HPC for Weather Forecasting

John Michalakes

1 Introduction: Weather and HPC

Numerical weather prediction (NWP) and high-performance computing have grown up together. Even before the computational means existed, L. F. Richardson of the UK Met Office had published a numerical foundation for forecasting the weather [33]. The first computer-generated forecast of the atmosphere had to wait until 1950 and was conducted by Jule Charney's meteorology group within John von Neumann's ENIAC project at Princeton's Institute for Advanced Study. By the 1970s, advances in models and computing capability allowed the skill of numerically generated forecasts to outpace forecasting that relied solely on expert meteorologists interpreting weather observations [17, 38]. Today the list of major weather services that develop and run operational weather forecasting systems includes the European Center for Medium Range Weather Forecasts (ECMWF) and its member national services, the U.S. National Weather Service within NOAA, the U.S. Navy's Fleet Numerical Meteorology and Oceanography Center (FNMOC) and Naval Research Laboratory (NRL), the U.K. Met Office (UKMO), Meteo France, the German National Weather Service (DWD), Environment Canada, the Japan Meteorological Agency, the Korea Meteorological Administration, and the China Meteorological Administration.

Historically, an *exponential* rate of increase in supercomputing power has fueled a *linear* pace of forecast skill improvement (Fig. 1). Each decade's 1000-fold increase in computing power has enabled larger numbers of higher resolution forecasts, better representations of the physics of the atmosphere, and more sophisticated assimilation of greater volumes of observational data to provide better

J. Michalakes (✉)
University Corporation for Atmospheric Research, Boulder, CO, USA
e-mail: michalak@ucar.edu

© Springer Nature Switzerland AG 2020
A. Grama, A. H. Sameh (eds.), *Parallel Algorithms in Computational Science and Engineering*, Modeling and Simulation in Science, Engineering and Technology,
https://doi.org/10.1007/978-3-030-43736-7_10

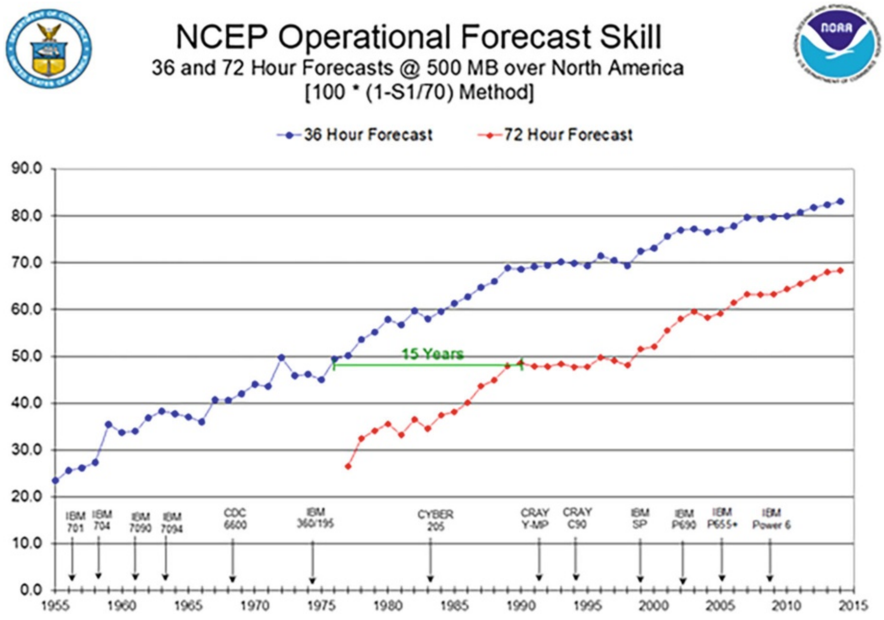


Fig. 1 Anomaly correlation, a measure of forecast skill (100 = perfect), increases linearly as computing increases exponentially over successive generations of supercomputer at the U.S. National Weather Service [20]

initial conditions. The result has been to add 1 day of forecast skill every decade for the last 40 years [3]. Five-day forecasts today are as accurate as 3-day forecasts 20 years ago. Today, twenty of the fastest 500 supercomputers in the world are dedicated to weather forecasting, consuming 7% (60 PFLOPs) of the total compute capacity of the Top500 list in November, 2017[1]. Continuing this trend into the era of exascale supercomputers is the ongoing challenge for weather forecast centers.

Operational weather forecasting involves running a large suite of applications: preprocessors, post-processors, and the model itself (Fig. 2). Preprocessors combine data streaming in from weather stations, aircraft, and satellites with archives of climatological data and previously generated forecasts to produce initial conditions for the new forecast. The forecast model takes this initial state of the atmosphere and computes an approximation of the future state over a succession of many small time intervals until the desired end time of the forecast is reached—as little as 12 h or as long as 16 days, depending on the needs of the center (climate predictions run longer still from seasonal to decadal scales). Periodic output over the course of the forecast is fed into a myriad of post-processors and downstream models that produce specialized products with analysis and visualization for use by forecasters

[1]https://www.top500.org/lists/2017/11/.

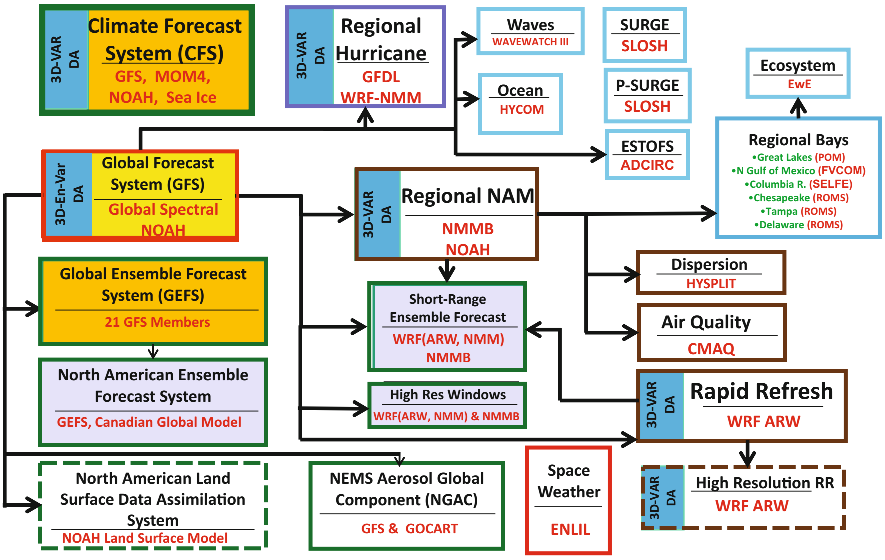


Fig. 2 Production suite at the National Centers for Environmental Prediction (NCEP) in 2014, presented as part of a NOAA annual review. The model itself, the Global Forecast System (GFS), appears as the red box in the first column. Illustration by William Lapenta, NOAA/NWS. Used with permission [44]. The diagram is drawn this way to illustrate the system was becoming too complex. In fact, the system is actually more complex than shown, since data assimilation and other model preprocessors are subsumed within the blue area of the small GFS box

and end users. End-to-end forecasting involves both large amounts of data handling and large amounts of computational horsepower. The focus of this chapter is on the computational requirements and challenges of the forecast model at the heart of the operational weather forecast system.

Models vary according to their use. Climate models simulate characteristics of the atmosphere from seasonal to century time scales at relatively low resolutions. Models designed for real-time weather forecasting run at higher resolution over time scales short enough to fit within the limits of predictability for weather forecasting, from several hours to usually no longer than 2 weeks [43]. The domains for weather and climate models also vary. Models may forecast the entire global atmosphere or a specific region. Global models are constrained by available computing to relatively modest resolutions (currently grid cells no smaller than 9–13 km to a side). Finer than this and the model will not run fast enough for the forecast to be timely[2]. Regional model domains are smaller and can run at higher resolutions but typically require data generated by global models to provide lateral boundary conditions. As computers become more powerful a convergence has begun such that global models

[2]The U.S. National Weather Service requires a forecast rate of 8.5 min per forecast day.

will soon capture finer-scale turbulent and convective processes important for local weather, especially severe storm forecasting.

2 History

The history of NWP is tied closely to the steady but occasionally disruptive evolution of supercomputing over the last half-century. As today's computational scientists scratch their heads wondering how to design efficient codes for exascale systems on the horizon, it is comforting to realize that every previous generation of supercomputer forced scientific programmers to devise codes and data structures tortured in some way to run efficiently on the HPC architecture at hand.

Until the 1990s, supercomputers used for NWP were expensive room-size devices with a single processor. Very fast for their day, speed came from high clock rates (hundreds of megahertz!) and special vector processing units, hardware that could perform many floating point operations over successive data elements during each clock cycle. Today's architectural analogs are vector or SIMD[3] instructions on conventional CPU cores (e.g. Intel's AVX instruction set) and fine-grained parallelism over warps of threads on GPUs. Multi-port memories and high capacity buses were needed to provide the bandwidth necessary to keep up with the processors. These high-performance memory systems contributed further to the already high cost (millions of dollars) of vector supercomputers in the 1970s, 1980s, and 1990s. The impact on software design was also considerable. Weather calculations most naturally expressed in one dimension of the domain had to be rewritten to operate over whatever dimension happened to be vectorizable. For example, subroutines that computed a vertical process such as convection up and down a single column of grid cells had to be rewritten to run horizontally over multiple columns because data dependencies in the vertical inhibited vectorization.

