modeling tool from climate to weather in support of NASA's evolving Earth-science missions and thereby enhancing national capabilities in Earth system modeling, analysis, and prediction. GOES leverages the hierarchical nature of the Earth System Modeling Framework [3] which enables flexibility in assembling diverse model configurations. The general framework provided by the ESMF is extended by a custom software infrastructure layer, Modeling Analysis and Prediction Layer (MAPL) [4], that connects the ESMF with each individual science component. MAPL provides enhanced capabilities including the management of connectivity and data exchanges between components, default component behaviors, enforcement of GEOS consistency conventions, configurable asynchronous I/O, timings, memory, and performance profiling.

GEOS uses the following components to enable global Earth system predictions across a broad range of spatiotemporal resolutions.

- Atmospheric Dynamics: finite-volume dynamical core (FV3) [5]
- Parameterized Convection: Grell and Freitas (GF) scheme
- Atmosphere Boundary Layer: eddy-diffusivity mass-flux (EDMF) scheme
- Cloud Microphysics: GEOS uses a suite of packages with varying degrees of complexity in aerosol-cloud interactions, namely:
    - For the current retrospective reanalysis, MERRA-2, GEOS uses a simplified prognostic scheme, single moment (1M) for cloud liquid water and ice with a PDF-based scheme for large-scale condensation and cloud cover.
    - The model has been updated to use the GFDL scheme, which is an advanced 1M scheme with a prognostic precipitation (rain, snow and graupel) capability.
- Aerosols: Emissions of aerosols and trace gasses in GEOS are managed via the following:
    - HEMCO which includes user-selectable inventory, flow-dependent, and interactive emissions.
    - GOCART provides a bulk estimate of aerosol mass.
    - MAM estimates both particle mass and size distribution.
- Atmospheric Chemistry
    - GEOS Chemistry-Climate Model
    - Global Modeling Initiative troposphere-stratosphere chemistry module
    - GEOS-Chem code
- Radiation: RRTMG codes
- Gravity Waves
    - State-dependent non-orographic scheme
    - Mountain blocking scheme including anisotropic effects of the subgrid orography
- Land Modeling: GMAO Catchment land model
- Coupled Model: Oceans and Sea-Ice
    - Modular Ocean Model (MOM) version 5
    - Community Ice codE (CICE) version 4
- Ocean Biochemistry: NASA Ocean Biology Model (NOBM)

The GMAO's data assimilation development effort is driven by the diversity of NASA's Earth observations and therefore encompasses multiple components of the Earth system. The GMAO is currently working to develop a Joint Effort for Data Assimilation Integration (JEDI) based unified GEOS data assimilation system to achieve its science and production objectives in weather analysis and prediction, reanalysis, composition forecasting and S2S prediction. JEDI will provide a unified infrastructure for data assimilation that harnesses state-of-the-art object-oriented design practices to build scientific software that is efficient, flexible, and easy to use. This will also enable both GEOS and GFS models to interface with the assimilation code in JEDI on the models' native grids.

Towards an Exascale Hardware Accelerator GEOS The Computational Information and Science Technology Office's (CISTO) Advanced Software Technology Group (ASTG) in collaboration with the GMAO is also undertaking an effort to develop a next-generation GEOS codebase. Recognizing that accelerator-based HPC systems provide a promising platform to meet future GEOS requirements, the ASTG is looking to leverage such systems by incorporating a domain-specific language (DSL) into GEOS. DSL adoption brings an opportunity for code portability of GEOS across multiple architectures, and importantly, will allow GEOS to leverage GPU-accelerated computing systems. It also provides developers a higher-level language that can improve development productivity by abstracting the details of the computing architecture. To jump-start the DSL adoption, ASTG leveraged a GridTools for Python (GT4Py) DSL port of FV3 (gtFV3) developed by the Allen Institute for AI (AI2). The ASTG is developing an approach to incorporate GTFV3 into GEOS and determine whether this approach makes it tractable to achieve performance gains on accelerators over CPUs.

The ASTG is taking incremental steps to develop the accelerator-based GEOS codebase. The first step involved the integration of gtFV3 into GEOS through an interface layer that connects the Fortran-based GEOS codebase with the Python-based gtFV3 using the C Foreign Function Interface (CFFI) for Python. This interface passes relevant data between gtFV3 and the GEOS framework that communicates with other components. The second step involves computing the physics parameterizations on the GPU, and for this part, the ASTG is leveraging OpenACC as the programming model for physics due to its programming flexibility. Although OpenACC has been mostly proprietary to NVIDIA GPUs, the ASTG believes that the OpenACC port can potentially guide development of an OpenMP-offloading implementation if needed due to the relatively similar feature set of both programming models.

