Hardware/Software Co-design of Global Cloud System Resolving Models

Michael F. Wehner [1], *Leonid Oliker* [1], *John Shalf* [1], *David Donofrio* [1], *Leroy A. Drummond* [1],*Ross Heikes* [3], *Shoaib Kamil* [1 2], *Celal Konor* [3], *Norman Miller* [1], *Hiroaki Miura* [4], *Marghoob Mohiyuddin* [1 2], *David Randall* [3], *Woo-Sun Yang* [1]

[1] CRD/NERSC, Lawrence Berkeley National Laboratory, Berkeley CA, 94720, USA
[2] EECS Department, University of California, Berkeley, CA, 94720, USA
[3] Department of Atmospheric Science, Colorado State University, Fort Collins, CO 80523, USA
[4] Center for Climate System Research, University of Tokyo, Kashiwa Chiba 277-8568, Japan

Manuscript submitted 07 04 2011

We present an analysis of the performance aspects of an atmospheric general circulation model at the ultra-high resolution required to resolve individual cloud systems and describe alternative technological paths to realize the integration of such a model in the relatively near future. Due to a superlinear scaling of the computational burden dictated by the Courant stability criterion, the solution of the equations of motion dominate the calculation at these ultra-high resolutions. From this extrapolation, it is estimated that a credible kilometer scale atmospheric model would require a sustained computational rate of at least 28 Petaflop/s to provide scientifically useful climate simulations. Our design study portends an alternate strategy for practical power-efficient implementations of next-generation ultra-scale systems. We demonstrate that hardware/software co-design of low-power embedded processor technology could be exploited to design a custom machine tailored to ultra-high resolution climate model specifications at relatively affordable cost and power considerations. A strawman machine design is presented consisting of in excess of 20 million processing elements that effectively exploits forthcoming many-core chips. The system pushes the limits of domain decomposition to increase explicit parallelism, and suggests that functional partitioning of sub-components of the climate code (much like the coarse-grained partitioning of computation between the atmospheric, ocean, land, and ice components of current coupled models) may be necessary for future performance scaling.

To whom correspondence should be addressed.
Michael Wehner, 1 Cyclotron Road, MS: 50F1650, Berkeley, CA 94720
e-mail: MFWehner@lbl.gov

1. Introduction

As early as the beginning of the 20th century, modeling of the Earth's atmosphere was proposed to be possible by an appropriate integration of the fluid equations of motion (Bjerknes 1904). As the descriptions of atmospheric processes and computer resources advanced, operational numerical weather simulation eventually followed, as did the development of global atmospheric general circulation models. Advances in the treatment of the physical processes in the atmosphere continued through the latter part of the 20th century, accompanied with rapid computational and observational advances allowing long-term simulation of the climate system. However, systematic errors in climate modeling persist to this day with one of the most fundamental sources for such errors being the description and simulation of clouds and their interaction with sources of solar and infrared radiation (Solomon et al. 2007). Current global climate models used to project the effects of anthropogenic changes to greenhouse gases and other pollutants cannot resolve individual clouds or cloud systems due to computational constraints on the horizontal resolution. Hence, complex subgrid scale processes, such as cumulonimbus convection, are parameterized rather than directly simulated. The advent of global cloud system resolving models (GCSRMs) offers the possibility to break the current deadlock in the predictability of cloud and precipitating systems by replacing mesoscale cumulus convection parameterizations with direct numerical simulation of cloud systems (Randall et al. 2003). GCSRMs resolve cloud systems rather than individual clouds and must then contain parameterizations of cloud microphysical processes. These microphysical cloud processes are closer to first principles than are mesoscale cloud behaviors and offer the possibility of far more realistic simulation of the entire atmosphere and climate system. However, the computational requirements of a global resolution model of near 1 km resolution remain a limiting factor for current generation supercomputers. A recent National Research Council study (NRC 2001) states that current computer architectures are insufficient for climate modeling needs, and a 2009 meeting of the World Meteorological Organization (Shukla et al. 2009) has set the goal of a 1 km resolution climate model as a 10-year goal.