Later, faster but similarly architected systems were developed by connecting several vector processors to the same memory for parallelism over different tasks (task parallelism) or different sections of the domain (data parallelism). This more coarse-grained mechanism, called "microtasking" at the time, is analogous to medium-grain thread parallelism (e.g. OpenMP, pthreads) today. The move to thread parallelism was not overly disruptive since the codes had already been restructured for vectors. Then as now, however, contention for memory bandwidth meant that only a few processors could be added to provide more speed. In other words, the systems could not scale. Ultimately, the cost of building and operating successively faster supercomputers using vector/shared-memory designs became prohibitive.

In the 1990s, a new design for constructing supercomputers from many more less powerful processors pushed past the shared-memory scaling barrier. Processors were organized as nodes on a network, each accessing data exclusively from its own

[3]Single-instruction, multiple-data stream.

memory to avoid the scaling bottleneck. The nodes worked in parallel by exchanging messages over a network, the carrying capacity of which (bandwidth) increased with the number of nodes. Distributed memory message passing was the third and coarsest level of parallelism and could scale to arbitrarily large configurations.

The move to distributed memory supercomputers was unavoidable but deeply disruptive for the NWP community. Significant effort was expended during the 1990s to update models that had been developed for vector supercomputers. Each of the major weather services undertook programs to rapidly convert their large investments in modeling software but struggled with their earlier legacy software designs. The U.S. Department of Energy founded an entire program to convert atmospheric and ocean models used for climate prediction to these new systems [8, 22]. Global address spaces had to be decomposed—that is, broken up—into distributed memory subdomains to be run as separate processes (tasks) over many nodes. Data dependencies needed to be analyzed and explicit mechanisms implemented to buffer and exchange data as messages between separate processes. Entirely new problems of debugging and profiling parallel programs at scale remain areas of active computer science research and development today.

Today's supercomputers are still built as networks of coarse-grain parallel nodes exchanging messages, but also incorporate the other two earlier forms of parallelism: fine-grained within each processor core (vectors or GPU threads) and medium-grained between processors on a node (OpenMP and pthreads).[4] There is no longer any limit to scaling other than money, electrical power, and, more fundamentally, the fraction of parallelism available in the application itself (Amdahl's law). And herein lie both the practical and fundamental disruptions for weather prediction going forward into the exascale era.

The practical disruption is simply that current and next generation supercomputing architectures will require so much parallelism (estimates go to millions of threads) that there is not any level of parallelism that can be ignored: fine-grain vector parallelism at the loop level all the way out to hitherto underexploited coarse-grain distributed memory parallelism over the vertical grid dimension, between different physics subroutines, and between the components (atmosphere, ocean, land, sea-ice, the ionosphere, and other physical systems) in coupled earth system models. In many cases this will mean rediscovering and implementing fine-grained parallelism (discussed in a later section) that was disregarded when microprocessor-based clusters replaced vector supercomputers.

Mining all available parallelism also means rediscovering and implementing shared-memory thread programming, which was largely discarded because the first generations of distributed memory supercomputers had only a single-core processor on each node. Even as nodes with multiple multicore processors have become prevalent, hybrid MPI/OpenMP has only recently begun to show better performance

[4]There is also at least a fourth: instruction level parallelism at the processor core level that exists to some extent even in otherwise outwardly sequential programs. ILP is limited and generally hidden from and outside the control of the programmer.

than parallelizing entirely over single-threaded MPI parallel tasks. This is partly because of improvements to memory systems on new generations of multicore processors and nodes. ECMWF reported optimal performance using eight OpenMP threads per MPI task running the current IFS model on 48 nodes of their Phase-1 Cray XC30 system [46]. But hybrid MPI/OpenMP programming is also taking hold because it will be unavoidable: scaling to larger problem sizes, models run out of pure-MPI parallelism. ECMWF also reported that running the IFS at high resolution was not possible with only MPI parallelism because too much memory was needed to replicate data over many MPI tasks on each node.

The more fundamental disruption from moving to more powerful generations of HPC systems is that future increases in supercomputing speed must come solely from increased parallelism, and that a real-time weather forecast is not weakly scalable. Weak scaling is the ability of an application to run at the same speed using more processors as problem size is increased. For a weather model, increasing problem size means adding grid points and, for a global weather model, the only way to add grid points is to increase resolution. But increasing spatial resolution requires increased temporal resolution: many smaller time steps are needed to produce the same length of forecast. Since time steps must be executed sequentially, complexity increases with resolution in one more dimension, the temporal, than the available parallelism. The cost for higher resolution balloons in terms of number of processors and electricity needed (Fig. 3).

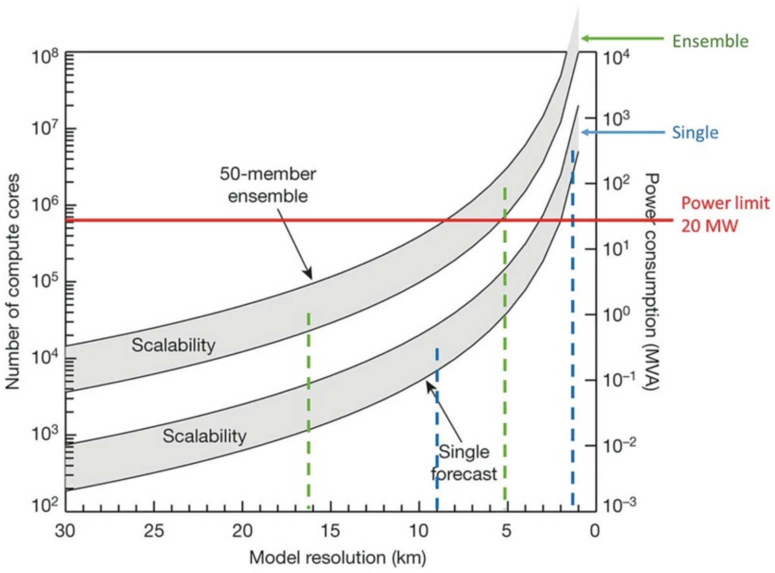


Fig. 3 Projected resources required to scale operational forecasting to higher resolution [3]. Global weather models are not weakly scalable with resolution because the temporal dimension must also be refined and is inherently sequential. Additional notations are from [23]

Note, this lack of weak scalability is the result of running the model *deterministically*: one run from a single set of initial conditions to predict one possible future state of the atmosphere. There is considerable value to consumers from *probabilistic* weather information. For example output from an *ensemble* of many runs of a hurricane forecast, each with a perturbed set of initial conditions, is used to generate a Cone of Uncertainty[5] for where the storm will make landfall. Adding members to the ensemble increases parallelism without constraining the time step, so ensemble forecasts are weakly scalable, at least computationally. The volume of model output generated by the ensemble increases with more members too, so the scaling problem does not disappear but instead shifts to I/O. Moreover, if one also increases the resolution of each ensemble member, as shown in Fig. 3, weak scaling is again problematic.

Parallel-in-time algorithms that can exploit scale separation in partial differential equations to provide parallelism over the time dimension are possible [19] but application to operational weather forecasting is likely distant.

3 Models, Grids, and Parallelization

The specific approach to parallelizing a weather model depends on the choice of numerical scheme and how mapping the mesh onto a spherical geometry is addressed. Various grid geometries and numerical fixes have been developed to adapt "numerical methods to the spherical geometry of the earth, which presents unique problems, usually and vaguely referred to collectively as the pole problem" [48]. For example, in a latitude/longitude grid (Fig. 4) the narrowing of grid cells approaching the two poles requires a smaller time step or unwanted filtering for stability. Icosahedral (soccer ball) meshes have 12 pentagons. Cubed-sphere meshes have corners. The main types of model are grid point, spectral, and finite element (which includes spectral element).

Grid-point models using finite-difference and finite-volume methods evolve the model state (wind velocities, temperature, pressure, moisture, and other tracers) in physical space, directly at each cell of the grid.

Spectral models avoid distortions and singularities by first transforming the gridded representation of the global state to a series of spherical harmonics. Additionally, spectrally computed derivatives are higher-order and non-local, providing more accuracy than finite-difference methods for a given cost. Three dimensional Helmholtz solvers are expensive in grid-point models but essentially free in spectral models [40].

A third type, *spectral element* models are a hybrid formulation: finite-volume methods between the elements and spectral methods local within each element [29].

[5] https://www.nhc.noaa.gov/aboutcone.shtml.

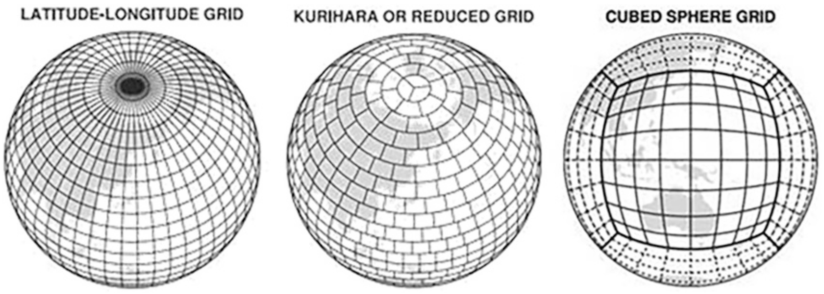


Fig. 4 Sampling of grids used in weather and climate models [48]

The first weather models prior to the 1970s were grid-point formulations, but numeric and computational advantages of globally spectral methods ushered in a heyday that is only now beginning to wane as grid-point methods resurge. Scientific and numerical factors have been central to the progression, but as noted above and in the discussion that follows, a key driver has been the disruptive evolution of HPC architectures. This section derives heavily from [48], an authoritative and to a significant extent eyewitness account of the evolution and types of models used for weather and climate modeling.

