Performance Comparison of the A-grid and C-grid Shallow-Water Models on Icosahedral Grids

December 10, 2021

JACQUES MIDDLECOFF
Cooperative Institute for Research in the Atmosphere, Colorado State University, Fort Collins, and Global Systems Laboratory, NOAA/Earth System Research Laboratories, Boulder, Colorado

YONGGANG G. YU
National Center for Atmospheric Research, Boulder, Colorado

MARK W. GOVETT
Global Systems Laboratory, NOAA/Earth System Research Laboratories, Boulder, Colorado

ABSTRACT

This study uses a single software framework to compare the CPU performance of Arakawa A-grid (NICAM) and C-grid (MPAS) schemes for solving the shallow-water equations on icosahedral grids. The focus is on high resolution weather prediction. Performance analysis shows the simpler structure of the A-grid equations enable compiler optimization-based efficiency gains that the C-grid equations cannot match. Strong scaling runs at 3.5km resolution show the A-grid is three times faster than the C-grid, enabling the A-grid to run at 50% higher resolution in only 15% more time. A performance comparison with the MPAS shallow-water model is included which demonstrates that our software implementation of the C-grid is robust and comparisons are fair.

1. INTRODUCTION

Many dynamical core systems have adopted the icosahedral geodesic grid for global and regional simulations of weather and climate, e.g. ICosahedral Nonhydrostatic Model (ICON; Wan et al. 2013; Zangl et al. 2015), Nonhydrostatic ICosahedral Atmospheric Model (NICAM; Satoh et al. 2008), Model for Prediction Across Scales (MPAS; Skamarock et al. 2012), and the Nonhydrostatic Icosahedral Model (NIM; Lee and MacDonald 2009). The companion paper, "Comparing Numerical Accuracy of Icosahedral A-grid and C-grid Schemes in Solving the Shallow-Water Model" (Yu et al. 2020) describes a single software framework to compare the Arakawa A-grid (NICAM) and C-grid (MPAS) schemes for using the finite volume method (FVM) to solve the shallow-water equations on an icosahedral grid. The companion paper focuses on comparing the accuracy, stability, and conservation of mass, momentum, and energy. This paper extends the work by Yu et al. (2020) to evaluate the computational efficiency of the two methods. The work relies on the single software framework introduced by Yu et al. (2020) to make the comparison between the A and C grids as fair as possible. Comparison is also made between the C-grid implementation introduced here and the MPAS shallow-water model to ensure the C-grid implementation is robust.

This paper is organized as follows: Section 2 discusses the shallow-water equations and their significance. Solutions to the shallow-water equations are introduced using the Arakawa A and C grid schemes applied to the Icosahedral grid on a sphere. Section 3 introduces a single software framework used to discretize both the A and C grid schemes. Discretization and parallelization methods are also described. Performance and accuracy of the A-grid, C-grid, and MPAS implementation of the C-grid shallow-water model is compared and analyzed in Section 4. A conclusion is given in Section 5.

2. METHODOLOGY OVERVIEW

The shallow-water equations are a set of partial differential equations describing the essential wave propagation found in atmospheric circulation. In shallow-water flows the dimension in the vertical is much smaller than in the horizontal so the vertical dimension can be removed by averaging over the depth. Implementing a shallow-water model to study flow in a single atmospheric layer is a first step in dynamical core development, as appeared in the literature (e.g., FV3, NICAM, MPAS, etc.). In this paper the FVM and the icosahedral grid are used to solve the shallow-water equations on a sphere which comprise the continuity and the Euler equations:

$$\frac{\partial h}{\partial \tau} = -\nabla\cdot(hV),$$

$$\frac{\partial V}{\partial \tau} = -(f+\xi)k\times V - \nabla\left[\frac{1}{2}V^2 + g\left(h+h_s\right)\right],$$

where h represents fluid depth, τ denotes time**,** **V** is the velocity vector, $f = 2\Omega sin(\theta)$ is the Coriolis parameter, $\xi = k \cdot \nabla \times V$ is relative vorticity, and h_s is the underlying topography.

Grid staggering schemes were introduced by Winninghoff (1968) and summarized by Arakawa and Lamb (1977). In this paper we evaluate the computational cost of two popular schemes: the A-grid and the C-grid. The A-grid scheme is the only unstaggered Arakawa type grid and refers to the method in which prognostic variables are all specified at the center of a grid cell. The C-grid is a staggered scheme in which only mass field variables are specified at the cell center and the normal components of velocity are specified on cell edges. (The tangential velocity on C-grid is not solved but reconstructed from the normal velocity component).

The companion paper Yu et al. (2020) shows the C-grid staggering scheme on icosahedral grids provided by TRSK (Thuburn et al. 2007 and Ringler et al. 2009) excels in numerical noise control and total energy conservation, yielding exceptional long-time integration stability. Its weakness lies in the lack of model error reduction with increasing resolution in specific test cases (especially the maximum error). A clear weakness of the A-grid method from Tomita et al. (2004) is grid-point noise which manifests much less on the C-grid. The A-grid method conserves potential enstrophy well and shows a linear reduction of error with respect to increasing resolution (Yu et al. 2020). For high-resolution weather codes error reduction with increasing resolution gains importance compared to long-time integration stability. Here the A and C grid schemes are implemented on icosahedral grids by us independently, and theoretical aspects and numerical results are reported by Yu et al. (2020).

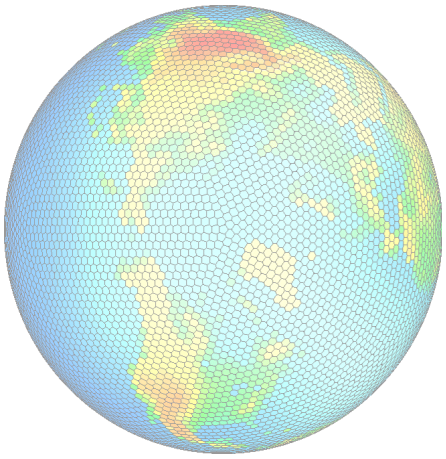

Figure 1. The icosahedral grid.

Figure 1 shows an illustration of the Icosahedral grid projected on the globe. The icosahedral grid is composed of hexagons and ten pentagons. Icosahedral grids are generated using the

widely accepted spring dynamics method for the A-grid (Tomita et al. 2001) and the Centroidal Voronoi Tessellations (CVT) method from Du et al. (1999) for the C-grid. Grid quality is strictly controlled and the published results from Tomita et al. (2001) and Ringler et al. (2010) are well reproduced by Yu et al. (2020) fulfilling the requirement for a side-by side comparison.

The method used to generate the grid (Wang and Lee 2011) divides the globe into ten equal rhombi which is an ideal starting place for an MPI decomposition of the globe. One rhombus is shown in Figure 2a and the ten rhombi are shown in Figure 2b. For less than ten MPI tasks the rhombi are combined and for more than ten MPI tasks the rhombi are subdivided further into smaller rhombi with almost equal sides providing a good perimeter to area ratio ensuring optimal inter-processor communications. The subdivided rhombi have nearly equal area ensuring the workload is evenly distributed between MPI tasks.