ASTG targeted a simple demonstration model using gtFV3 and a simplified physics scheme called Held-Suarez (HS). In creating the model, ASTG achieved the following:

- Ported the Held-Suarez physics scheme to OpenACC and validated the port.
- Developed an interface to integrate gtFV3 into the GEOS framework.
- Tested and validated the gtFV3 core with the GEOS framework.
- Successfully completed a GPU and CPU benchmark of the gtFV3 + HS scheme which demonstrates performance parity between a single CPU socket and a single GPU.

Optimizations are currently underway for the gtFV3 + HS case. This includes modifications to the MAPL framework for device memory management to reduce memory transfers between the hardware accelerator and host, the integration and verification of a more performant GT4Py backend called DaCe, and creation of single precision version of gtFV3. With these modifications, it is expected that the gtFV3 + HS case will outperform the CPU implementation when comparing parity between the number of nodes and number of GPUs used.

References:
[1] https://gmao.gsfc.nasa.gov/
[2] https://github.com/GEOS-ESM/GEOSgcm
[3] https://earthsystemmodeling.org/
[4] https://github.com/GEOS-ESM/MAPL
[5] Putman, Lin, Finite-Volume transport on various cubed-sphere grids, Journal of Computational Physics, Volume 227, Issue 1, 10 November 2007, Pages 55-78, https://doi.org/10.1016/j.jcp.2007.07.022

MPAS Model The Model for Prediction Across Scales (MPAS) is a nonhydrostatic global model that uses finite difference/volume type differencing stencils of 2nd order on Voronoi cells using a C grid. The split-explicit time-integrator based on the three-stage Runge-Kutta method is used to evolve the equations forward in time. The governing equations are written in conservation form with potential temperature as the thermodynamic variable. MPAS allows for the use of variable grids using the spherical centroidal Voronoi tessellation (SCVT) approach. MPAS relies on MPI and OpenACC to run on CPU and Nvidia GPU hardware.

NEPTUNE Model The Navy's Environmental Prediction sysTem Using a Nonhydrostatic Engine (NEPTUNE) is the U.S. Navy's future atmospheric prediction system. NEPTUNE is a deep atmosphere nonhydrostatic unified model based on the NUMA dynamical core and is configurable as either a global weather prediction model, a limited-area regional model, or a 500-km deep thermospheric prediction system. NEPTUNE is a fully compliant NUOPC component and will eventually replace NAVGEM as the atmospheric component of the Earth System Prediction Capability (ESPC). NEPTUNE is a Common Community Physics Package (CCPP) compliant model and a wide range of physics suites are available for testing and evaluation. Initial conditions for NEPTUNE are generated with a variational data assimilation approach through the JEDI interface, as such a tangent linear approximation and adjoint model have been developed for NEPTUNE. NEPTUNE uses an element based Galerkin spatial discretization and Horizontally Explicit Vertically Implicit (HEVI) time integration method to solve the deep atmosphere, non-hydrostatic equations on a hexahedral grid using local 4^{th}-order polynomials (5^{th}-order accurate). The governing dynamic equations are solved in advective form using a continuous Galerkin approach while the transport scheme is formulated in flux form with a discontinuous Galerkin discretization. NEPTUNE relies on MPI and OpenMP to scale on CPU hardware.

References:

[1] https://ams.confex.com/ams/2019Annual/meetingapp.cgi/Paper/349225

NUMA Model The Nonhydrostatic Unified Model of the Atmosphere (NUMA) is a unified global and limited-area deep-atmosphere nonhydrostatic dynamical core that uses continuous and discontinuous Galerkin methods of arbitrary order. All state variables are co-located (A grid). NUMA contains a number of time-integration strategies including fully explicit, fully-implicit (Jacobian-free Newton-Krylov), and implicit-explicit (IMEX). Within these choices of time-integrators, there exist fully three-dimensional (3D) versions or one-dimensional (1D) that can be used in a so-called horizontally explicit vertically implicit (HEVI) approach. For global simulations, the fastest time-to-solutions are given by the 1D nonlinear HEVI and 3D linear IMEX methods (with preconditioning). NUMA contains a number of different forms of the governing equations including both advective and conservation forms, with formulations that are space-weather capable. NUMA can use unstructured grids as long as each element is composed of a hexahedron (cube). NUMA also has the capacity to use statically and dynamically adaptive mesh refinement (AMR). Although NUMA is primarily a dynamical core, it also contains the capability to run with warm rain or ice microphysics. NUMA relies on MPI and OCCA to scale on both CPU and GPU hardware and, more recently, on OpenACC for GPUs. A light-weight version of NUMA (called xNUMA) is also available which is used for experiments in multi-scale modeling framework simulations.