3.1 Spectral Dynamics

The spectral transform method was first developed in 1970 [9, 30] and became the dominant dynamical core for global weather and climate modeling for the two decades that followed. Today, ECMWF's world-leading IFS model is the premier example of a spectral model. Other examples include the U.S. National Weather Service's Global Forecast System[6], the U.S. Navy's Global Environmental Model (NAVGEM), and the Japan Meteorological Agency's Global Spectral Model.

Whereas grid-point models represent fields as values at discrete points, spectral models use expansions on a series of spherical harmonics. Each vertical layer of a horizontal field is represented as an M by N array of spectral coefficients. The M dimension corresponds to increasing wave numbers in the zonal (west-east) dimension of the domain; N corresponds to increasing wave numbers in the meridional (equator to pole) dimension. The M and N dimensions of spectral space extend to infinity but are truncated for computational purposes above a certain wave number. If the truncation is the same in both M and N dimensions, it is said to be triangular and has the favorable property of being isotropic and not subject to the

[6]The spectral dynamics in the U.S. weather service's Global Forecast System has reached end of life and has been replaced by FvGFS, a grid point model.

pole problem: discontinuities and time steps constrained by narrowing grid lines near the poles in grid-point models. Higher truncation limits correspond to higher spatial resolution. The spectral dynamics in use in the GFS at the U.S. National Weather Service truncated M and N at wave number 1534, equivalent to a grid of slightly over three million grid points covering the earth's surface at a resolution of 13 km. ECMWF's IFS model achieves finer (9 km) spatial resolution for physics and advection using fewer (1279) waves in M and N through the use of a cubic rather than linear mapping between grid points and the highest frequency wave in spectral space, and by using an octahedral adaptation to IFS's reduced physical-space grid.

Only a portion of a spectral model time step is computed in the spectral domain. Non-linear terms of the Eulerian dynamics, semi-Lagrangian transport, and physics—subgrid-scale radiative heating and cooling, convection, turbulence, surface drag, and other physical processes—are computed on grid points. To move between spectral and grid-point representations, a forward and inverse spectral transform is computed every model time step. The grid point to spectral transform first applies an FFT to each west-east circle of grid values along latitude lines of the domain, producing vectors of Fourier coefficients that correspond to wave numbers in the M spectral dimension. Next, a Legendre transform is applied in the equator-to-pole dimension to construct the N dimension of spectral space. Each resulting m,n spectral coefficient is the sum of the products of the m element of each Fourier vector times the Legendre coefficient for the Gaussian latitude from which the Fourier vector was computed. The Legendre transform is algorithmically equivalent to a matrix multiply, and can be implemented using calls to DGEMM in LAPACK.

The computational complexity of the combined N Fourier transforms is $O(N^2 \log N)$, where N is the truncation number. Overall, the spectral transform is dominated by the $O(N^3)$ complexity of the Legendre transform. ECMWF has been able to devise a "Fast" Legendre Transform (FLT) that is $O(N^2 \log N^3)$. The FLT exploits similarities of associated Legendre polynomials at all the Gaussian latitudes but with different wave number and then precomputes and reuses an approximate representation of the matrices. The FLT is less efficient than DGEMM at lower resolutions but breaks even and continues to improve at T2047 and higher [45].

Sensitivity to rounding error requires that Gaussian weights used in the Legendre transform be computed using double (64 bit) floating point precision, even when other parts of the model are computed at lower precision [7]. Other operations in the spectral transform and other parts of the model may be computed using single (32-bit) floating point precision but weather centers are only now beginning to explore reducing precision for better computational efficiency [31].

The spectral transform method has advantages for parallel computing and software engineering because virtually all computations in spectral dynamics are dependency-free and perfectly parallel, both over wave components in spectral space and in the two horizontal dimensions of physical grid space. From a software point of view, the parallelism in a spectral model is highly encapsulated. Code to implement message passing is compact and isolated to within the subroutines that transform back and forth between grid and spectral space each model time step.

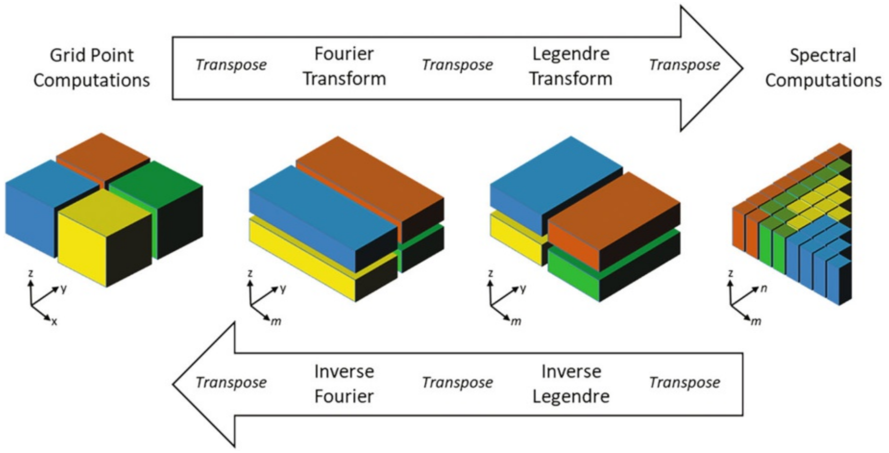


Fig. 5 Schematic of the forward and reverse spectral transforms in a time step of a spectral model, showing the decompositions of the physical, Fourier, and spectral domains over four tasks with transposes between the decompositions

Parallelizing the spectral transforms involves interprocessor communication that can be implemented in either of two ways: distribute the FFTs and DGEMMs themselves or transpose the data between decompositions to allow serial Fourier and Legendre transforms to be used. The advantage of using distributed FFT and DGEMM packages is that parallelism is built-in, the routines are portable and are likely to have been optimized for the computational platform. On the other hand, transpose implementations of spectral transforms are more flexible and general, permitting the use of non-power of two serial FFT packages. Transposes also ensure identical order of operations in the transforms giving results that are bit-for-bit reproducible on different numbers of MPI tasks. Generally, transpose implementations are favored because they send less data than distributed implementations, an advantage on systems where transfer (bandwidth) costs are high relative to message startup costs [11]. Figure 5 shows parallel transposes and data layouts for one time step of a typical spectral transform model.

Modern spectral models incorporate semi-implicit semi-Lagrangian transport (SLT) for advection because SLT is unconditionally stable and allows longer time steps. As with fully Lagrangian methods, SLT involves calculating the trajectories of parcels of a fluid over time; the difference being that SLT interpolates forward or backward trajectories relative to a fixed grid at each time step.

The issue for parallelizing SLT occurs when parcels flow to an area of the domain on another processor. As with finite-difference and finite-volume methods, these dependencies and dependencies associated with interpolation stencils are addressed by communicating with neighboring tasks to update halo- or ghost-regions around a task's subdomain. Anisotropy of the domain closer to the poles will require more data to be sent to update increasingly wide halos. Fortunately, the reduced grid used for computational efficiency elsewhere in the model (Fig. 4) also helps address the communication costs near the poles for SLT.

Spectral models have been extremely successful since the 1970s and are still deployed in major forecast centers. Nevertheless, the spectral method is approaching obsolescence on new generations of supercomputers that will require applications that can exploit 10^5 to 10^6 way parallelism without losing efficiency to parallel overheads such as interprocessor communication. Of course, any model of fluid flow requires communication between processors, but communication cost for local methods such as explicit finite-difference and finite-volume remains constant with increasing domain sizes and numbers of processors. Cost for non-local communication in spectral models increases as a function of domain size. Communication cost for high-resolution 5 km and 2.5 km experimental runs of ECMWF's IFS on up to a quarter-million processor cores of the TITAN supercomputer at Oak Ridge National Laboratory reached 75% of the total cost of the spectral transforms [46]. ECMWF estimates that stopgap improvements such as the Fast Legendre transform, the cubic grid-to-spectral mapping, and an octahedral grid-reduction geometry can extend the spectral IFS model's life for a time, but are exploring grid-point formulations for scaling to higher resolution [42].

3.2 Grid-Point Dynamics

The first computer models of the atmosphere were grid-point models, which flourished during a period of active development beginning with the U.S. Joint Numerical Weather Prediction Unit in the 1950s and lasting until a two-decade hiatus around the advent of spectral transform models in the 1970s. A key focus was to address the pole problem inherent in Cartesian latitude-longitude grids, leading to development of novel and promising quasi-uniform mesh geometries such as those shown in Fig. 4. These included composite and overset grids, icosahedral and geodesic grids, reduced latitude-longitude grids, Fibonacci grids (these were later), and regular polyhedra circumscribed to the sphere, most commonly the cubed sphere. The new approaches were generally successful at addressing the pole problem but presented other issues for solution quality: noise and interpolation error at boundaries of overset meshes, the edges and corners of faces on the cubed sphere or at the 12 pentagons in hexagonal meshes. Numerous schemes to reduce or eliminate these issues were developed and the topic remains an active focus of research and development today.[7]

Generating grids that are composed of Cartesian grids involves projecting the component grids onto curvilinear coordinates of the sphere. The cubed-sphere grid in the U.S. National Weather Service's next model, FvGFS, is composed of six

[7]The PDEs on the Sphere workshop series (https://pdes2017.sciencesconf.org) have focused on the problems of grids and numerical methods for weather, climate, and ocean circulation since 1990.