The global climate models used for the 2007 IPCC (Intergovernmental Panel on Climate Change) 4th assessment report (AR4) ranged in horizontal resolution from about 400km to about 100km at the equator (Solomon et al. 2007). Several modeling groups are currently preparing models as fine as 50km for the 5th assessment report. These production models must be able to simulate decades to centuries of the climate in a reasonable amount of time to provide input to these reports. This constraint provides a meaningful metric for the minimum computational throughput of a production climate model. Although somewhat arbitrary, if a model can simulate the climate a thousand times faster than real time, the production demands of century scale climate simulation can be met in about three weeks of dedicated machine time. A millennial scale control run simulation would require a full year. For a point of reference, the standard version of the Community Climate System Model (CCSM), a state of the art climate model from the National Center for Atmospheric Research in Boulder, Colorado was integrated at a rate of about 1650 times faster than real time for the IPCC AR4. This is expected to increase to about 6000 times faster than real time for the IPCC fifth assessment report on current systems. GCSRMs will require at least a thousand times more computational cycles than current generation models like CCSM. One of the principal results reported in this paper is a credible estimate of the computational rates and other hardware attributes necessary to integrate GCSRMs in a manner similar to the production climate models of today. We arrive at these results through an analysis of several codes prepared at the Colorado State University in Fort Collins, Colorado. The estimates of overall machine requirements then provide the basis for our alternative path towards exascale scientific computing.

In previous papers, we proposed a radically new approach to high-performance computing (HPC) design via application-driven hardware and software co-design to leverage design principles from the consumer electronics marketplace (Wehner et al. 2008; 2009, Donofrio et al. 2009). Consumer electronics such as cell phones, digital cameras and portable music players have become ubiquitous in our modern lives. Because of the need to prolong battery life, designers of these portable devices must grapple with energy efficiency issues to an extent not necessary for traditional desktop computing applications. Hence, we look to this industry for solutions to the impending "power wall" that mainstream HPC approaches must soon face. The hardware/software co-design methodology we propose has the potential of significantly improving energy efficiency, reducing procurement cost, and accelerating the development cycle of exascale systems for targeted high-impact applications. Our vision of hardware/software co-design involves extensive collaboration between specialists in climate model development, software engineering and hardware design, especially chip architectures and network layouts. We show in this paper that this alternative path to exascale computing is ideally suited to the production usage of GCSRMs as climate models and estimate that such facilities could be up and running in as little as five years, as has been previously demonstrated by similar systems applied to other scientific disciplines such as the Anton molecular dynamics supercomputer (Shaw et al. 2009). Optimal hardware and software characteristics for the efficient execution of a specific climate model can be found iteratively. Much of this part of the co-design process can be automated, as we demonstrate in Section 6. However, iteration in the design of the climate model algorithm is not so readily automated. While there are indeed many possibilities to explore in this regard, we focus this study on a snapshot of a particular set of global atmospheric models. Actual full implementation of such a co-design strategy as we present here would entail expanding the model set to

include other types of atmospheric models along with similar models of the other important components of the climate system, such as the ocean, land, cryosphere and biosphere. The optimal set of hardware and software characteristics in such a co-designed system would be a compromise dictated by the differences between the important model components. Nonetheless, these characteristics would be far more specialized than that of a general purpose architecture and would lead to much more power and cost efficient simulations of climate models at these ultra-high resolutions.

2. Classes of Global Cloud System Resolving Models

The equations governing the atmosphere can be divided into roughly two categories. The first contains the equations of fluid motion, known as the Navier-Stokes equations. The second contains nearly everything else, including the description of clouds, radiation, turbulence and other resolved and/or unresolved processes. These equations in the second category act as source terms to the Navier-Stokes equations. They are generally operator split from the solution of the Navier-Stokes equations and can be governed by very different time scales. The Navier-Stokes equations, often referred to simply as the "dynamics", are nonlinear partial differential equations whose explicit stability is governed by a restriction known as the Courant condition, a relationship between grid spacing, time step and wind speed. At the ultra-high horizontal resolutions necessary to simulate cloud systems, the Courant stability criterion causes the dynamics portion of an atmospheric model to dominate the total computational burden because of severe time step restrictions. Typically, on a kilometer-scale grid, the Courant condition on the dynamics time step in an explicit scheme is determined by the high winds of the jet stream to be about 3 seconds. Implicit schemes allow this time step to be relaxed for a stable solution, but accuracy concerns limit this relaxation to about an order of magnitude or less. As mentioned above, other physical processes may have their own controlling time scales. For instance, the diurnal cycle, which is fixed at twenty-four hours, directly controls the incoming solar radiation and indirectly controls the outgoing infrared radiation. A relaxation of the radiation time step to several minutes is often used (R. Pincus 2010, personal communication). However, it may be desirable to associate the time step controlling other processes, such as those involving cloud physics or turbulence, with the dynamics time step.