References:
[1] https://frankgiraldo.wixsite.com/mysite/numa
[2] https://frankgiraldo.wixsite.com/mysite/publications
[3] https://frankgiraldo.wixsite.com/mysite/xnuma

SHiELD: The System for High-resolution prediction on Earth-to-Local Domains[1] is developed by the FV3 team at Geophysical Fluid Dynamics Laboratory (GFDL/NOAA). SHiELD uses FV3 as its dynamical core and the Flexible Modeling System (FMS) as its infrastructure and computational framework. FV3 [2,3] uses a multidimensional flux-form advection scheme, its horizontal discretization employs the C-D grid system, allowing for exact diagnosis of cell-mean vorticity and accurate calculation of fluxes. FV3 solves the non-hydrostatic compressible Euler equations on a gnomonic cubed-sphere grid with a Lagrangian vertical coordinate. The algorithm is fully explicit except for fast vertically propagating sound and gravity waves which are solved by a semi-implicit method and the Lagrangian vertical advection. The dycore supports variable grid resolution techniques such as grid stretching and multiple grid nesting.

SHiELD uses a physics package originally adopted from the Global Forecast System (GFS) and heavily updated. SHiELD uses the in-line GFDL microphysics scheme, a Turbulent-Kinetic Energy Eddy-Diffusivity Mass-flux (TKE-EDMF) boundary layer scheme, scale-aware simplified

Arakawa–Schubert (SAS) convection, the Rapid Radiative Transfer Model for GCMs (RRTMG), the Noah land surface model and a mixed layer ocean.

SHiELD development philosophy follows a "one code, one executable, one workflow" unified modeling approach which led to four major configurations, all heavily tested and continuously updated:
(a) SHiELD: global 13km with 91 vertical levels for medium-range global prediction; soon to be upgraded to global 6.5 km for medium-range global extreme weather prediction;
(b) T-SHiELD: global 13km with 63 vertical layers (lowest mid-level at 15m and model top at 64Pa) with a static 3 km nest spanning the tropical North Atlantic for tropical cyclone forecasts with 75 vertical layers (lowest mid-level at 10m and model top at 200Pa), emphasizing the boundary layer;
(c) C-SHiELD: global 13km with a 3 km nest over the contiguous United States (CONUS) and 63 vertical layers (lowest mid-level at ~15 m, model top at 64 Pa) for severe weather prediction;
(d) X-SHiELD: global 3km GCRM with 79 vertical layers.

There are other configurations under development as well, such as, S-SHiELD global 25-km with 91 vertical levels for subseasonal to seasonal prediction and Tele-SHiELD 13km global with 4.3km first nest, and a telescoping 1.4-km nest for Northeast Corridor areas. SHiELD can also run in doubly-periodic mode for kilometer- and hectometer-scale process modeling, and as a kilometer-scale regional climate model.

The FV3 team at GFDL is also developing Pace, which is a Python implementation of SHiELD that uses the GridTools for Python (GT4Py) domain specific language and Data Centric parallel programming framework (DaCe) to achieve performance portability across different computing architectures, including GPUs and CPUs. The initial GT4py port of the FV3 Dynamical Core, PyFV3, was completed by a team at the Allen Institute for Artificial Intelligence and is now maintained by GFDL and co-developed with NASA GSFC's Advanced Software Technology Group. The SHiELD physics parameterizations are currently under development, and the full model will be GPU-capable in FY25. GFDL is also working to integrate Pace into the Flexible Modeling System (FMS) framework to allow it to couple to other GFDL models.

References:
[1] Harris, L., Zhou, L., Lin, S.-J., Chen, J.-H., Chen, X., Gao, K., et al. (2020). GFDL SHiELD: A unified system for weather-to-seasonal prediction. *Journal of Advances in Modeling Earth Systems*, 12, e2020MS002223.https://doi.org/10.1029/2020MS002223
[2] Harris, L., Chen, X., Putman, W., Zhou, L., & Chen, J.-H. (2021). A scientific description of the GFDL Dynamical Core, C, 1–109. https://doi.org/10.25923/6nhs-5897
[3] https://www.gfdl.noaa.gov/fv3/
[4] https://www.gfdl.noaa.gov/shield/
[5] https://github.com/NOAA-GFDL/SHiELD_build/

[6] Dahm et al. (2023) Pace v0.2: a Python-based performance-portable atmospheric model, *Geoscientific Model Development*, 16, 9. https://doi.org/10.5194/gmd-16-2719-2023

SPEAR The Seamless System for Prediction and Earth System Research is a next generation GFDL modeling system for seasonal to multi-decadal projections (i.e., climate model). As a complete climate system, it contains components for the atmosphere (AM4.0), land (LM4.0), ocean (MOM6), and sea ice (SIS2). The atmosphere model (AM4.0) is the same model used in GFDL's CM4 and leverages the hydrostatic FV3 dynamical core. SPEAR also uses the Flexible Modeling System framework (FMS). All of these aspects are shared between GFDL's Seamless Modeling Suite (CM4, ESM4, SPEAR, and SHiELD), four configurations of FMS-, FV3-, and MOM6-based modeling framework.