Cartesian grid faces of a cube inflated out to the surface of an enclosing sphere. The global version of the NCAR Weather Research and Forecast (WRF) model [41] is an overset mesh scheme comprised of two Cartesian meshes, one covering each hemisphere and then projected onto polar stereographic coordinates. In both cases, and aside from the extra work involved to handle the corners and edges on the cubed sphere and the overlap regions of the overset mesh, the component grids themselves are Cartesian and straightforward from a coding point of view. Traversing the domain and accessing values for neighboring grid cells is done using array index arithmetic inside multiply nested loops that are easily recognized and optimized by modern compilers.

Approaches for non-Cartesian grids are more complex and interesting from numerical, geometric, and computational points of view. The Non-hydrostatic Icosahedral Model (NIM) developed at NOAA uses an icosahedral mesh constructed mostly of hexagons with 12 pentagons. The Model for Prediction Across Scales (MPAS) developed by NCAR and Los Alamos National Laboratory uses a centroidal Voronoi tessellation (CVT) of arbitrary polygons that aligns to an icosahedral hexagonal mesh but that allows further in-place refinement (Fig. 6) to focus higher resolution over an area of interest: for example, the Gulf of Mexico and western Atlantic during hurricane season. The CVT in MPAS obeys the additional constraint that lines connecting neighboring cell centers bisect the neighbor edges and intersect at right angles. This supports an unstructured generalization of Arakawa C-grid staggering used to overcome problems with the representation of gravity waves in collocated grids while addressing problems reproducing geostrophic balance that stem from the discretization of the Coriolis force [36].

The unstructured horizontal dimensions in the MPAS grid are represented as arrays of vertical columns of the domain. The relationships between adjacent cell centers, edges, and vertices are computed when a mesh is generated and stored as integer arrays for each column. Traversing the grid and accessing neighbor values

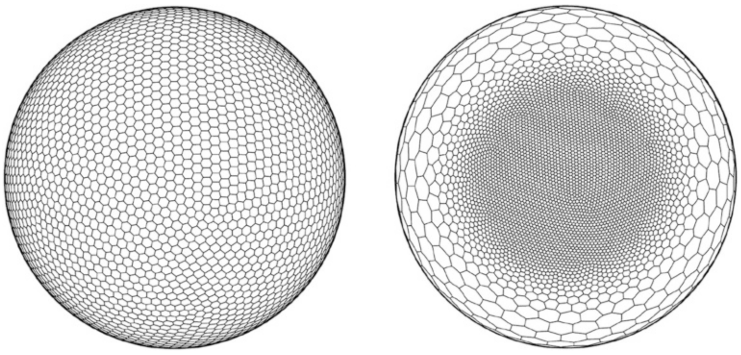


Fig. 6 Centroidal Voronoi Tessellation (CVT) of a quasi-uniform (left) and variable resolution MPAS mesh. The meshes shown contain the same number of grid cells [32]. © American Meteorological Society. Used with permission

Algorithm 1 (Lloyd method) *Given a domain Ω, a density function $\rho(\mathbf{x})$ defined on Ω, and a positive integer n.*

```
0. Select an initial set of n points $\{\mathbf{x}_i\}_{i=1}^n$ on $\Omega$ ;
1. Construct the Voronoi regions $\{V_i\}_{i=1}^n$ of $\Omega$ associated with $\{\mathbf{x}_i\}_{i=1}^n$;
2. Deterimine the (constrained) mass centroids of the Voronoi regions
   $\{V_i\}_{i=1}^n$; these centroids form the new set of points $\{\mathbf{x}_i\}_{i=1}^n$;
3. If the new points meet some convergence criterion, return $\{(\mathbf{x}_i, V_i)\}_{i=1}^n$
   and terminate; otherwise, go to step 1.
```

Fig. 7 Lloyd algorithm for constructing centroidal Voronoi tessellations used in the MPAS model [35]

for computation require indirect indexing, a computational penalty compared to iterating over Cartesian meshes. This impact can be offset by ordering grid points in memory to be stored successively in vertical vectors that can be vector-parallel on CPUs [25] and thread parallel on GPUs.

Relative to Cartesian meshes, generating unstructured meshes is complicated and expensive and is typically done offline. The MPAS grid generation program is based on a method originally developed at Bell Laboratories for signal processing (Lloyd 1982) and applied to generating CVTs on the sphere (Fig. 7). The Lloyd method is sequential and essentially trial and error, so that creating a global mesh at a new quasi-uniform resolution or generating a global mesh with new areas of refinement may require days of computer time to converge[8]. Fortunately, once generated, the meshes can be reused, rotating to a different orientation if necessary to expose a different part of the domain to the area of mesh refinement. Improvements in mesh-generation speed have been obtained using GPUs and work to improve the quality and speed of generated CVTs is ongoing [21] (Engwirda 2017 JIGSAW-GEO).

3.2.1 Domain Decomposition

Once generated, decomposing unstructured grids involves finding a partitioning that assigns approximately equal numbers of grid columns to MPI tasks for load balance while minimizing the surface area to adjacent partitions to minimize the volume of data communicated. The MPAS model uses the METIS package from the University of Minnesota to decompose its domain over tasks (Fig. 8).

[8]Skamarock, W., personal communication.

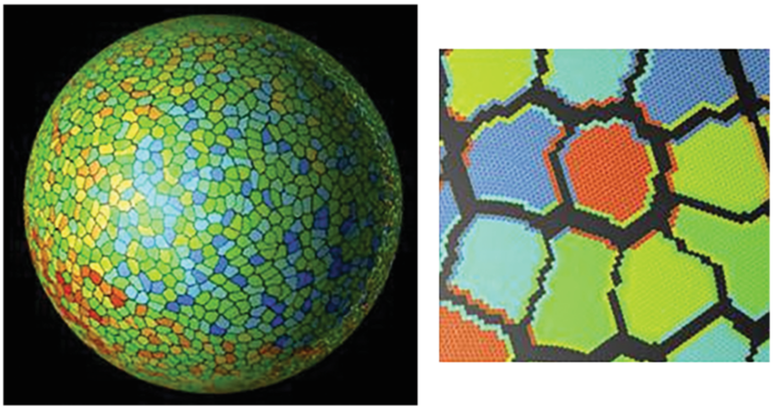


Fig. 8 MPAS unstructured grid decomposition that minimizes the amount of overlap between subdomain edges, thus minimizing the amount of data that must be communicated. Reproduced with permission [34]. Used with permission. Partitioning was generated using the METIS package from U. Minnesota [24]

METIS uses recursive bisection and K-way partitioning to find a decomposition that minimizes computational imbalance which results from uneven distribution of work and communication imbalance which results from edge-cuts between vertices of the mesh.

The icosahedral mesh of the NIM model consists of the ten rhombus-shaped faces and is decomposed over tasks in two steps. First, each face is assigned to a separate set of MPI tasks. Then each face is decomposed over its set of tasks in checkerboard Hex[9]-board? fashion. The decomposition originally required NIM to run on multiples of 10 MPI tasks. Subsequent refinements using 20 rhombuses allowed multiples of 1, 2, or 5 MPI tasks. Ordering the grid columns on each task in spirals eliminated the need copy cells into buffers to send and receive data to neighboring tasks through MPI (Fig. 9).

Space filling curves have been used to order and decompose elements in the HOMME and NEPTUNE spectral element models on cubed-sphere grids [4, 6].

3.2.2 Load Imbalance

Partitioning may also need to account for varying amounts of work for a given column depending on location in the domain and the time of the simulation. Sources of load imbalance can be static or can vary over the course of a simulation. An example of a static imbalance occurs when the number of processors does not divide the number of grid columns without a remainder. Or with limited-area

[9]https://www.hexwiki.net.

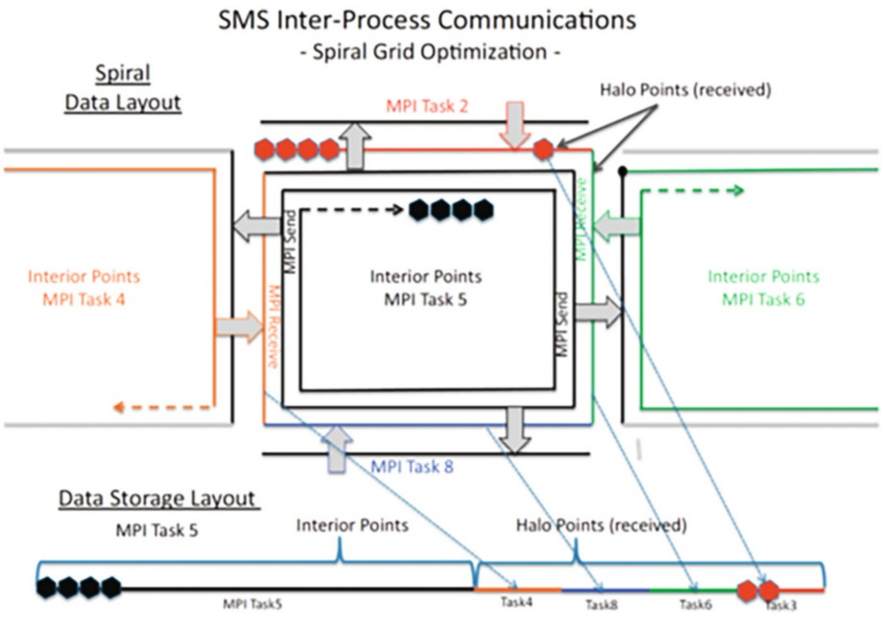


Fig. 9 Spiral traversal of icosahedral grid cells for MPI tasks in the NOAA NIM model, allowing interior data and data stored for interprocessor halo exchanges stored contiguously in memory. The ordering allows exchange sections of the computational storage to be passed directly to MPI without additional copies. © American Meteorological Society. Used with permission [18]

(not global) domains, the computations on the lateral boundaries may involve less work. Static imbalances can be addressed by assigning different numbers of grid columns to processors. Dynamic imbalances may be associated with processes such as cloud physics that require more computation around convective systems (storms).