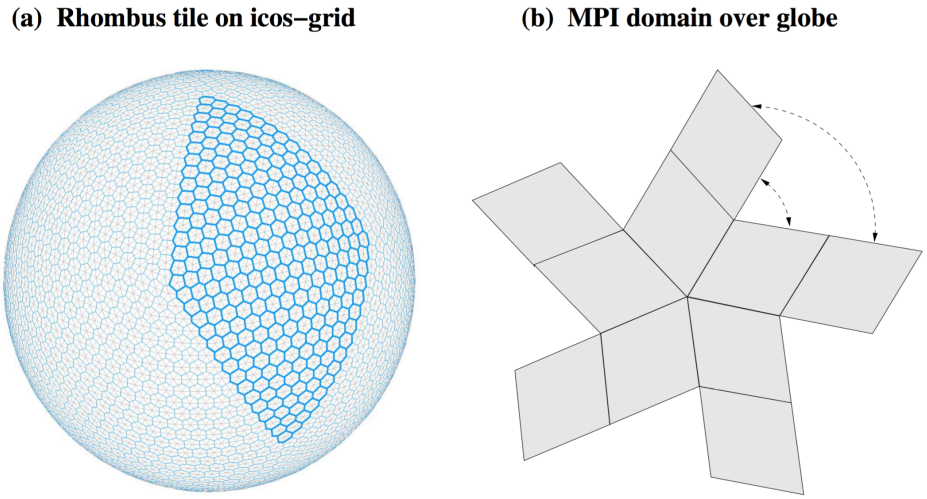

Figure 2. MPI domain decomposition on the icosahedral grid

3. SOFTWARE IMPLEMENTATION

3.1 A Single Software Framework for A-grid and C-grid schemes

To make the comparison between the two schemes as fair as possible, the Arakawa A and C grids are incorporated into a single software framework referred to as the Shallow-Water Model (SWM). The SWM framework consists of a main routine that calls an initialization routine followed by the main time-step loop. The time-step loop contains a classic Runge-Kutta RK4 integration which calls the shallow-water model dynamical core. Results are output after the time-step loop has completed. The initialization routine and the dynamical core can be switched at run-time between the A-grid and the C-grid.

The single framework provides the advantage of sharing:

- Compiler (Intel), hardware (Intel Skylake 2.4GHz) and utilities
- Time integration method (Runge-Kutta 4th order)
- Icosahedral grid (Wang and Lee 2011)
- MPI based parallelization
- Compiler optimization level
- Floating-point precision (single)
- Spatial discretion (icosahedral based FVM without numerical damping)
- Time step
- Timing mechanism, GPTL (Rosinski 2010)

3.2 Indirect Addressing

The icosahedral grid structure imposes constraints on how neighboring cells are accessed. Different from the conventional rectangular lat-lon grid, icosahedral grids have six neighbors except for the ten singularity points which have five neighbors. Despite this difference, since the globe is subdivided into rhombuses, the cells can be referenced as a collection of logically rectangular 2D arrays so the cell centers can be indexed using a traditional rectangular grid (Randal 2002). We used indirect addressing based on several factors. Indirect indexing requires a lookup table to identify neighboring cells, which requires an additional memory access. However, in full 3D weather models the cost of the extra memory reference can be greatly reduced if the vertical loop is the innermost loop index. Since the vertical dimension in high resolution models is 60-120 levels, the cost of the indirect address can be amortized over the levels which results in negligible impact (MacDonald et al., 2010). Further, indirect addressing provides more freedom in coding, allows more flexibility in the layout of the data decomposition, and offers other performance advantages that are discussed below.

The code was parallelized using the Scalable Modeling System (SMS) compiler that converts serial code and SMS directives into MPI parallel code. SMS also includes an MPI-based run-time library that supports domain decomposition, inter-process communications, reductions, and other capabilities needed to run on distributed memory systems (Govett et al., 2003).

3.3 Inter-process Communications

Inter-process communication is needed to exchange information between neighboring MPI tasks. MPI communication is typically accomplished in three stages: pack data, communicate, and unpack data. The packing and unpacking operations are used to copy data into or out of message buffers that are needed to communicate halo data between neighboring MPI tasks. These buffers are used to organize the data so it is contiguous in memory, a necessary step for efficient MPI communications. To illustrate, Figure 3a shows the memory layout for a rectangular MPI region using a standard 2D decomposition where continuous lines indicate data that is stored contiguously in memory. The red lines indicate perimeter regions from which data is sent to neighboring MPI tasks. The red lines show that data is stored contiguously in memory in the horizontal dimension, but the data is non-contiguous in the vertical dimension thereby requiring packing operations. Further, the required memory references in the vertical dimension are inefficient because cache is not properly utilized. Traditionally the halo region is a simple one cell extension of the data region so the halo would also be contiguous only in the horizontal direction thereby requiring inefficient unpacking in the vertical dimension.

In addition to the traditional pack-comm-unpack style communications, SMS supports no pack communications. This approach can be used because indirect addressing allows cells to be organized in any order. Figure 3b illustrates a spiral layout supported by SMS. The spiral layout is shown only for the outer three layers because the ordering of the interior points is arbitrary, based on the size and shape of the MPI sub-regions. Unlike the rectangular layout, the perimeter of the spiral layout has data which is contiguous for the entire perimeter. Sending data is accomplished in one MPI send for each side with no packing except for one copy of the first cell into the last cell of the left-hand perimeter. Data is sent directly from the perimeter of each domain. Indirect addressing also allows the halo to be placed anywhere so space is reserved at the end of the data array and the data is received directly into the halo with no unpacking.

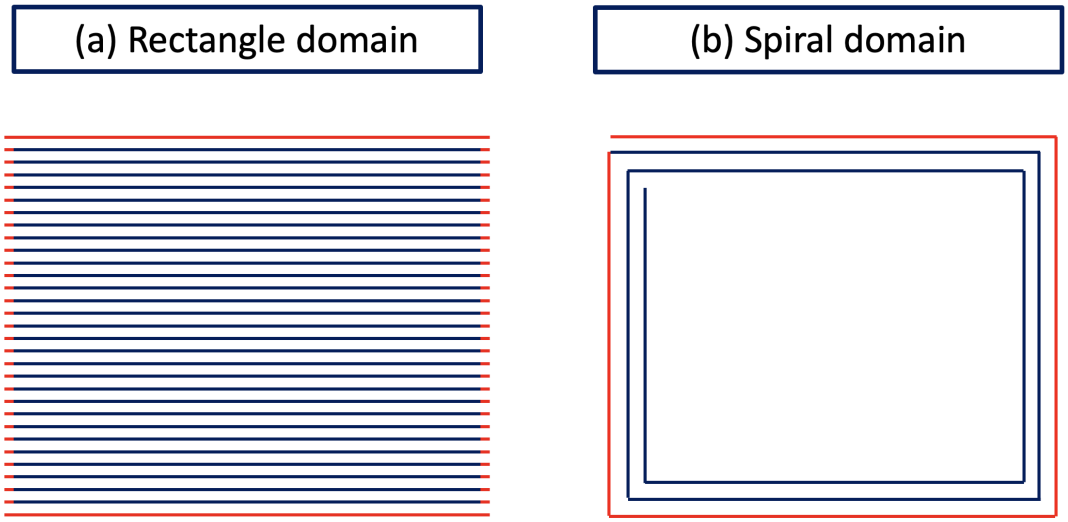

Figure 3. Rectangular and Spiral domain layout where the red lines indicate perimeter regions from which data is sent to neighboring processes.