References:
[1] https://www.gfdl.noaa.gov/spear/
[2] https://agupubs.onlinelibrary.wiley.com/doi/full/10.1002/2017MS001208

UFS Model The Unified Forecast System (UFS) model is the operational weather prediction system of NOAA. The UFS atmospheric component consists of FV3 as the dynamical core with GFS physics and GFDL microphysics, has CCPP physics drivers and runs at C384 (~25km) with 64 vertical levels. The UFS ocean component consists of MOM6 at ¼ degree tripolar grid with 75 hybrid vertical levels. It also uses CICE5 and WaveWatch III. FV3 uses a flux-form semi-Lagrangian method on a CD grid using a cubed-sphere mesh and solves the compressible equations (nonhydrostatic) with Lagrangian vertical coordinates. FV3 can use stretched and refined grids in the horizontal. UFS relies on MPI for running on CPU hardware.

References:
[1] https://dtcenter.org/sites/default/files/events/2021/03-ufs-overview.pdf
[2] https://www.gfdl.noaa.gov/fv3/

4. Assessment Criteria

In this section, we collect benchmark results as a step towards estimating the ability of ESMs to run as fully configured weather or climate models on Exascale computers. As of this writing, there are no full ESMs running on Exascale computers, but we have collected a variety of results designed to shed light on GPU performance, CPU and GPU comparisons, programming models and performance portability, as well as estimated energy consumption.

Since ESMs are composed of several major model components meant to model different parts of the earth system, making assessments of these model components (e.g., dynamics, chemistry, physics, ocean, etc.) will be helpful in understanding the degree of ESM readiness. (For example, dynamics runs on GPUs but chemistry does not). The models typically run multiple operations simultaneously (e.g., ocean and atmosphere components, communications

& computation, I/O and compute, etc.) so there will be limitations in the diagnosis of these performance assessments.

Note that the models represented here are quite different in terms of configuration, complexity and maturity of GPU ports, so data in this report cannot be used for model intercomparison, but is instead meant to show GPU potential and quantify GPU readiness.

We currently have results from three efforts:

1. A full GCRM from the E3SM project, running at 3.25 km global resolution on an Exascale system. These results are used to evaluate E3SM's GPU strategy (C++/Kokkos) across CPU and multiple vendor GPU systems. They demonstrate that exascale systems can achieve better than 1 SYPD on a 3.25 GCRM.
2. NUMA dynamical core benchmarks from NPS, evaluating the Fortran/OpenACC approach to GPU porting. The recent results using OpenACC show that NUMA can achieve results approaching those attained with the OCCA hardware-agnostic API (10% of peak versus 17% for the fastest kernels). Moreover, the GPU results show that codes can run between 5-8x faster simulations on GPUs versus CPU for the same amount of energy consumption. We also see that the increase in performance from the A100 to H100 GPUs is quite substantial (1.4x).
3. GFDL SHiELD Preliminary results with PyFV3 on NVIDIA GPUs

4.1 SCREAM: Simple Cloud Resolving E3SM Atmosphere Model

The E3SM project has a computational mission to run efficiently on DOE's leadership computing facilities. This includes Frontier installed at Oak Ridge National Laboratory and the Aurora system at Argonne National Laboratory. Frontier is the first exascale system in the TOP500 list, with 9472 nodes each with 4 AMD MI250 GPUs. Aurora is even larger with 63744 Intel Ponte Vecchio GPUs. E3SM's GPU strategy is to rewrite the component models in C++ and make use of the Kokkos and YAKL performance portability libraries. Kokkos and YAKL provide parallel *for loops* as well as some hierarchical parallelism and abstract the on-node execution model. They currently support all major CPU and GPU systems, with GPU backends based on the vendor's preferred language (e.g. CUDA, HIP, SYCL).

E3SM has recently completed the port to C++ of their global cloud resolving model SCREAM, and has started work on porting their ocean model which will be called OMEGA.

In this report, E3SM has contributed GCRM benchmarks, taken from Taylor et al., 2023. The benchmarks use the full SCREAM model, running with a nonhydrostatic dynamical core with 10 transported hydrometers. SCREAM uses P3 microphysics and SHOC turbulence and boundary layer scheme and RRTMG++, a C++ port of the RTE+RRTMGP radiation parameterization which computes gas optical properties and radiative fluxes. The benchmark uses the model as it was run for SCREAM's contribution to the DYnamics of the Atmospheric general circulation Modeled On Nonhydrostatic Domains (DYAMOND) model intercomparison project (B. Stevens et al.,

Progress in Earth and Planetary Science, 2019). The benchmarks use prescribed sea surface temperatures, and are coupled to an active land model and land model and sea ice model (but with prescribed ice extent).