Arriving at a perfectly balanced load is usually not possible. A weather model is multi-phasic, performing different physical or dynamical processes in the course of a time step and also different mixes of these processes from one time step to another. Since each phase of computation may have a different load profile, using an optimal decomposition to balance each phase is impractical. Inefficiency from load imbalance in the 5–20% range is usually not enough to justify the cost of redistributing work and data between each phase. An exception is the imbalance associated with the diurnal cycle in radiation physics that is computed only in the sunlit half of the domain. Here the imbalance is large and regular enough for load balancing to provide a benefit, even with the cost for relocating the data [10, 37].

3.3 Element-Based Dynamics

Finite element and spectral element methods, used widely in aerospace and other applications of computational fluid dynamics, are being applied to weather modeling because they are high order local methods that scale well computationally. Each element computes a local solution to the desired level of accuracy using an expansion on a set of orthogonal basis functions, not unlike the calculations done globally in spectral models described above. The element-local solutions are combined to form a global solution using either a continuous or discontinuous Galerkin method: the local solution at points along the faces of each element is summed with the edge solutions of the element's neighbors in a process called direct stiffness summation (DSS). DSS requires only nearest neighbor communication and the amount of data communicated is constant with respect to numerical order. Thus, element-based methods provide the accuracy and high computational intensity of globally spectral methods but without domain-wide interprocessor communication that inhibits scalability. Element-based methods are also well suited to complex geometries and lend themselves to adaptive mesh refinement [2, 12, 14, 15, 29]. Examples of models using element-based dynamical cores include the NUMA[10] dynamical core in the U.S. Navy's NEPTUNE and the HOMME dynamical core used in the Community Earth System Model[11] and the Department of Energy's Energy Exascale Earth System Model (E3SM)[12]. The UK Met Office is developing the finite element Gung-Ho[13] dynamical core for its new LFRic[14] modeling system.

Benchmarking NOAA's Next Forecast Model

In 2015 the National Weather Service needed to replace its aging Global Spectral Model. Six dynamical cores from development teams in the USA were evaluated: NOAA/GFDL's FV3, NOAA/NCEP's NMM-UJ, NOAA/ESRL's NIM, NCAR's MPAS, and Naval Research Laboratory's NEPTUNE based on the Naval Postgraduate School's NUMA model. FV3, NMM-UJ, and NEPTUNE used a cubed-sphere grid; NIM and MPAS used icosahedral/unstructured. Numerically, NMM-UJ used finite-difference; FV3, NIM, and MPAS were finite-volume; and NEPTUNE/NUMA used spectral elements. ECMWF's spectral/semi-Lagrangian IFS was included for comparison.

Computational performance and scaling were benchmarked on Edison, a large Cray supercomputer at the Department of Energy's NERSC facility. The first chart shows performance results for the models running a 13 km resolution workload (up to 3.5 million cells). The horizontal dotted at 1.0 is the speed threshold for forecasting. The second chart shows strong scaling efficiency for a higher resolution

[10]http://faculty.nps.edu/fxgirald/projects/NUMA/Introduction_to_NUMA.html.

[11]http://www.cesm.ucar.edu.

[12]https://e3sm.org/.

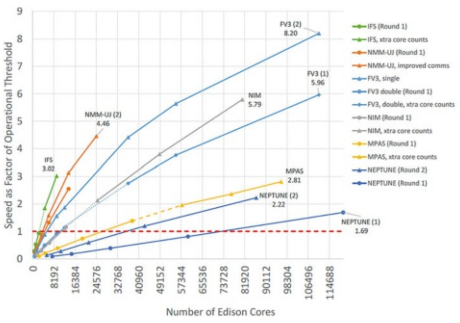
[13]https://www.metoffice.gov.uk/research/foundation/dynamics/next-generation.

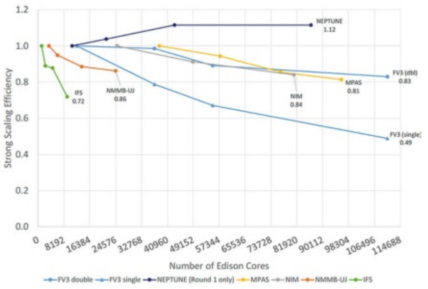
[14]https://www.metoffice.gov.uk/research/modelling-systems/lfric.

3 km resolution workload (up to 65-million cells) expected to be commonplace within the next decade.

The fastest models scaled the least well, a not unexpected result. Computationally heavy models like MPAS and NEPTUNE perform more work per processor making the overhead from communication proportionately less costly. Non-local communication in IFS's spectral transforms hindered its scaling. FV3 ran 1.36 times faster (and scaled less well) at single precision than double precision and gave acceptable results [27].

The evaluation concluded in 2016 with selection of GFDL's FV3. The reports from all phases of testing are available online from the National Weather Service.





(https://www.weather.gov/sti/stimodeling_nggps).

For a given forecast configuration, element-based methods are more costly in terms of floating point operations than finite-difference and finite-volume based approaches but provide greater accuracy and scalability to large numbers of parallel threads on current and next generation HPC architectures. The NUMA spectral element dynamical core was the first ever to achieve operational forecast speed at a uniform global resolution of 3 km (1.8 billion cells), scaling with 99% efficiency to the full 786-thousand cores of the IBM Blue Gene/Q Mira system at Argonne National Laboratory [28]. In NOAA's 2015 intercomparison to choose the next dynamical core for the U.S. National Weather Service, the NUMA/NEPTUNE dynamical core was the most costly but also the most efficient running up to the full number of processors available (see "Benchmarking NOAA's Next Forecast Model").

3.4 Physics

The parts of a weather model that provide forcing terms that drive atmospheric dynamics—radiative heating, evaporation, condensation, convection, chemistry, turbulence, surface drag, and other physical processes—are collectively known as physics (the usage may be singular or plural). Physics differentiate weather and climate models from more general computational fluid dynamics applications.

Physics packages in a model are parameterizations because they are simplified representations of processes that occur at subgrid scales, too fine to be resolved by the dynamics. Physics is where much of the predictive skill of a model resides. Adapting a physics package to a particular forecast application involves tuning—adjusting parameters within the physics package—to remove forecast error and biases at a given forecast scale with respect to observations.

Physics usually represents processes that act only in the vertical dimension, and is perfectly parallel between adjacent columns in the horizontal domain dimensions; however, physics work-per-column depends on the state of the atmosphere and is a major source of load imbalance. There are opportunities for parallelism over different physics packages—for example, running radiative transfer concurrently with convention and other physics. Parallelism in the vertical dimension is typically limited or non-existent.

The computational cost of physics is a significant fraction of the overall cost of a model run, anywhere from 20% to half of a typical forecast depending on the configuration. Radiative transfer (Fig. 10) and cloud microphysics (Fig. 11) are typically the most expensive physics components unless chemistry is also employed. In that case, for air quality and pollution predictions, the cost of simulating chemical reactions in the atmosphere and for advecting large numbers of chemical tracers

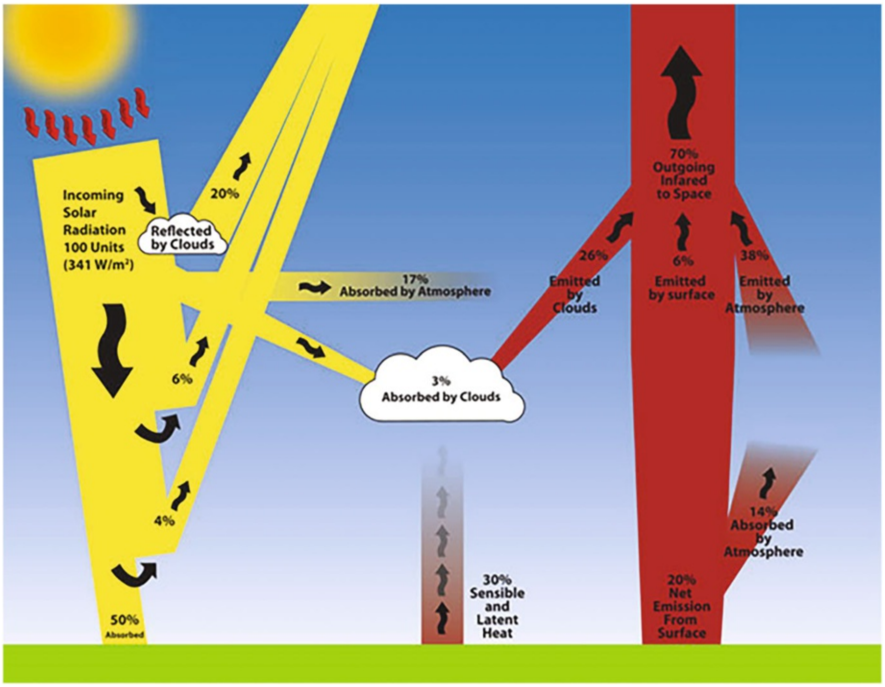

Incoming Solar Radiation 100 Units (341 W/m²)
Reflected by Clouds
20%
17% Absorbed by Atmosphere
6%
3% Absorbed by Clouds
4%
50% Absorbed
30% Sensible and Latent Heat
70% Outgoing Infared to Space
26% Emitted by Clouds
6% Emitted by surface
38% Emitted by Atmosphere
14% Absorbed by Atmosphere
20% Net Emission From Surface