No pack communications can speed up inter-process communications quite a bit since packing and unpacking data can take up to 40% of the communications time (Middlecoff 2015). On most supercomputers packing and unpacking cannot be overlapped with other calculations. For comparisons that show detailed performance benefits of no-pack communications, see Middlecoff 2015, and for comparisons that show the benefits on the CPU, MIC and GPU, see Govett et al. 2017.

3.4 Variable Storage on the Icosahedral Grid

For the A-grid it is natural to store prognostic variables at cell centers since all variables are defined at cell centers. For the C-grid the fluid depth h is defined at cell centers but the normal velocities V_n are defined on cell edges so typically the discretized variables are stored accordingly; h is stored at cell centers and the V_n are stored at cell edges. The size of the array for storing h is NIP (Number of Icosahedral Points), but the array size to store V_n is 3*NIP. Consequently, the parallelization of C-grid models such as MPAS typically use two decompositions to support the distributed data.

The SWM software framework stores all variables at cell centers. This is true for both the A-grid and the C-grid. Storing all variables at cell centers in natural for the A-grid since all variables are defined at cell centers. For the C-grid V_n are defined at cell edges so to store V_n at cell centers requires a restructuring of V_n. V_n becomes a two dimensional cell centered variable of the form V(6,NIP) where 6 is the number of hexagon sides and NIP is the Number of Icosahedral Points. V(IS,IP) then holds side IS of cell IP. The six sides of a given hexagon are shared by neighboring

hexagons, so for a given hexagon the V_n are calculated on three sides and then are copied to the V_n on the adjacent sides of the neighboring hexagons as shown in Figure 4.

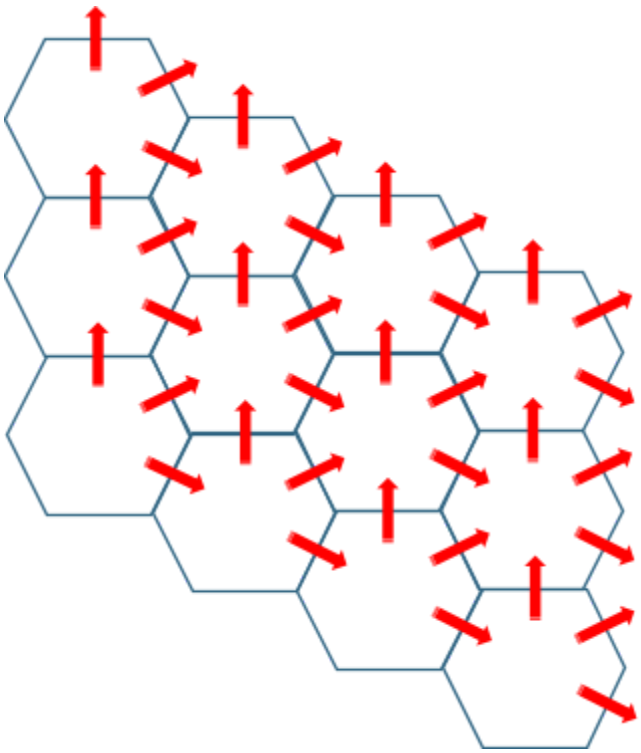

Figure 4. Hexagonal cells where the red arrows illustrate the calculation (arrow base) and copying (arrow direction) of V_n

C-grid advantages of storing V_n at cell centers:

- Avoids the cost and complications of two decomposed dimensions
- Better parallel performance

C-grid disadvantages of storing V_n at cell centers:

- Involves an extra copy of calculated sides to other sides
- Doubles the memory requirement for V_n
- Reduces cache coherence for V_n because the arrays are larger
- Reduces serial run time by 2%.

4. ACCURACY AND PERFORMANCE

4.1 Scientific Accuracy

Scientific accuracy of our SWM A-grid and C-grid implementation is evaluated in the companion paper (Yu et al. 2020). Error norm results for simulations of 50 days for steady-state geostrophic flow (Williamson et al. (1992) test case 2) for the A-grid and C-grid are reproduced here (Figure 5 of Yu et al. (2020)). The figure shows both error norms in the depth field, $L_\infty(h)$ and $L_2(h)$, decrease linearly with the number of grid points for the A-grid method. In contrast, for the C-grid method this is true only for the $L_2(h)$ norm (RMS error); the C-grid $L_\infty(h)$ norm (maximum error) barely decreases with grid resolution. This is a known imperfection of the C-grid method (Ringler et al. (2010), Peixoto (2016), Yu et al. (2020)). Although Figure 5 shows the $L_2(h)$ norm for the C-grid superior to that of the A-grid at lower resolutions, at 56km resolution the A-grid shows a slightly faster L2 norm convergence rate. Since we are interested in resolutions higher than 56km, we examine L_2 norm convergence rates at higher resolution.