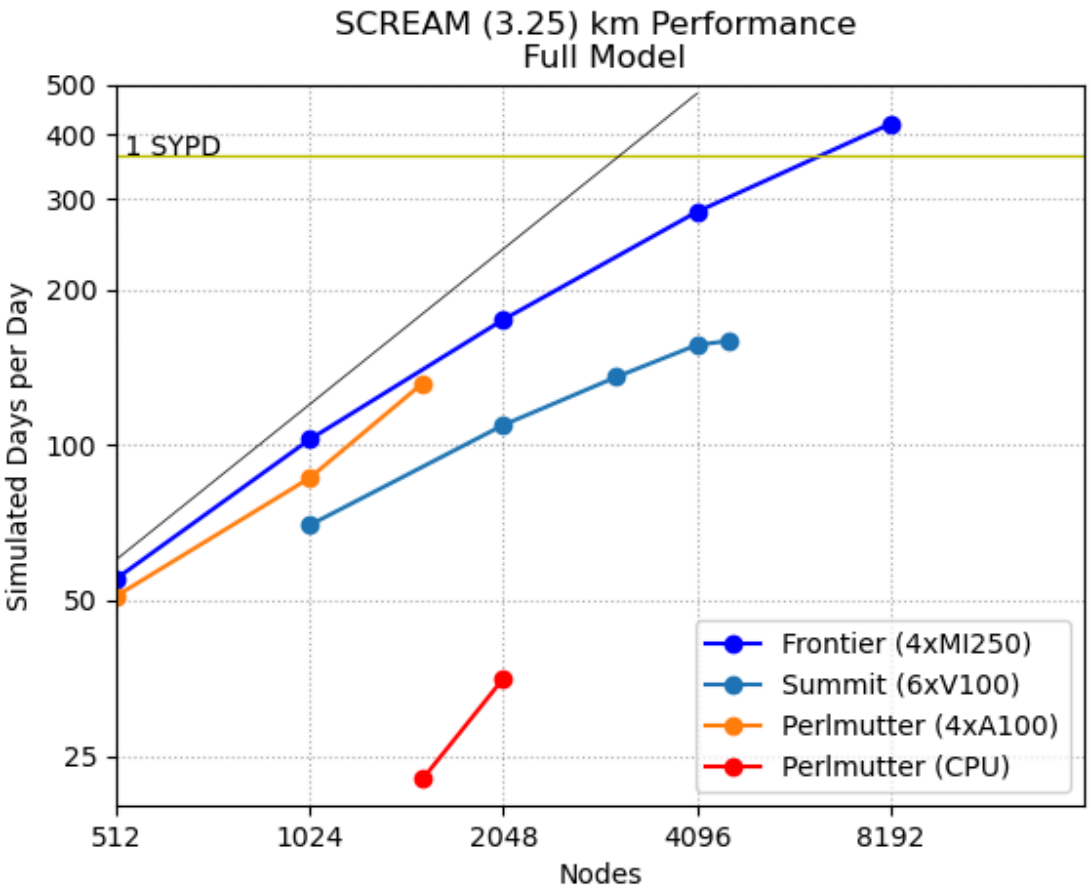

Figure 4.1: Strong scalings of the SCREAM 3.25 km GCRM model running on the exascale Frontier system and two pre-exascale systems.

The SCREAM GCRM results are presented in Figure 4.1. Strong scaling is shown on Frontier, Summit, Perlmutter CPU, and Perlmutter GPU. Frontier is an exascale GPU system with 4 AMD MI250 GPUs per node. Summit is an older GPU system with 6 NVIDIA V100 GPUs per node. Perlmutter has both GPU nodes (1536, each with 4 NVIDIA A100 GPUs) and state-of-the-art CPU nodes (3072 nodes, each with two 64 core AMD EPYC Milan 8863 CPUs). The thin black line shows perfect scaling. Throughput without I/O is measured in simulated-days-per-day, plotted as a function of compute nodes. The results show the performance portability of the C++/Kokkos approach which allows the code to run on CPU systems, multiple NVIDIA GPU systems and an AMD GPU system. On Perlmutter CPU and GPU, the 1536 node result gives a direct comparison between GPU and CPU performance, where the state-of-the-art GPU node (4 A100s, obtaining 132 SDPD) is 5.8x faster than the state-of-the-art dual-socket CPU node (obtaining 22.6 SDPD). In addition, power consumption measurements from the Perlmutter

benchmarks show that while the benchmark is running, the 1536 GPU nodes use 1150 W/node, while the CPU nodes use 690 W/node. Thus on a per-Watt basis, the Perlmutter GPU nodes are 3.5x more efficient than the CPU nodes.