Fig. 10 Heating and cooling from incoming shortwave and outgoing longwave solar radiation as modeled by the Rapid Radiative Transfer Model. Illustration by AER Corp. Used with permission

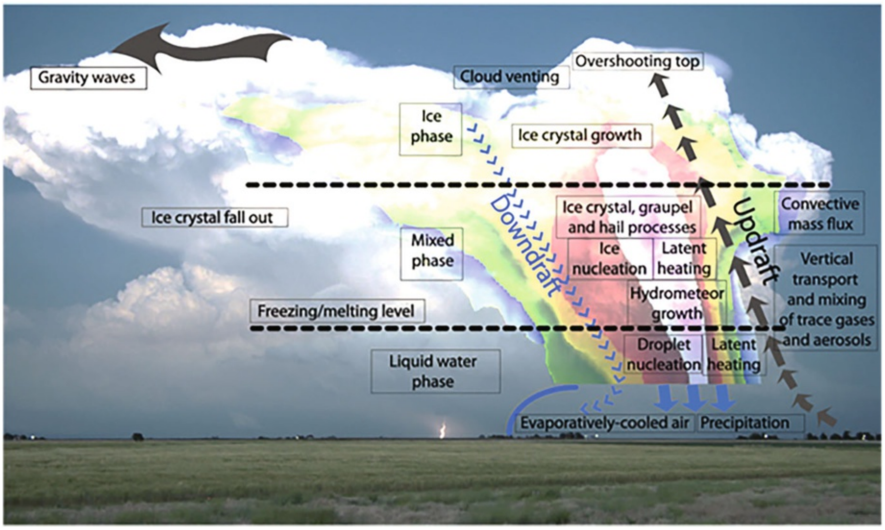


Fig. 11 Cloud microphysics models subgrid-scale moisture processes governing production of precipitation in multiple forms and thermodynamic feedbacks from evaporation and condensation in active convection. Illustration by Rob Seigel, Colorado State University. Used with permission

may be several times greater than the cost of the entire rest of the model. Physics is also state-heavy. While dynamics requires no more than a half-dozen or so prognostic variables per grid cell, the combined working set for a full-physics meteorological application is at least an order of magnitude larger (two orders larger with chemistry). In spite of large working sets, physics makes greater use than dynamics of exponent, log, square root, power, and other intrinsic operations and is therefore more computationally intense than the model overall.

From a software point of view, physics code must be updated more frequently than dynamics and is a source of inconsistency in a model's software repository. The physics packages within a given model may have been developed and contributed by groups of experts outside a model development team using different vertical coordinates, representations of physical fields, and coding practices.

4 Challenges for Next-Generation HPC

HPC systems are increasingly out of balance. Floating point capability is increasing but the usable percentage is decreasing because only the number of floating point units that can be constructed and powered for a given area of silicon and watt of electricity continues to increase exponentially (and that may end soon). Rates of increase for memory system, network, and I/O performance have slowed. The 8 billion transistor Knights Landing (KNL) processor, Intel's most recent (and last) generation of Intel's Many-Integrated Core (MIC) architecture, was rated at up to three TFLOPs peak performance (2.2 TFLOPs measured). To achieve that, however, an application would need to perform seven floating point operations for every byte accessed from KNL's high-bandwidth (700 GB/s) MCDRAM memory. The number one ranked system on the Top500[15] list at this writing was the 200 PFLOPs Summit system at Oak Ridge National Laboratory, which comprises 28,000 NVidia Volta (V100) GPUs. To reach the rated 7 TFLOPs peak performance on the V100 GPU, an application must perform nine operations for every byte accessed. By contrast, the highest computational intensity (CI) measured for a full NWP model (non-spectral transform) is 0.7 operations per byte [28], an order of magnitude gap between application intensity and realizable floating point performance that is widening with time.

One may argue that realized percentage of peak performance is an artificial metric and that time-to-solution is what matters. If a model is scalable, why not use larger numbers of processors to reach the required simulation speed? In the first place, as discussed above, real-time deterministic weather forecasting does not weakly scale with resolution because the sequential temporal dimension must also be refined. But even within this fundamental scaling limit, there are also sound practical reasons to worry about efficiency. Although it may be possible to run a forecast using 15–20 MW of electricity, wasting all but a few percent is difficult to justify. And application parallelism itself is a limited resource. Scaling to more tasks and threads without using the resources available to each thread efficiently leaves performance on the table and limits additional speedup unnecessarily.

Roofline analysis [47] characterizes realizable performance in terms of how an application maps to the memory system and computational capabilities of a processor. The idea, illustrated in the roofline plot for a Knights Landing (KNL) processor (Fig. 13), is that performance (vertical axis) is bound by memory system performance in the sloping part of the roofline. In that region, the memory system cannot provide operands fast enough to keep up with the floating point units of the processor. When CI is high enough and the roofline is flat, the application is bound only by its ability to saturate the speed of the floating point units. The several sloping parts of the roofline in the figure correspond to levels of the KNL memory system from the fastest and smallest level one cache out to the DRAM main-memory on the KNL device. The memory image of a weather model is too small to fit entirely within the L1 and L2 caches but does fit within the 16 GB MCDRAM, the high-bandwidth (nominally 400 GB/s, 387 GB/s measured) on-chip memory of the KNL. Thus, MCDRAM bandwidth is the limiting factor on KNL for applications with CI of <7 FLOP/byte. The shaded area shows how little of the KNL's peak performance is used by weather models with an overall CI of <1. Optimization involves restructuring loops and data structures to increase memory locality, moving CI to the right; and then increasing vector utilization to use as much of the increased headroom under the roofline as possible.

[15]June 2018 Top500 list, https://www.top500.org/lists/2018/06/.

The example in the figure is the roofline plot of an expensive subroutine from the NEPTUNE model's profile that diffuses energy cascading to wave numbers too high to be resolved. The routine already has better than average CI, but additional improvement was obtained using an AoS to SoA (array of structures to structure of arrays) transformation. The original version of the code copied from a model array into local spectral element arrays and back again. Fields in the state array were stored together for each point (AoS), so that traversing a field required non-stride-one accesses. In the optimized code, fields for each element were stored in an element structure (SoA). The element structures were stored as an array (AoSoA) that replaced the original state array. The diffusion routine was modified to be called over each element structure in the state array and compute using the field data in place without copying in and out of the routine. This optimized memory restructuring moved CI to the right from 1.1 to 1.5 FLOP/byte.

Remaining optimization involved increasing vector and FMA (fused multiply-add) utilization by reorganizing loops to make it easier for the compiler to generate vector and FMA instructions, nearly doubling performance. A key benefit of roofline analysis is signaling when the programmer has more work to do. In this case, Fig. 12 shows considerable unexploited headroom remained beneath the 700 GFLOPs roofline for MCDRAM, suggesting other sources of inefficiency, for example, incomplete vector utilization, instruction latency, or load imbalance between threads.

Figure 13 shows the end result of a several month cycle of AoS to SoA and other optimizations to improve CI and vectorization in the NEPTUNE model. The solid bars are performance of the unoptimized code for a workload small enough for a single node running on generations of Intel Xeon processors and Cavium Corp's ARM-based ThunderX2 processor. The hatched bars show the increase in simulation rate (simulated time over wall clock) after optimization.

4.1 Next Generation HPC and the Programming Challenge

In the absence of increasing clock rates, effort now focuses on-processor fine-grained parallelism: threads on GPUs and vector instructions on CPU cores. On GPUs, approaches have ranged from inserting OpenACC or OpenMP directives to offload computation to the GPU to complete recoding into NVidia's CUDA programming language. Because of the difficulty of generating and maintaining a separate GPU version, the only instance of an entire weather model converted to CUDA by hand was the Japan Meteorological Agency's ASUCA model [39]. The authors showed their code running 80 times faster on the GPU, but with caveats. The comparison was relative to original Fortran code running on a single CPU core. Moreover, single-precision GPU performance was compared to performance of the original code at double precision. Taking this into account, one estimates that a node-for-node GPU to CPU comparison with equivalent configurations would have

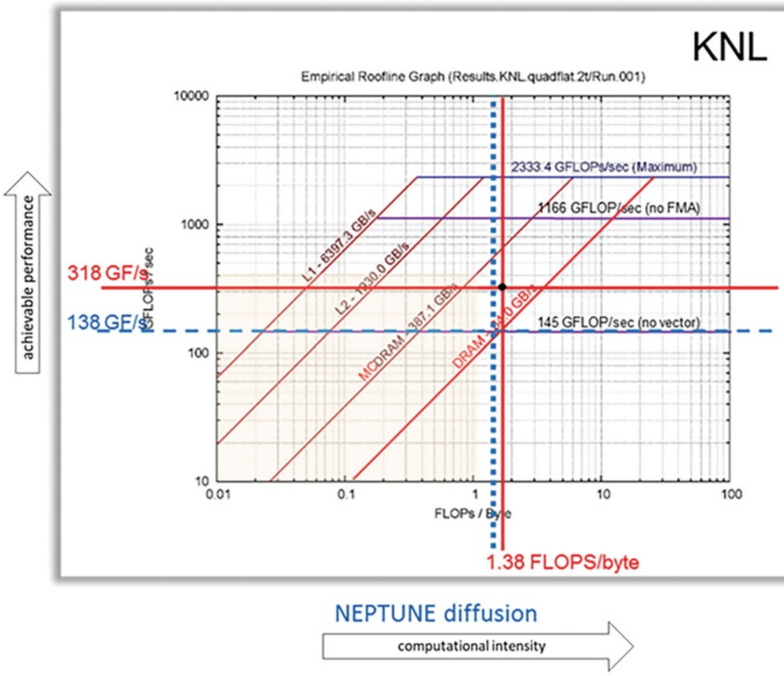


Fig. 12 Roofline plot for a costly diffusion routine in the NEPTUNE spectral element model. The plot was generated by running the UC Berkeley's Empirical Roofline Toolkit on an Intel Xeon Knights Landing processor, then annotated with computational intensity (CI) and performance that was measured for the diffusion kernel. Dotted lines show original measurements, solid show after optimization. The shaded box shows the portion of the KNL's theoretical peak performance that can be utilized by weather models having overall CI of <1.0

yielded two- to four-times speedup, a ratio that has remained consistent with other NWP codes on successive generations of hardware.