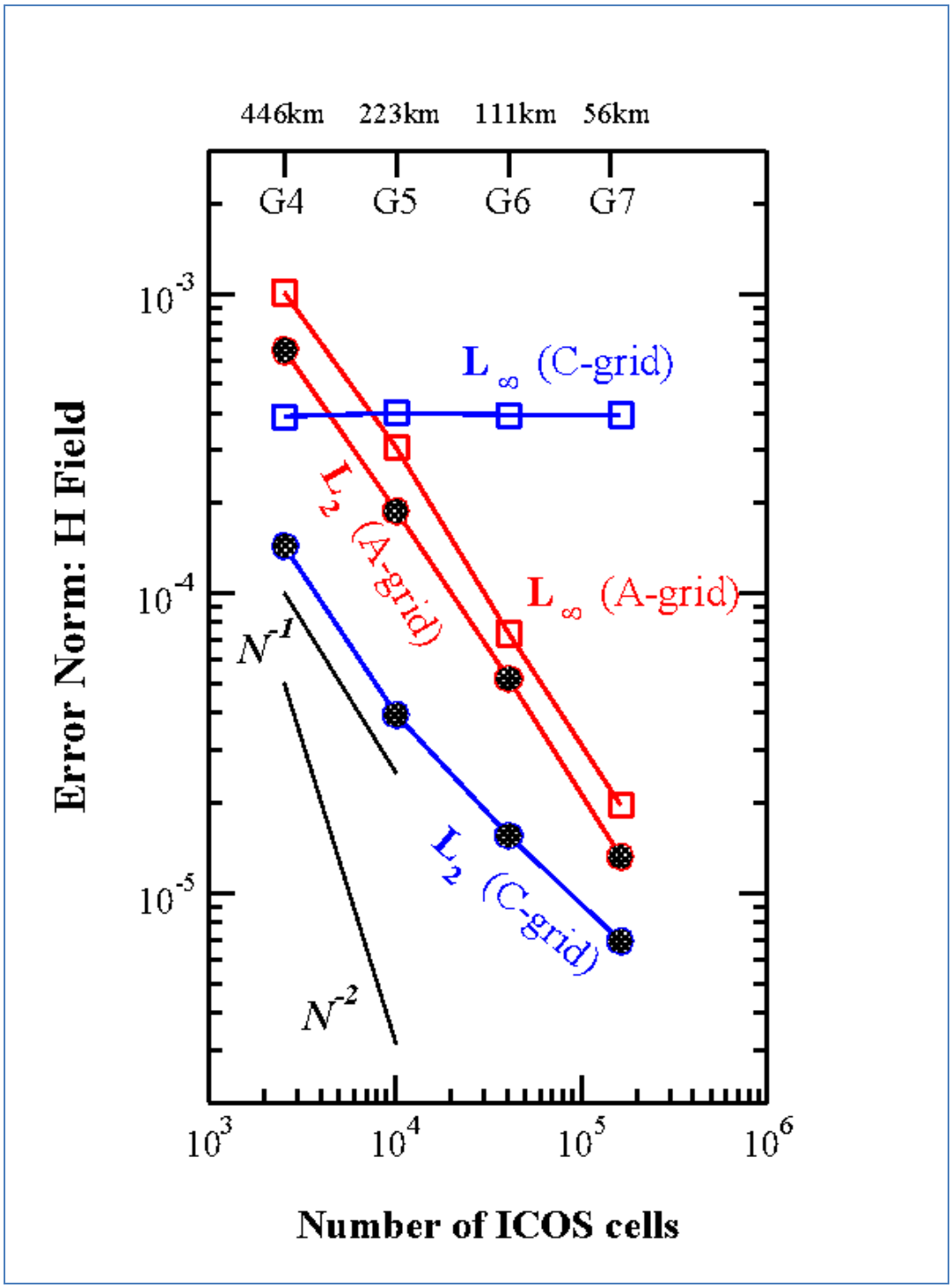

Figure 5. Convergence rates at 50 days for the L_2 and L_∞ error norms, Yu et al. (2020) (Yu's Test Case 2). (Reproduced from the Monthly Weather Review with permission from the American Meteorological Society © Copyright 23 September 2020 AMS.)

4.2 Performance and Scaling

To integrate performance and accuracy, the A-grid and C-grid $L_2(h)$ error norms from test case 2 are plotted as a function of forecast time (y vs. x in Figure 6), at prescribed grid resolution (labels in Figure 6). The forecast time is defined as the CPU time for 80 Intel Skylake cores to complete a 20-day forecast for fixed 223km to 23km resolutions. At higher resolutions Figure 6 shows the A-grid achieves higher accuracy than the C-grid and increasingly outperforms the C-grid as resolution increases to 28km.

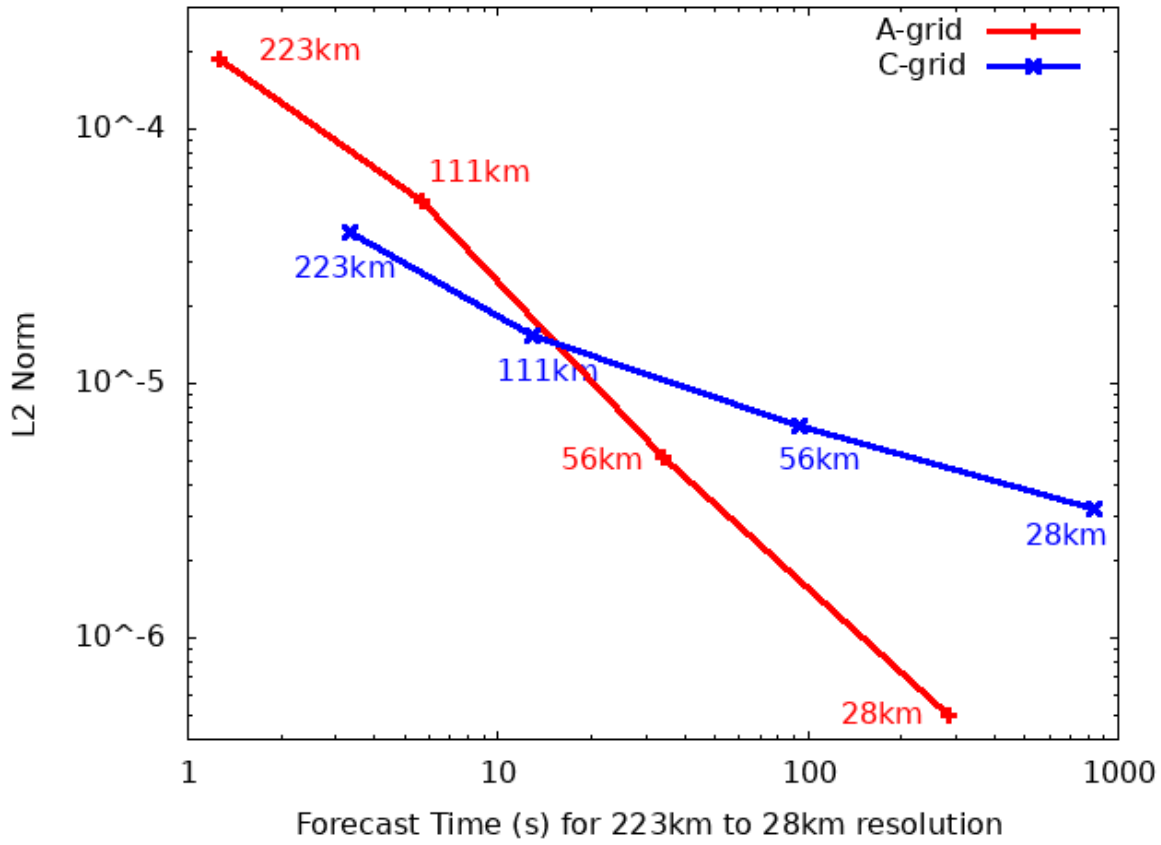

Figure 6. Convergence rate of the $L_2(h)$ error norm for A and C grids as a function of forecast time, accuracy and resolution. Forecast times are for a 20-day forecast using 80 Intel Skylake cores.

Figure 7 shows strong scaling for 14km 1-day runs for the SWM A-grid, the SWM C-grid, and the MPAS core_sw shallow-water model. MPAS core_sw and the MPAS icosahedral grid were downloaded from the MPAS web site (https://mpas-dev.github.io, v7.1). Icosahedral grids similar to the MPAS grid are generated from our SWM code using an algorithm described in Du et al. (1999). The black line shows the slope of a perfect parallel efficiency of 1.00. The linearity of the log-log plots indicates nearly ideal scaling for all models.