The Earth, to a high degree of accuracy, can be approximated as a sphere. Numerical integration of the Navier-Stokes equations on a sphere has been performed in many different ways over the years. In the 1960s, the usage of spherical harmonic functions permitted extremely elegant and accurate discretizations to the dynamics equations. This spectral transform method arguably became the most widespread used technique in global atmospheric models for many decades and still finds common usage today. An $N \log N$ dependence on series length in the Fast Fourier Transforms (FFT) and an N^2 dependence on series length in the Legendre Transforms cause the arithmetic count per time step to increase superlinearly as resolution increases. Furthermore, in the absence of parallel FFT and Legendre transforms in the regime of interest, one dimensional domain decomposition strategies limit parallelism in these codes. However, despite predictions to the contrary, the methodology has been demonstrated to be competitive with grid based methods at resolutions up to T1279, approximately 16km at the equator (ECMWF 2010). It remains to be seen if spectral transform solutions can be extended to kilometer scale discretizations.

Grid based methods originally were based on latitude-longitude meshes. This class of grid has the advantage of being logically rectangular with easily calculated cell areas. It has the disadvantage of polar singularities at the top and bottom of the sphere. The effect of the singularities is to cause the cells to become long and narrow near the pole. A variety of specially constructed filtering operators have been constructed to damp computational instabilities and allow explicit time steps determined by the grid spacing at lower latitudes overcoming the grid deficiency. However, at a resolution of 1 km at the equator, the most poleward cells have an aspect ratio in excess of 10,000. Accurate solution of the Navier-Stokes equations in such high aspect ratio grids is notoriously difficult. It is unlikely that any latitude-longitude discretization of the Navier-Stokes equations at cloud system resolving resolutions would be accurate enough or computationally affordable.

Fortunately, research into other grid discretizations is well developed. The "cubed sphere" mesh (Sadourny 1972, McGregor 1996) is one of these, and begins with a cube, discretized on each face into square cells. Then the cube is topologically transformed into a sphere. As an analogy, consider a child's inflatable toy shaped as box that is overinflated into a ball. The original cells undergo a regular transformation and remain orthogonal to each other on the surface of the sphere. The eight corners of the original cube become special points where connectivity of the mesh is different. At these points, three cells border on each other. Everywhere else four cells border on each other. At very high resolution, the cells near the corners of an orthogonal cubed sphere mesh can vary greatly in area compared to cells near the interior of the original cube faces. This difference in area can be rectified by relaxing the orthogonality of the mesh resulting in a favorable range of cell areas across the entire sphere (Rancic et al. 1996). The Rancic equal angle grid is used in nearly all modern cubed sphere techniques. Recent work has shown that solutions on highly nonorthogonal gnomic grids are of an acceptable accuracy and far more computationally efficient that conformal orthogonal grids (Putnam and Lin 2007). Hence, modern finite element and finite volume methods do not require special treatment at these special points.