Directive-based approaches using OpenACC and OpenMP allow code to be implemented, maintained, and optimized on both CPU and GPU architectures. NOAA's Earth System Research Laboratory developed a single-source implementation of the NIM model (one of the models described in the box) using OpenACC directives [18]. The authors showed a two to three times performance benefit for GPUs compared to conventional multicore CPUs and 1.3 times compared to MIC on a device-to-device basis. This was without accounting for the additional cost of moving data between the GPU device and its host processor. The authors showed up to a 2× GPU to CPU benefit in terms of hardware cost in dollars, after internode communication and overhead for transferring data between the host and GPU device were addressed.

Meteo Suisse has deployed a GPU version of the COSMO model that was implemented using Gridtools (formerly STELLA), a domain specific framework of C++ templates and libraries developed by the Swiss National Supercomputing

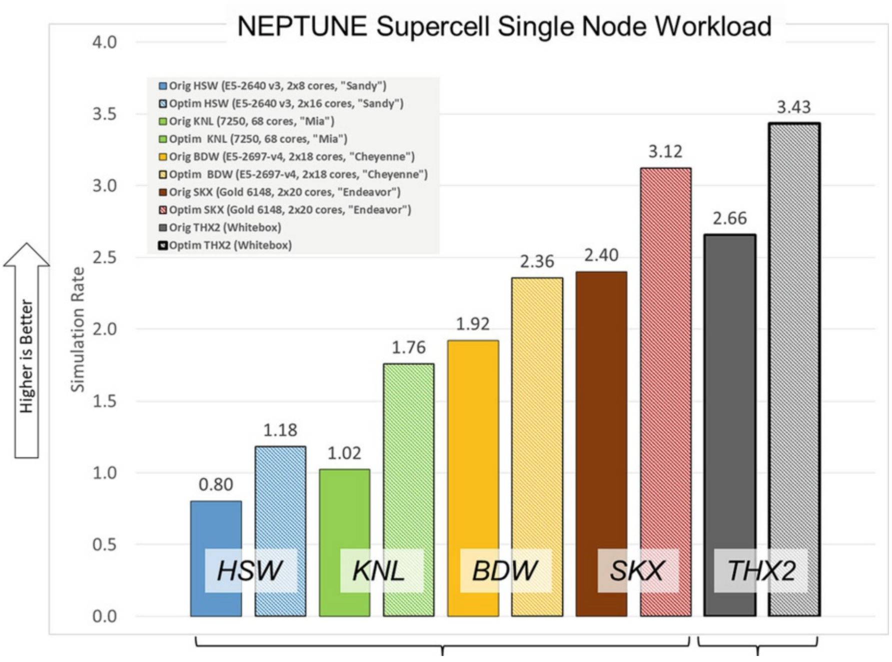


Fig. 13 Original and optimized performance (as simulation rate) of the NEPTUNE model on single-node workload over a successive generations of multicore CPUs

Center. Gridtools uses template metaprogramming to embed the DSL within the C++ host language. At compile time the DSL is translated into an executable with OpenMP threading on CPU architectures and CUDA for GPUs [13]. Physics in the COSMO model was adapted to GPU outside of Gridtools using OpenACC. The authors reported between two and three times faster performance on the GPU compared to multicore CPU, depending on the amount of work (number of grid points) per node.

On the MIC architecture, application speedups relative to conventional multicore CPUs are similar to speedups seen on GPU, but with considerably less programming effort. This is because vector and parallel programming on the Knights Landing is fundamentally the same as for conventional multicore Xeon processors. WRF and other models able to use both MPI for message passing and OpenMP for threading ported easily to MIC. Programmers can focus attention on exploiting fine-grain parallelism, usually by helping the compiler recognize and generate vector instructions and by restructuring code and data to make more efficient use of cache and memory. The Knights Landing version of the MIC ran a standard WRF benchmark 1.7 times faster than one node (two sockets) of an Intel Xeon (Broadwell) processor [16].

Porting and optimizing NWP codes for next generation architectures remain areas of active effort and research, and are the focus of numerous conference and workshop series.[16], [17], [18] The European Union's Energy-efficient Scalable Algorithms for Weather Prediction at Exascale (ESCAPE) is a 3-year project to address the problem for weather and climate services in the EC, stated as follows:

> Existing extreme-scale application software of weather and climate services is ill-equipped to adapt to the rapidly evolving hardware. This is exacerbated by other drivers for hardware development, with processor arrangements not necessarily optimal for weather and climate simulations.[19]

A key activity within ESCAPE has been to identify and package kernel benchmarks called Weather and Climate Dwarfs, after the original Berkeley Dwarfs [1], to focus co-design efforts between the applications and HPC research and manufacturing communities. In the USA, the HPC working group of the multi-agency Earth System Prediction Capability (ESPC) program comprises model developers and users from NOAA, NASA, DOE, the Dept. of Defense, and the National Science Foundation. Less far along than the European efforts, at this writing the ESPC group had defined initial requirements on which to undertake effort along the lines of the EC program [5].

5 Summary

> "Modern weather prediction is perhaps the most cooperative activity of our species"
> – Prof. Clifford Mass, Dept. of Atmos. Sciences. U. Washington [26]

Today, the numerically generated weather information available through print, radio, television, and the internet is public's most direct experience with high-performance computing. The half-century history of numerical weather prediction is a story of massive scientific and technical investment on an international scale; of steady progress fraught with technological disruption harnessing a billion-fold increase in computer power; and of the challenges for continuing to add value from numerically generated forecasts into the exascale era.

Acknowledgements Thank you to Tom Henderson, Kevin Viner, Jim Doyle, John Dennis, Michael Duda, Jacques Middlecoff, and Jordon Powers for reading the manuscript and providing their valuable advice.

[16]NCAR Multicore Workshop series: www2.cisl.ucar.edu/events.

[17]ECMWF Workshop on HPC in Meteorology: events.ecmwf.int.

[18]AMS Symposium on HPC for Weather, Water and Climate: ams.confex.com.

[19]http://www.hpc-escape.eu/.

References

1. Asonovic, K., Bodik, R., Catanzaro, B., Gebis, J., Husbands, P., Keutzer, K., … Yellick, K. (2006). *The landscape of parallel computing research: a view from Berkeley.* Electrical Engineering and Computer Sciences. University of California at Berkeley. Retrieved from http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html
2. Bao, L., Nair, R., & Tufo, H. (2014). A mass and momentum flux-form high-order discontinuous Galerkin shallow water model on the cubed-sphere. *Journal of Computational Physics, 271*, 224-243.
3. Bauer, P., Thorpe, A., & Brunet, G. (2015, September 3). A quiet revolution of numerical weather prediction. *Nature, 525*, 47-55. Retrieved from www.nature.com/doifinder/10.1038/nature14956
4. Burstedde, C., Wilcox, L., & Ghattas, O. (2011, May). p4est: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees. *SIAM J. Sci. Comput., 33*(3), 1103-1133. Retrieved from https://doi.org/10.1137/100791634
5. Carman, J., Clune, T., Giraldo, F., Govette, M., Gross, B., Kamrath, A., … Whitcomb, T. (2017). *Position paper on high performance computing needs in earth system prediction.* ESPC position paper, National Earth System Prediction Capability, Silver Spring, MD. doi:10.7289/V5862DH3
6. Dennis, J. (2003). Inverse space-filling curve partitioning of a global ocean model. *Proceedings fo IEEE International Parallel and Distributed Processing Symposium*, (p. 7). doi:10.1109/IPDPS.2003.1213486
7. Drake, J., Flanery, R., Semararo, D., Worley, P., Foster, I., Michalakes, J., … Williamson, D. (1995). *Parallel Community Climate Model: Description and User's Guide.* Oak Ridge National Laboratory.
8. Drake, J., Semeraro, B., Worley, P., Foster, I., Michalakes, J., Toonen, B., … & Williamson, D. (1994). *PCCM2: A GCM adapted for scalable parallel computers.* Chicago, IL: Argonne National Laboratory. Retrieved from https://www.osti.gov/biblio/10114472
9. Eliasen, E., Machenhauer, B., & Rasmussen, E. (1970). *On a numerical method for integration of the hydrodynamical equations with a spectral representation of the horizontal fields.* Report No. 2, University of Copenhagen, Institute for Teoretisk Meteorologi.
10. Foster, I., & Toonen, B. (1995). *Load Balancing Algorithms for the Parallel Community Climate Model.* Technial memorandum ANL/MCS-TM-190, Argonne National Laboratory.
11. Foster, I., & Worley, P. (1997). Parallel algorithms for the spectral transform method. *SIAM Journal on Scientific Computing, 18*, 806-837. doi:10.2172/10168301
12. Fournier, A., Taylor, M., & Tribbia, J. (2004). A spectral element atomsopheric model (SEAM). *Monthly Weather Review, 132*, 726-748.
13. Fuhrer, O., Osuna, C., Lapillonne, X., Gysi, T., Cumming, B., Bianco, M., … Schulthess, T. (2014). Towards a performance portable, architecture agnostic implementation strategy for weather and climate models. *Supercomputing Frontiers and Innovations, 1*(1), 45-62. doi:10.14529/jsfi140103
14. Gaberšek, S., Giraldo, F., & Doyle, J. (2012). Dry and moist experiments with a two-dimensional spectral element model. *Mon. Wea. Rev., 140*, 3163-3182.
15. Giraldo, F., & Rosmond, T. (2004, January). A scalable spectral element Eulerian atmospheric model (SEE-AM) for NWP. *Monthly Weather Review, 132*, 133-153.
16. Gokhale, I., & Michalakes, J. (2016). Weather Research and Forecasting (WRF). In J. Jeffers, J. Reinders, & A. Sodani, *Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition* (pp. 499-509). Morgan Kaufman.
17. Golding, B., Mylne, K., & Clark, P. (2004). The history and future of numerical weather prediction in the Met Office. *Weather, 59*(11), 299-3-6. doi:10.1256/wea.113.04
18. Govett, M., Rosinski, J., Middlecoff, T., Lee, J., MacDonald, A., Wang, N., … Duarte, A. (2017). Parallelization and performance of the NIM weather model on CPU, GPU and MIC