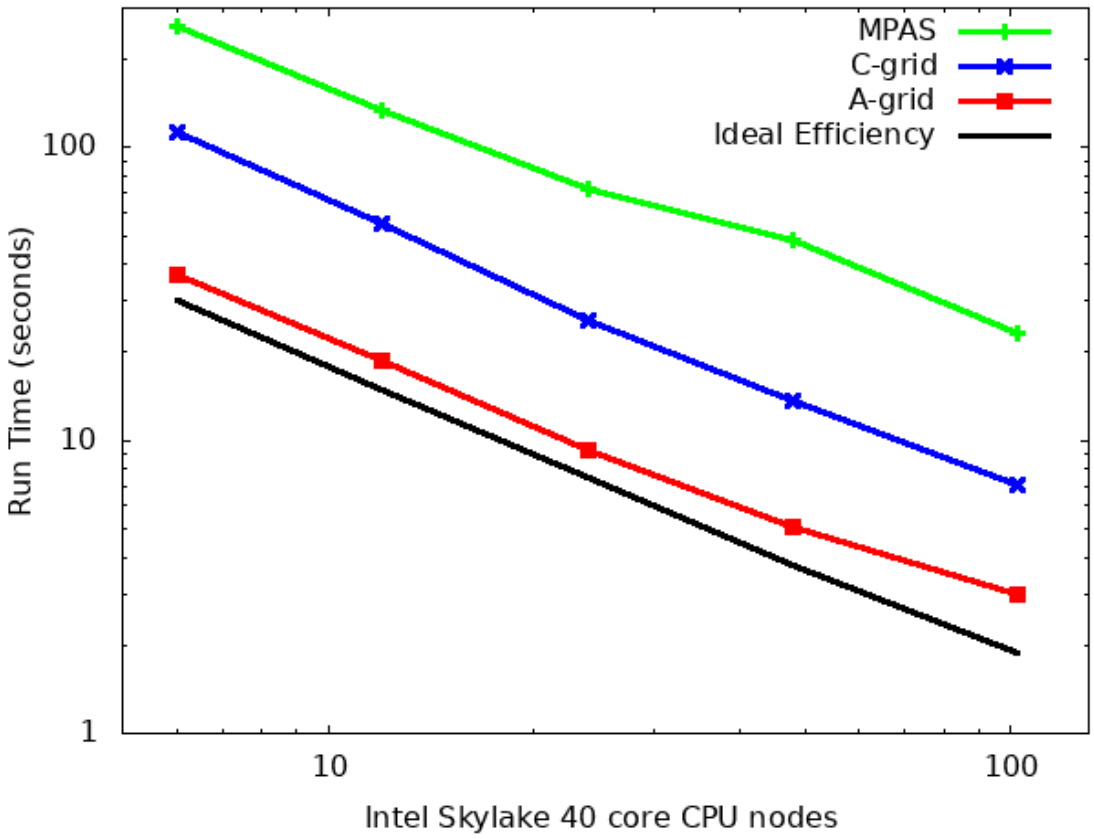

Figure 7. Scaling for 14km 1 day runs for the A-grid, C-grid and MPAS core_sw

Table 1 provides a detailed comparison of performance and efficiency. These results show that the SWM A-grid averages 2.8X faster than the SWM C-grid and the SWM C-grid averages 2.9X faster than the MPAS core_sw. Note that MPAS core_sw was developed during early stages of MPAS development and was not optimized[1].

[1] Communication from the MPAS team

NODES	A-grid Time (s)	A-grid Scaling Efficiency	C-grid Time (s)	C-grid Scaling Efficiency	MPAS Time (s)	MPAS Scaling Efficiency	MPAS/ C-grid	A-grid/ C-grid
6	36.97	1.00	113.3	1.00	261.20	1.00	2.30	3.07
12	18.69	0.99	54.80	1.03	134.15	0.97	2.45	2.93
24	9.22	1.00	25.58	1.11	72.63	0.90	2.84	2.77
48	5.05	0.91	13.60	1.04	48.16	0.68	3.54	2.69
102.4	2.96	0.69	7.03	1.01	23.12	0.71	3.29	2.38

Table 1. Scaling for 14km 1-day runs for the A-grid, C-grid and MPAS core_sw. Scaling efficiency is calculated based on the performance at 25 nodes instead of the traditional 1 node.

4.3 Performance Analysis

Anticipating next generation high-resolution non-hydrostatic weather prediction models, the A-grid and C-grid re compared at 3.5km resolution. Figure 8 shows strong scaling for a one day forecast at 3.5km resolution running on the Intel Skylake. Again, the black line shows the slope of a perfect parallel efficiency of 1.00. See Table 2 for actual efficiencies, run times, and the ratios of run times. The linearity of the log-log plots indicates ideal scaling. At 3.5km the A-grid averages 3X faster than the C-grid. Because the A-grid is faster, and as an example of what could be done, the A-grid performance at 2.3km resolution is included for a comparison with the C-grid at 3.5km resolution. Although the A-grid at 2.3km has 50% higher resolution than the C-grid at 3.5km, the A-grid takes only 15% more time to execute.