The models that are analyzed in the present paper are based

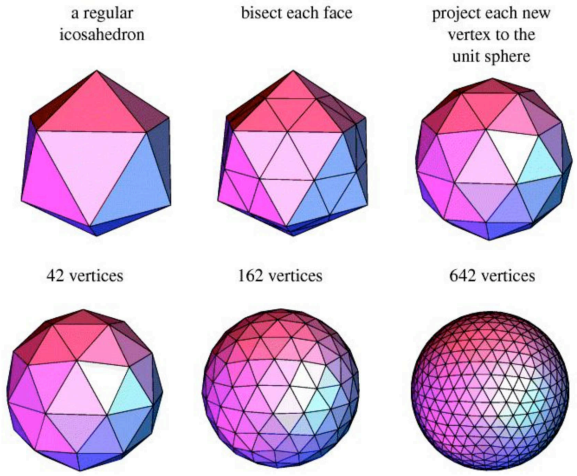


Figure 1: The geodesic mesh used by the Colorado State University group to represent the Earth's atmosphere is generated from an icosahedron (upper left panel). In this scheme, the triangular faces of the icosahedron are first bisected (upper middle panel). Then the new vertices are projected onto a sphere (upper right panel), as if it were a ball being inflated. This procedure is repeated (lower panels) until the desired resolution is obtained. This study's target resolution after 12 bisections is 167, 772, 162 vertices.

on a geodesic grid. In this class of mesh, the starting geometry is an icosahedron. A grid is generated by successively bisecting the triangular faces of the icosahedron as shown in Figure 1. Similar to the cubed sphere grid, a projection onto the sphere completes the grid generation. Because the distortion from icosahedron to sphere is smaller than from cube to sphere, the grid cells need no further modification to maintain an acceptable range of cell areas, even at ultra-high resolution (Heikes and Randall 1995). However, icosahedral meshes contain 12 points of special connectivity that require a special treatment and care must be taken to avoid imprinting these points on the numerical solution. Table 1 summarizes the number of cells and their approximate spacing between grid points for geodesic meshes as the number of bisections increases. The mesh of interest in this study is Level 12, with a ~2km cell size.

In this study, we are not advocating a particular methodology to the solution of partial differential equations on the sphere. The solution methods analyzed here are only a convenient first step in developing a comprehensive hardware/software co-design procedure. A convincing co-design strategy must contain analyses of all credible solution techniques and remain flexible enough to accommodate changes in those techniques.

Levels (# of Bisections)	# of Vertices	Average Cell Size (km)
1	42	4003.2
2	162	2001.6
3	642	1000.8
4	2562	500.4
5	10242	250.2
6	40962	125.1
7	163842	62.6
8	655362	31.3
9	2621442	15.7
10	10485762	7.8
11	41943042	3.9
12	167772162	2.0
13	671088642	1.0

Table 1: Properties of the geodesic grid. The average cell size is the average spacing between grid points

2.1. The Colorado State University Family of Geodesic Models

The geodesic grid described in the previous section provides a starting point for a uniform highly resolved GCSRM. The next step in building such a model is the choice of equations to be solved on the grid. At the coarse resolutions of the climate models used in the IPCC AR4 and the upcoming AR5, the hydrostatic approximation is generally used. This simplifying assumption, valid when the horizontal extent of a cell greatly exceeds its vertical extent, results in the vertical component of the wind being treated as a diagnostic rather than a prognostic field. At the ultra-high resolutions required to resolve cloud systems, the assumptions behind this approximation breakdown and the atmospheric flow has non-hydrostatic features. Not all atmospheric waves in a fully non-hydrostatic system are important to weather prediction or climate simulation. In particular, resolution of vertically propagating sound waves are unnecessary in these applications. If not somehow damped, they impose a severe time step restriction due to the large speed of sound and its effect on the Courant condition. The anelastic approximation damps these waves by neglecting the local time derivative of density in the continuity equation. This approximation uses a uniform reference state to advance the solution that is not a particularly good assumption for Earth with its wide variations from equator to pole. By contrast, the "unified" system (Arakawa and Konor 2009) restores the local time derivative of density in the continuity equation but as a diagnostic rather than a prognostic field in the primitive equations and does not use a reference state. The unified system damps the vertically propagating sound waves while permitting accurate representation of other wave processes on a wide

	momentum red	**vector vorticity blue**
hydrostatic	available	not planned
anelastic	available	available
unified	in development	in development

Table 2: The CSU family of geodesic grid based global atmospheric models.

range of horizontal scales, from three-dimensional turbulence to long Rossby waves. This system is fully compressible for quasi-hydrostatic motion and anelastic for nonhydrostatic motion. The continuous equations conserve both mass and total energy. Hence, it is ideal for use in GCSRMs.