processors. *Bulletin of the American Meteorology Society © American Meteorological Society. Used with permission*, 2201-2213.

19. Haut, T., & Wingate, B. (2014). An Asymptotic Parallel-in-Time Method for Highly Oscillatory PDEs. *SIAM Journal on Scientific Computing, 32*(2), 693-713. Retrieved from https://doi.org/10.1137/130914577
20. Henderson, T., Michalakes, J., Gokhale, I., & Jha, A. (2015). Numerical weather prediction optimization. In J. Reinders, & J. Jeffers, *High Performance Parallelism Pearls, Volume 2* (pp. 7-23). Morgan Kaufman.
21. Jacobsen, D., Gunzburger, M., Ringler, T., Burkardt, J., & Peterson, J. (2013). Parallel algorithms for planar and spherical Delaunay construction. *Geosci. Model Dev., 6*, 1353-1365. doi:10.5194/gmd-6-1353-2013
22. Jones, P. (1996). The Los Alamos Parallel Ocean Program (POP) and Coupled Model on MPP and Clustered SMP Architectures. In G.-R. Hoffman, & N. Krietz (Ed.), *Seventh ECMWF Workship on the Use of Parallel Processors in Meteorology* (pp. 226-238). Reading, UK: World Scientific.
23. Källén, E. (2016). Weather Prediction and the Scalability Challenge. *EASC 2016: Exascale Applications & Software Conference*. Stockholm, Sweden. Retrieved from https://youtu.be/WAqR4aUzpgo
24. Karypis, G., & Kumar, V. (1998). *METIS 4.0: Unstructured graph partitioning and sparse matrix ordering system.* University of Minnesota, Dept. of Computer Science and Engineering. University of Minnesota. Retrieved from http://glaros.dtc.umn.edu/gkhome/metis/metis/overview
25. MacDonald, A., Middlecoff, J., Henderson, T., & Lee, J. (2010). A general method for modeling on irregular grids. *International Journal of High Performance Computing Applications*. Retrieved from http://journals.sagepub.com/doi/abs/10.1177/1094342010385019
26. Mass, C. (2014, December 25). Is Numerical Weather Prediction One of Mankind's Greatest Achievements? Retrieved from http://cliffmass.blogspot.com/2014/12/
27. Michalakes, J., Govett, M., Benson, R., Black, T., Juang, H., Reinecke, A., & Skamarock, W. (2015). *AVEC Report: NGGPS Level-1 Benchmarks and Software Evaluation.* Technical report, NOAA, Office of Science and Technology Integration. Retrieved from https://www.weather.gov/sti/stimodeling_nggps_implementation_atmdynamics
28. Müller, A., Koera, M., Marras, S., Wilcox, L., Isaac, T., & Giraldo, F. (2018, April 5). Strong scaling for numerical weather prediction at petascale with the atmospheric model NUMA. *International Journal of High Performance Computing Applications*, 31. Retrieved from https://doi.org/10.1177/1094342018763966
29. Nair, R., Thomas, S., & Loft, R. (2005, April). A discontinuous Galerkin global shallow water model. *Monthy Weather Review*, 876-888. Retrieved from https://doi.org/10.1175/MWR2903.1
30. Orszag, S. A. (1970). Transform method for calculation of vector coupled sums: Application to the spectral form of the vorticity equation. *Journal of Atmospheric Science, 27*, 890-895.
31. Palmer, T., & Düben, P. (2014, August). The use of imprecise processing to improve accuracy in weather & climate prediction. *Journal of Computational Physics, 271*, 2-18. Retrieved from https://doi.org/10.1016/j.jcp.2013.10.042
32. Raucher, S., Ringler, T., Skamarock, W., & Mirin, A. (2012). Exploring a Global Multi-Resolution Modeling Approach Using Aquaplanet Simulations. *Journal of Climate © American Meteorological Society. Used with permission*, 2432-2452. doi:10.1175/JCLI-D-12-00154.1
33. Richardson, L. (1922). *Weather Predication by Numerical Process.* Cambridge University Press.
34. Ringler, T. (2018). Personal communication.
35. Ringler, T., Ju, L., & Gunzburger, M. (2008). A multiresolution method for climate system modeling: application of spherical centroidal Voronoi tessellations. *Ocean Dynamics, 58*(5-6), 475-498. doi:10.1007/s10236-008-0157-2

36. Ringler, T., Thuburn, J., Klemp, J., & Skamarock, W. (2010, May). A unified approach to energy conservation and potential vorticity dynamics for arbitrarily-structured C-grids. *Journal of Computational Physics, 229*(9), 3065-3090. Retrieved from https://doi.org/10.1016/j.jcp.2009.12.007
37. Rodrigues, E., Navaux, P., Panetta, J., Fazenda, A., Mendes, C., & Kale, L. (2010). A comparitive analysis of load balancing algorithms applied to a weather forecast model. *22nd International Symposium on Computer Architecture and High Performance Computing.* Petropolis, Brazil. doi:10.1109/SBAC-PAD.2010.18
38. Schuman, F. G. (1989). History of Numerical Weather Prediction at the National Meteorlogical Center. *AMS Weather and Forecasting, 4*, 286-296. Retrieved from https://journals.ametsoc.org/doi/pdf/10.1175/1520-0434%281989%29004%3C0286%3AHONWPA%3E2.0.CO%3B2
39. Shimokawabe, T., Aoki, T., Muroi, C., Ishida, J., Kawano, K., Endo, T., … Matsuoka, S. (2010). An 80-fold speeup 15.0 TFlops, full GPU acceleration of non-hydrostatic weather model ASUCA production code. *Proceeddings of the 2010 ACM/IEEE conference on Supercomputing (SC'10).* New Orleans, LA.
40. Simmons, A., Burridge, D., Jarraud, M., Girard, C., & Wergen, W. (1989). The ECMWF medium range prediction models, development of the numerical formulations and the impact of increased resolution. *Meteorol. Atmos. Phys., 40*, 28-60.
41. Skamarock, W., Klemp, J., Dudhia, J., Gill, D., Barker, D., Wang, W., & Powers, J. (2005). *A description of the advanced research WRF version 2.* Technical report. Retrieved from http://www.dtic.mil/docs/citations/ADA487419
42. Smolarkiewicz, P., Deconinck, W., Hamrud, M., Kühnlein, C., Mizdzynski, G., Szmelter, J., & Wedi, N. (2015, Autumn). An all-scale, finite-volume module for the IFS. *ECMWF Newsletter*, pp. 24-29. Retrieved from https://www.ecmwf.int/en/elibrary/14589-newsletter-no-145-autumn-2015
43. Stern, H., & Davidson, N. (2015, October). Trends in the skill of weather prediction at lead times of 1-14 days. *Q. J. R. Meteorol. Soc.*, 2726-2736. doi:10.1002/qj.2559
44. Tolman, H. (2017). *The production suite: looking forward.* National Oceanic and Atmospheric Administration, OSTI, College Park, MD. Retrieved from http://www.emc.ncep.noaa.gov/annualreviews/day-2/01a-Tolman_NPSR_2017_townhalls.pdf
45. Wedi, N. P., Hamrud, M., & Mozdzynski, G. (2013). A fast spherical harmoics transform for global NWP and climate models. *Monthly Weather Review, 141*, 3450-3461.
46. Wedi, N., Bauer, P., Deconinck, W., Diamantakis, M., Hamrud, M., Kühnlein, C., … Smolarkiewicz, P. (2015). *The modeling infrastructure of the Integrated Forecast System: Recent advances and future challenges.* Technical Memorandum 760, European Centre for Medium-Range Weather Forecasts, Reading, UK. Retrieved from https://www.ecmwf.int/sites/default/files/elibrary/2015/15259-modelling-infrastructure-integrated-forecasting-system-recent-advances-and-future-challenges.pdf
47. Williams, S., Waterman, A., & Patterson, D. (2009). Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM, 52*, 65-76. Retrieved from https://www.osti.gov/servlets/purl/963540
48. Williamson, D. (2007). The Evolution of Dynamical Cores for Global Atmospheric Models. *Journal of the Meteorological Society of Japan. Used with permission, 85B*, 241-269. Retrieved from https://pdfs.semanticscholar.org/5a42/471e95ec434e0eb08bf03380da4a578c420d.pdf