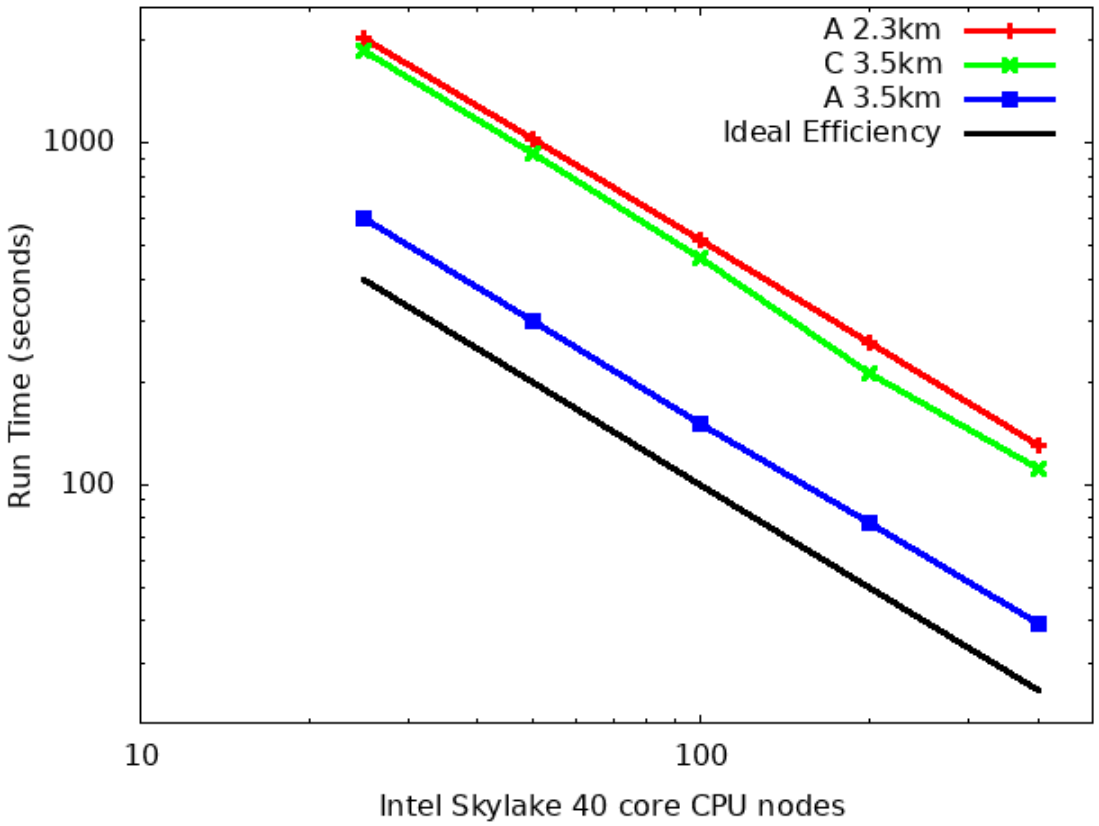

Figure 8. High-resolution scaling for the A and C grids.

NODES	A3.5km Time	A3.5km PE	C3.5km Time	C3.5km PE	A2.3km Time	A2.3km PE	C3.5km/ A3.5km	A2.3km/ C3.5km
25	604	1.00	1869	1.00	2043	1.00	3.10	1.09
50	300	1.01	929	1.01	1029	0.99	3.10	1.11
100	151	1.00	459	1.02	522	0.98	3.05	1.14
200	77.3	0.98	213	1.10	260	0.98	2.75	1.22
400	39.2	0.96	111	1.05	130	0.98	2.84	1.17
MEAN							2.97	1.15

Table 2. High-resolution strong scaling in seconds for the A and C grids. PE is parallel efficiency calculated based on the performance at 25 nodes instead of the traditional 1 node.

Another way to view strong scaling is with the metric giga zone-cycles/second (GZCS) defined by:

$$GZCS = NIP*Cycles/Time/10^9$$

Where:

NIP = the Number of Icosahedral Points over all processors
Cycles = the number of time steps to complete the run
Time = the time in seconds to complete the run

Figure 9 shows GZCS as a function of 40 core nodes. The black lines represent perfect scaling for the 2.3km A-grid and the 3.5km C-grid. The gap between the two A-grid runs and the C-grid run illustrates the large performance difference observed.

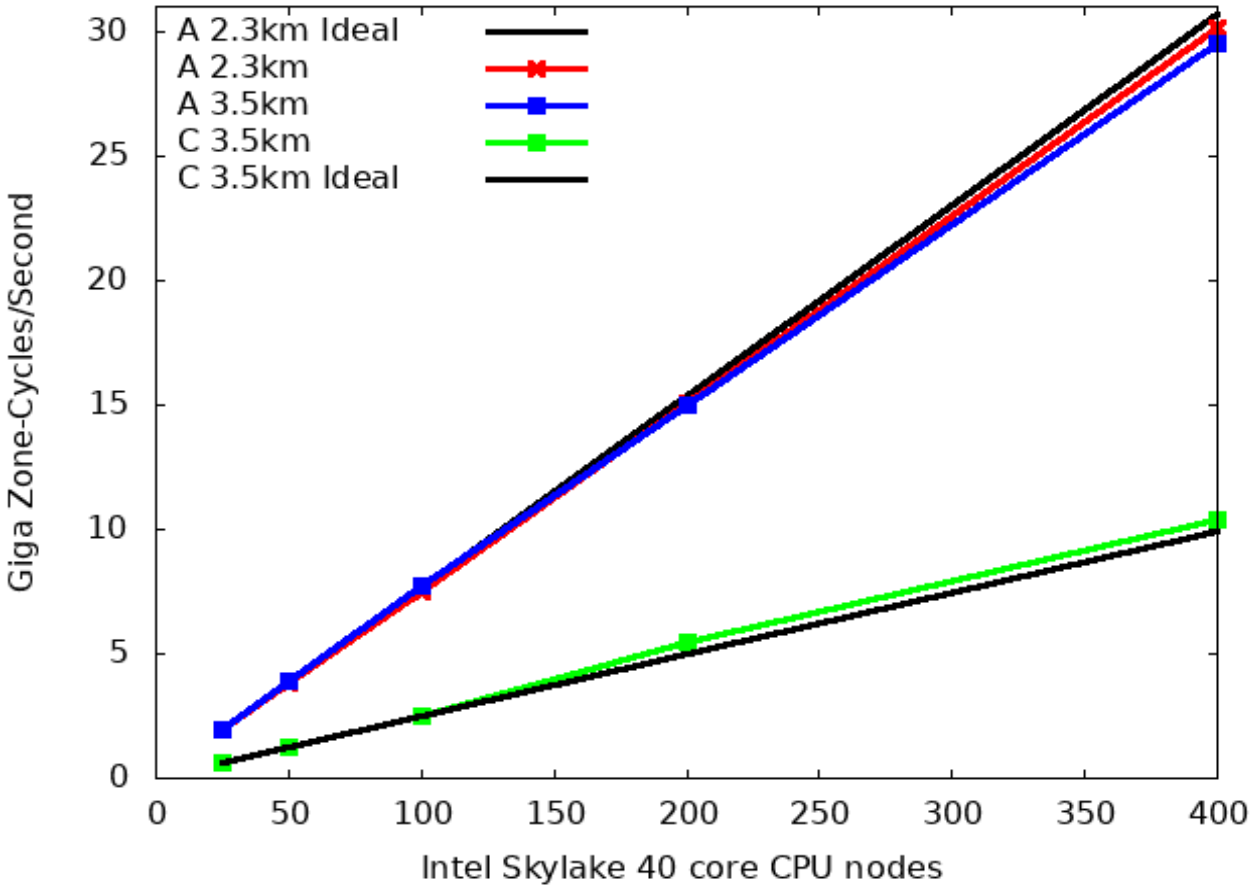

Figure 9. High-resolution strong scaling of the A and C grids presented as Zone cycles/second.