References:

- M.A. Taylor et al., (2023) The Simple Cloud-Resolving E3SM Atmosphere Model Running on the Frontier Exascale System, in SC23: International Conference for High Performance Computing, Networking, Storage and Analysis, Denver CO, US (2023) 10.1145/3581784.3627044.

4.2 NUMA: Results on NVIDIA GPUs

The NUMA group has been experimenting with OpenACC to explore the best way to support Fortran code with minimal changes while still gaining performance improvements on GPUs—currently the only practical route to exascale computing in the U.S., as of this report. In this study, we ported the entire dynamical core to the GPU but only describe the approach for the CreateRhs routine which is representative of all routines used throughout the NUMA code. We decompose CreateRhs into 4 kernels: the gradient (Grad), divergence (Div), global-to-local copy (Glob2Loc) and local-to-global copy (Loc2Glob); these kernels are used for both explicit and implicit time-integration. The copies refer from global gridpoint arrays to local element-wise arrays typical of finite element methods. The kernels are shown in Figure 4.2.1.

```
function OPTIMIZED_RHS
    !$acc kernels
    !$acc loop gang vector collapse(5)
    for e = 1 : N_e, k = 1 : N_ζ, j = 1 : N_η, i = 1 : N_ξ, m = 1 : N_var do
        I = intma(i, j, k, e)
        q_e(m, i, j, k, e) = q(m, I)                          Kernel 1: Global to Local
    end for
    !$acc end kernels

    !$acc kernels
    !$acc loop gang vector collapse(5)
    for e = 1 : N_e, k = 1 : N_ζ, j = 1 : N_η, i = 1 : N_ξ, m = 1 : N_var do
        Compute local derivatives                             Kernels 2 and 3: Gradient and Divergence
    end for
    !$acc end kernels

    !$acc kernels
    !$acc loop gang vector collapse(5)
    for e = 1 : N_e, k = 1 : N_ζ, j = 1 : N_η, i = 1 : N_ξ, m = 1 : N_var do
        I = intma(i, j, k, e)
        rhs(m, I) = f(q(m, i, j, k))                          Kernel 4: Local to Global
    end for
State !$acc end kernels

end function
```

Figure 4.2.1: NUMA Kernels for the right-hand-side vector (rhs).

On the left panel of Fig. 4.2.2, we show the roofline model using an OpenACC implementation on an A100 which has a peak performance of 9.7 teraflops per second (with maximum memory of 80 GB). The blue squares and green diamonds represent the gradient and divergence kernels for different polynomial degrees (N=4,5,6,7,8); we only show one result for the copies because

they give the same result regardless of polynomial degree. On the right panel, we show the results using an OCCA API implementation on the OLCF Titan (see [1]). The dots of the same color represent the results for different polynomial degrees (N=3,4,5,6,7,8); as expected, the higher the degree of the polynomial the better the performance. The best performance achieved so far with OpenACC is around 12% of peak performance whereas for OCCA it sits near 17% (although this comparison includes different kernels and different hardware); however, the OpenACC result achieves nearly twice the performance for the same arithmetic intensity (compare the blue squares on left with the black squares on the right).

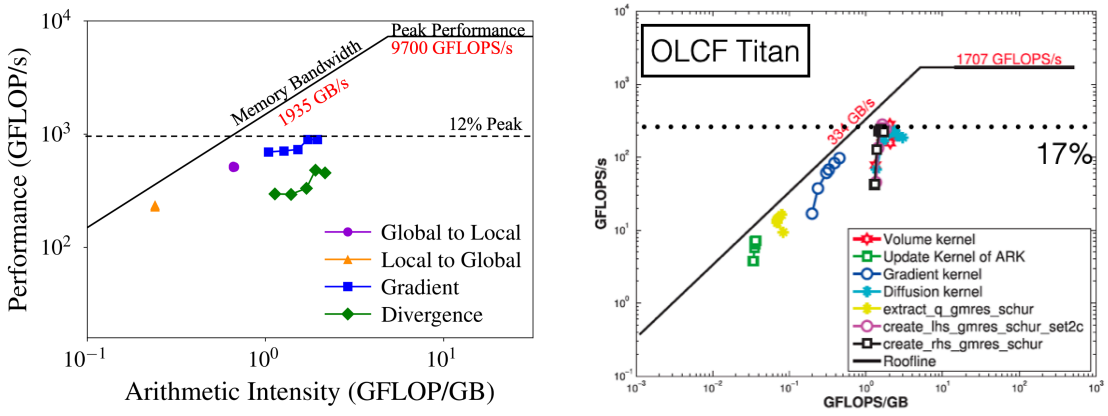

Figure 4.2.2: NUMA Roofline models showing the performance of the main kernels. The left panel shows the OpenACC results on a NVIDIA A100 and the right panel shows results using OCCA on OLCF Titan (NVIDIA K20x).