To understand why the A-grid is faster than the C-grid, consider Example 1, a typical A-grid loop from the implementation presented here:

```
do ip=1,NIP
  do is=1,nprox(ip)
    isp=ispA(is,ip)
    jp=prox(is,ip)
    kp=prox(isp,ip)
    Fc(1,is)=FV(1,ip)*wtsph(1,is,ip) + FV(1,jp)*wtsph(2,is,ip) &
                                     + FV(1,kp)*wtsph(3,is,ip)
    Fc(2,is)=FV(2,ip)*wtsph(1,is,ip) + FV(2,jp)*wtsph(2,is,ip) &
                                     + FV(2,kp)*wtsph(3,is,ip)
    Fc(3,is)=FV(3,ip)*wtsph(1,is,ip) + FV(3,jp)*wtsph(2,is,ip) &
                                     + FV(3,kp)*wtsph(3,is,ip)
    Fc(4,is)=FV(4,ip)*wtsph(1,is,ip) + FV(4,jp)*wtsph(2,is,ip) &
                                     + FV(4,kp)*wtsph(3,is,ip)
    Ec(is)= &
    (.5*(FV(2,ip)**2+FV(3,ip)**2+FV(4,ip)**2) + g*(FV(1,ip)+hb_a(ip)))* &
                                                wtsph(1,is,ip)+ &
    (.5*(FV(2,jp)**2+FV(3,jp)**2+FV(4,jp)**2) + g*(FV(1,jp)+hb_a(jp)))* &
                                                wtsph(2,is,ip + &
    (.5*(FV(2,kp)**2+FV(3,kp)**2+FV(4,kp)**2) + g*(FV(1,kp)+hb_a(kp)))* &
                                                wtsph(3,is,ip)
  enddo
enddo
```

Example 1. Typical A-grid loop nest. NIP is the Number of Icosahedral Points and nprox(ip) is the number of polygon sides (5 or 6).

The A-grid implementation uses a four-vector (h, Vx, Vy, Vz) for prognostic variables. Since most CPUs have an L1 cache line of four single precision words, and since all tests are single precision, the four prognostic variables are organized into arrays of the form FV(4,NIP). The same was done for intermediate variables such as Fc.

The form FV(4,NIP) improves L1 cache hits and overall cache performance but a more important factor in the speed of the A-grid is the expression reuse or, more precisely, variable reuse given by the recurrence of the variables FV and wtsph in the above loop nest. Variable reuse is further investigated below by separating the performance factors.

More effort was devoted to optimizing the C-grid than the A-grid, but no way was found to improve variable reuse for the C-grid. In the ICOS-center-decomposition method we used for the C-grid, four arrays are designated for prognostic variables (Yu et al. 2020): h (fluid depth) at cell centers and the three V_n (normal velocity) on cell edges. However, since h is calculated at cell centers and V_n is calculated at cell edges, the calculations are separate sets of equations which necessitates the calculation of h and V_n in separate loops. Consequently, there is no reuse between h and V_n and there is no cache advantage to organizing h and V_n into a four-vector. And the V_n equations themselves present little opportunity for reuse.

For serial runs at 223km resolution (10,242 grid points), the Performance Application Programming Interface (PAPI) was used to analyze some of the factors that contribute to speedup. Table 3 shows the PAPI results without optimization.

Operation	A-Grid	C-grid	C/A
Run Time (seconds)	26.960	26.911	1.00
L1 Data Cache Misses	1.64×10^8	5.77×10^8	3.52
L2 Data Cache Misses	1.43×10^8	5.33×10^8	3.73
L1 Inst Cache Misses	2.82×10^7	1.60×10^7	0.57
L2 Inst Cache Misses	4.95×10^6	1.22×10^7	2.46
Floating-point Ops	1.22×10^{10}	9.98×10^9	0.82
Loads and Stores	1.02×10^{11}	1.06×10^{11}	1.04
Computational Intensity	0.120	0.0942	0.79

Table 3: PAPI results from non-optimized code.

Without optimization, the A-grid has 1.22X more single precision floating-point operations than does the C-grid but runs essentially as fast as the C-grid. The main difference is the data cache misses where the A-grid shows over a 3X advantage, due in part to the four-vector organization of FV(4,NIP). The instruction cache misses are over an order of magnitude less than the data cache misses and do not all favor the A-Grid. The A-grid cache advantage more than overcomes the 1.22X more floating-point operations.

Operation	A-Grid	C-grid	C/A
Run Time (seconds)	1.901	4.304	2.26
L1 Data Cache Misses	1.59×10^8	5.52×10^8	3.47
L2 Data Cache Misses	1.42×10^8	5.26×10^8	3.70
L1 Inst Cache Misses	4.75×10^6	8.01×10^6	1.69
L2 Inst Cache Misses	3.60×10^6	5.96×10^6	1.66

Floating-point Ops	1.10×10^{10}	9.96×10^{9}	0.90
Loads and Stores	9.04×10^{9}	2.12×10^{10}	2.35
Computational Intensity	1.22	0.465	0.38
GigaFLOPS	6.42	2.32	0.36

Table 4: PAPI results after optimization -O2 -ftz -fp-model precise -xHOST -axcore-avx512

After optimization (Table 4), the A-grid maintains an over 3X advantage in data cache misses while instruction cache misses are insignificant. After optimization the A-grid has half as many loads and stores as the C-grid which supports the idea that the difference in performance is due to higher variable reuse in the A-grid code. The better cache coherency of the A-grid, even without optimization, also supports the importance of variable reuse in the A-grid performance. Since the floating-point operations are similar, the A-grid 2.26X computational advantage is due to fewer loads and stores.

To further understand the results, vectorization and FMA were also measured. While the A-grid loops vectorize, vectorization only improves performance by 18%. The limited improvement is due to the short inner loops with six iterations and the use of indirect addressing. Examining the C-grid, the loops do not vectorize because the inner loop contains an IF statement. However, the IF statement reduces the amount of work by 50% which more than compensates for the lack of vectorization. In summary, vectorization provides an 18% advantage and FMA provides an 8% advantage for the A-grid over the C-grid. It should be noted for weather models containing a vertical dimension, a contiguous and vectorizable innermost vertical loop amortizes the cost of both the IF test and indirect addressing. To understand the impact of a vertical dimension on the A-grid and C-grid variants, an innermost vertical dimension was artificially added as the innermost loop in the SWM code. Running at 3.5km resolution on 200 nodes of Intel Skylake with 192 vertical levels, the A-grid is 2.9X faster than the C-grid which is about the same advantage as shown by the shallow-water version (3X).

Table 5 shows results for optimized 3.5km 100 node runs with MPI barriers in place to isolate exchange times and compute times.

Operation	A-Grid	C-grid	C/A
Total runtime (seconds)	223	631	2.83
Compute-time (seconds)	153	448	2.93
Communications-time (seconds)	9.26	30.7	3.32

Communications-time/Compute-time	0.061	0.069	1.13

Table 5: Optimized 3.5km runs with MPI barriers.

Table 5 shows that communications account for less than seven percent of the compute runtime. The 3X difference for communications is likely due to 4 times more data that must be communicated for the C-grid. Communication and computation times for the C-grid are 3.32 and 2.93 times longer respectively. In addition, the ratio of communications to compute time is similar (1.13) for both grids. As a result, the longer C-grid communications time (30.7 seconds) does not have a large effect on the ratio of C-grid runtime to A-grid runtime (1.13). Further, the fact that the compute time and exchange time for the C-grid is about three times that of the A-grid explains why the C-grid scales better than the A-grid: it is because the increased time puts the C-grid at an earlier point on the efficiency curve where scaling is better. The superior efficiency of the C-grid at 14km resolution shown in Table 1 also supports the proposition that the better scaling of the C-grid is due to its higher level of computational work.

5. CONCLUSION

Performance of the Arakawa A-grid (NICAM) and C-grid (MPAS) schemes solving the shallow-water equations applied to the icosahedral grid on a sphere was compared in one framework. Performance of the MPAS shallow-water model was included to verify the C-grid implementation is robust. The $L_2(h)$ error norms of both grids were presented as a function of time to solution for fixed resolutions which shows the A-grid's superiority in $L_2(h)$ norm at high resolution. Strong scaling of the A and C grids running at 3.5km resolution was presented which shows the A-grid averages three times as fast as the C-grid which is attributed mostly to the A-grid equations which promote much better variable reuse. The A-grid at 2.3KM resolution was compared to the C-grid at 3.5km resolution showing the A-grid at 50% higher resolution is only 15% slower.

As the weather community moves to the next generation, non-hydrostatic and cloud resolving systems, the A-grid superior $L_2(h)$ and $L_\infty(h)$ error norm high resolution performance and superior speed makes the A-grid a good candidate for future high resolution global weather prediction models.

ACKNOWLEDGEMENTS

We thank Dr. Duane Rosenberg for suggestions and discussions which improved this paper. We thank Dr. Ning Wang for important discussions, clarifications, and suggestions relating to grid and decomposition techniques.

REFERENCES

Arakawa, A., and V. R. Lamb, 1977: Computational design of the basic dynamical processes of the UCLA general circulation model. Methods in Computational Physics, J. Chang, Ed., Vol. 17, Academic Press, 173–265.

Du, Q., V. Faber, and M. Gunzburger, 1999: Centroidal Voronoi Tessellations: Applications and algorithms. SIAM Rev., 41, 637–676, https://doi.org/10.1137/S0036144599352836.

Govett, M., Hart, L., Henderson, T., Middlecoff, J., and Schaffer,D. 2003: The Scalable Modeling System: directive-based code parallelization for distributed and shared memory computers. Parallel Computing. Volume 29, Issue 8, Pages 995-1020.

Govett, M., Rosinski, J., Middlecoff, J., Henderson, T., Lee, J., MacDonald, A., Wang, N., Madden, P., Schramm, J., and Duarte, A., 2017; Parallelization and Performance of the NIM Weather Model on CPU, GPU, and MIC Processors. Bulletin of the American Meteorological Society. Volume 98 Issue 10, Pages 2201-2213. https://doi.org/10.1175/BAMS-D-15-00278.1

Lee, J.-L., and A. E. MacDonald, 2009: A finite-volume icosahedral shallow-water model on a local coordinate. Mon. Wea. Rev., 137, 1422–1437, https://doi.org/10.1175/2008MWR2639.1

MacDonald, A.E., Middlecoff, J.F., Henderson, T.B., and Jin-Luen Lee, 2010: A General Method for Modeling on Irregular Grids. The International Journal of High Performance Computing Applications. Volume 25 Issue 4 Pages 392-403. https://doi.org/10.1177/1094342010385019

Middlecoff, J., 2015 NCAR 2015 Multicore Workshop, Session Four: Weather Models. Optimization of MPI Message Passing in a Multi-Core NWP Dynamical Core Running on Nvidia GPUs

Peixoto, P. S., 2016: Accuracy analysis of mimetic finite volume operators on geodesic grids and a consistent alternative. *J. Comput. Phys.*, **310**, 127–160.

Randall, D.A. & Ringler, Todd & Heikes, Ross & Jones, Philip & Baumgardner, John. (2002). Climate Modeling with Spherical Geodesic Grids. Computing in Science & Engineering. 4. 32 - 41. 10.1109/MCISE.2002.1032427.

Ringler, T. D., J. Thuburn, J. B. Klemp, and W. C. Skamarock, 2010: A unified approach to energy conservation and potential vorticity dynamics for arbitrarily-structured C-grids. *J. Comput. Phys.*, **229**, 3065–3090.

Rosinski, James (2010). CUG 2010 Proceedings. General Purpose Timing Library (GPTL): A Tool for Characterizing Performance of Parallel and Serial Applications, James Rosinski, Oak Ridge National Laboratory (ORNL)

https://cug.org/5-publications/proceedings_attendee_lists/CUG10CD/pages/1-program/final_program/CUG10_Proceedings/pages/authors/01-5Monday/4B-Rosinski-paper.html

Satoh, M., T. Matsuno,H. Tomita, H. Miura, T. Nasuno, and S. Iga, 2008: Nonhydrostatic ICosahedral Atmospheric Model (NICAM) for global cloud resolving simulations. J. Comput. Phys., 227, 3486–3514, https://doi.org/10.1016/j.jcp.2007.02.006

Skamarock, W. C., J. B. Klemp, M. G. Duda, L. D. Fowler, S.-H. Park, and T. D. Ringler, 2012: A multiscale nonhydrostatic atmospheric model using centroidal Voronoi tesselations and C-grid staggering. Mon. Wea. Rev., 140, 3090–3105, https://doi.org/10.1175/MWR-D-11-00215.1.

Thuburn, J., T. D. Ringler, W. C. Skamarock, and J. B. Klemp, 2009: Numerical representation of geostrophic modes on arbitrarily structured C-grids. *J. Comput. Phys.*, **228**, 8321–8335.

Tomita, H., M. Tsugawa, M. Satoh, and K. Goto, 2001: Shallow water model on a modified icosahedral geodesic grid by using spring dynamics. J. Comput. Phys., 174, 579–613, https://doi.org/10.1006/jcph.2001.6897.

Wan, H., and Coauthors, 2013: The ICON-1.2 hydrostatic atmospheric dynamical core on triangular grids—Part I: Formulation and performance of the baseline version. Geosci. Model Dev., 6, 59–119, https://doi.org/10.5194/gmdd-6-59-2013.

Wang,N., and J. Lee, 2011:Geometric properties of the icosahedralhexagonal grid on the two-sphere. SIAM J. Sci. Comput., 33, 2536–2559, https://doi.org/10.1137/090761355.

Williamson, D. L., J. B. Drake, J. J. Hack, R. Jakob, and P. N. Swarztrauber, 1992: A standard test set for numerical approximations to the shallow water equations in spherical geometry. *J. Comput. Phys.*, **102**, 211–224.

Winninghoff, F. J., 1968: On the adjustment towards a geostrophic balance in a simple primitive equation model with application to the problems and objective analysis. Ph.D. thesis, University of California, 161 pp.

Yu, Y. G., Wang, N., Middlecoff, J., Peixoto, P., Govett, M., 2020: Comparing Numerical Accuracy of Icosahedral A-Grid and C-Grid Schemes in Solving the Shallow-Water Model.

Monthly Weather Review, Volume 148 Issue 10,
4009-4033, https://doi.org/10.1175/MWR-D-20-0024.1

Zangl, G., D. Reinert, P. Ripodas, and M. Baldauf, 2015: The
ICON (ICOsahedral Non-hydrostatic) modelling framework
of DWD and MPI-M: Description of the non-hydrostatic dynamical
core. Quart. J. Roy. Meteor. Soc., 141, 563–579, https://doi.org/10.1002/qj.2378.