We compared the energy consumption (in megajoules, MJ) versus problem size for various NVIDIA GPUs including the V100, A100 and H100 cards. In Fig. 4.2.3, the energy consumption reported represents a worst-case scenario where we assume that the hardware utilizes its maximum power consumption.

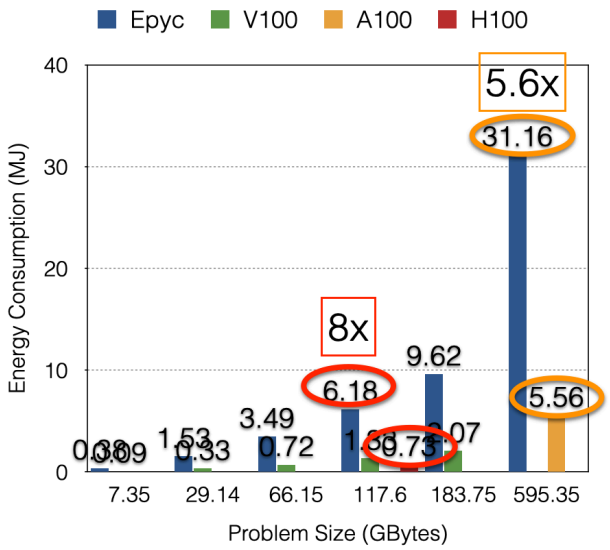

Figure 4.2.3: NUMA energy consumption for various problem sizes comparing a CPU (AMD Epyc) implementation versus a GPU implementation on V100, A100, and H100 hardware.

Figure 4.2.3 compares the performance for each problem size (in gigabytes, GBytes) on an AMD Epyc (Rome) CPU versus a specific NVIDIA GPU card. The red ovals illustrate the comparison for the H100 showing that the CPU requires 8x more energy for the same problem size (~ 117 GBybtes). Another way of stating this is that the H100 is 8x faster than the CPU for the same energy consumed. The yellow ovals denote the results for the A100 showing that the A100 is 5x faster than the CPU. Note that the H100 is about 1.4x faster than the A100 which is an encouraging result especially if a similar trend continues on future cards (we already expect this with the Blackwell GPU).

As a final result, we show in Fig. 4.2.4 strong and weak scaling results for a tropical cyclone simulation run on the NCSA's Delta computer on NVIDIA A100 GPUs with 80 GB HBM2 RAM and NVLink (ranked #256 on the top500 list as of the writing of this report).

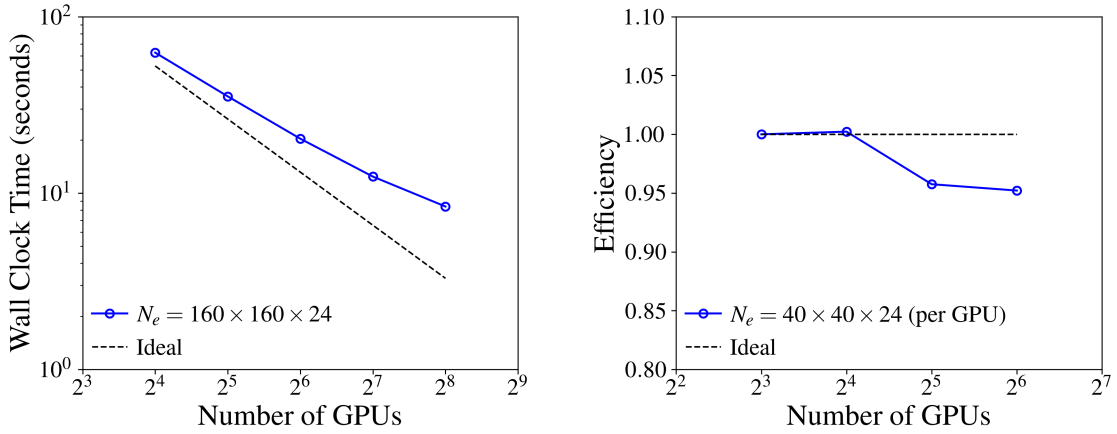

Figure 4.2.4: NUMA strong (left panel) and weak (right panel) scaling on NCSA's Delta using NVIDIA A100 GPUs. On the left panel, we maintain the problem size the same (Ne*(N+1)^3*Nvar=663 million DOF, where Ne=641,400 is the number of hexahedral elements, N =5 is the polynomial degree, and Nvar=5 is the number of variables in the Euler equations). On the right panel we keep the problem size per GPU the same (41.5 million DOF per GPU, with 2.65 billion DOF for the largest problem shown). From ref. [4].

The left panel of Fig. 4.2.4 shows that the GPU code does not achieve perfect strong scaling due to the GPUs not being fully saturated as we increase their number. In contrast, the right panel of Fig. 4.2.4 shows that if we keep the GPUs fully saturated, then we can maintain near perfect weak scaling by keeping the work per GPU constant as we increase their number.
In [4] we show that on Delta, to run the same simulation (per Joule) as one A100 GPU requires 154 AMD Milan cores ; therefore, to run a similar simulation as the 256 GPU runs shown in the strong scaling plot would require 40,000 CPUs (6 times the CPU capability of NSF Delta).

References:
[1] Abdi et al. IJHPCA 2017 https://journals.sagepub.com/doi/10.1177/1094342017732395
[2] Mueller et al. IJHPCA 2018 https://journals.sagepub.com/doi/10.1177/1094342017732395
[3] OCCA https://libocca.org
[4] Kang, Giraldo, and Camp IJHPCA 2025 (to be submitted, see https://frankgiraldo.wixsite.com/mysite/publications for updates).

4.3 GFDL SHiELD Preliminary results with PyFV3 on NVIDIA GPUs
Performance engineering of PyFV3 is ongoing through the Pace project. Figure 4.3.1 shows initial weak-scaling performance of the PyFV3 dynamical core on the Piz Daint supercomputer [1]; the Python code consistently achieves a nearly 4x speedup over the Fortran FV3 implementation and displays near-perfect weak scaling to the entire GPU partition of Piz Daint. Figure 4.3.2 shows strong-scaling results from 6-node simulations on Piz Daint [2]. This illustrates that the Python code scales better on the GPU than the Fortran code does on the CPU once problem sizes are large enough to overcome the GPU launch overhead. The strong-scaling

advantage persists until the GPU's memory is saturated, placing a limit on the domain sizes that can be run efficiently on each chip. Piz Daint's P100 GPUs each contain only 16GB of memory, however; contemporary GPUs with 40-80 GB of memory relax this constraint significantly, allowing for greater speedups over Fortran with more gridpoints per chip. An important note to Figure 4.3.2 is that the CPU backends had not been exposed to the same level of performance engineering at the time of publication, PyFV3 CPU performance is now much closer to the Fortran FV3 code and further optimization will achieve runtimes within 10% on the CPU.

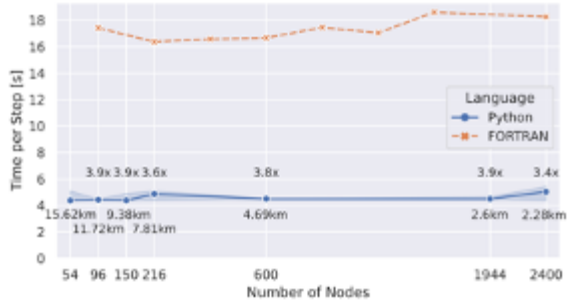

Figure 4.3.1: Node-by-node weak-scaling comparison of the Fortran FV3 and PyFV3 dynamical cores on the Piz Daint supercomputer (NVIDIA P100 and Intel Xeon) from [1]. Numbers below the blue Python line indicates model resolution and numbers above the blue line show speedup over the Fortran dycore.

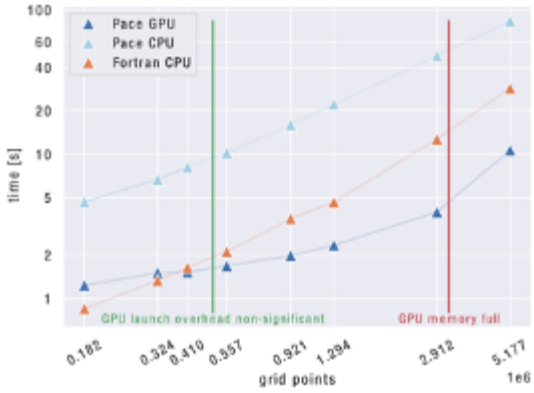

Figure 4.3.2: Runtimes for one invocation of the Fortran and Python dynamical cores run on six Piz Daint nodes from [2]. The vertical lines indicate the problem sizes below which GPU launch overhead dominates runtimes, and above which GPU memory limitations reduce strong scaling of the model.

References:

[1] Ben-Nun et al. (2022): "Productive Performance Engineering for Weather and Climate Modeling with Python" *Supercomputing 2022* https://doi.org/10.48550/arXiv.2205.04148

[2] Dahm et al. (2023) Pace v0.2: a Python-based performance-portable atmospheric model, *Geoscientific Model Development*, 16, 9. https://doi.org/10.5194/gmd-16-2719-2023