At Colorado State University (CSU), two alternative sets of primitive equations are under study. The first, referred to as the *red* approach, is more traditional and is based on momentum. The second, based on vector vorticity, is referred to as the *blue* approach (Arakawa and Konor 2009). A set of codes for each of these primitive equation formulations is under development. A pair of codes based on the anelastic approximation has been developed first, to be followed by extensions to the unified system. In a comparison of the normal mode solutions obtained by the fully-compressible, unified, nonhydrostatic anelastic, nonhydrostatic pseudo-incompressible and quasi-hydrostatic systems, the unified system produces virtually identical solutions to the fully-compressible system, with an exception that the vertically propagating acoustic waves are filtered in the unified system (Arakawa and Konor 2009). These codes have been designed based on an implicit time stepping scheme to allow for time steps in excess of that imposed by the Courant condition but requiring the solution of an elliptic equation. We note that other icosahedral code designs have used explicit or semi-explicit time stepping schemes forgoing the elliptic equation, for instance (Satoh et al. 2008). For the CSU red approach, the dynamical pressure is obtained from the elliptic solver, for the blue approach, the vertical velocity is the result. In both cases, the elliptic solver poses interesting computational challenges and opportunities that are discussed later in this paper. There are six possible permutations of the CSU codes, summarized in Table 2.

3. Computational Requirements of CSU Family of GCSRMs

In this section, we present a model of GCSRM computational requirements by measuring each of these components separately at resolutions coarser than that required to resolve cloud systems. Extrapolations to such ultra-high resolutions are made after suitable scaling behavior is demonstrated. Because a complete GCSRM is not available, it is not possible at this time to diagnose a working production code. However, the computational characteristics of a GCSRM can be described if each component can be profiled independently. Hence in the analysis that follows, we separate the code into the following components: 1) *Dynamics*: (a) Equations of motion (momentum or vorticity equations), (b) Advection equations, (c) Elliptic solver and (2) *Physics*: (a) Fast time scales, (b) Slow time scales.

In this breakdown, we have separated advection of passive tracer quantities from the solution of momentum or vorticity because the number of tracers is not necessarily fixed. Furthermore, we separated the elliptic solver from the rest of the dynamics as its efficient evaluation presents special challenges. The source terms to the dynamics (i.e. the physics) is further separated into two broad categories: *fast*, those processes evaluated at the dynamics time steps and *slow*, those processes evaluated at larger time steps.

3.1. Equations of Motion

Theoretically, the number of computational cycles required to integrate the equations of motion for a fixed time period scales with the cube of the horizontal grid size. For instance, if the horizontal resolution were doubled, there would be four times as many horizontal grid points. Additionally, the time step would be halved because of the Courant condition thus doubling the number of calls to the relevant routines and loops. However, memory requirements are independent of time step, so this computational requirement scales with the square of the horizontal grid size. The result is that the ratio of required memory size to cycle rate (e.g. bytes/flops) decreases with the inverse of the grid size as resolution in increased. Hence, in our example of a doubling of horizontal resolution, the bytes per flops ratio is halved. This fundamental property of fluid codes is a critical consideration in the design of efficient machines tailored to these applications.

We evaluate this theoretical scaling behavior of the CSU red codes by running the Jablonowski test problem at multiple resolutions (Jablonowski and Williamson 2006). This three-dimensional test problem does not contain radiation or moisture specifications and is a test solely of the dynamical core. Table 3 shows the number of floating point operations required to integrate the momentum and continuity equations for a single day at several resolutions as measured in the CSU prototype red code using the TAU performance monitoring and tuning system (Shende and Malony 2006) and hardware counter data gathered with the PAPI utility (Browne et al. 2000). To verify that the extrapolations of these measured numbers offer a good prediction of the theoretical number of operations that will likely be required for the full simulation at the higher resolutions, we compared actual measured results to extrapolations from neighboring resolutions. The extrapolated results are obtained through simple linear extrapolation by taking the measured highest resolution result and scaling to both finer and coarser resolutions by a factor of eight. The linear trend pre-

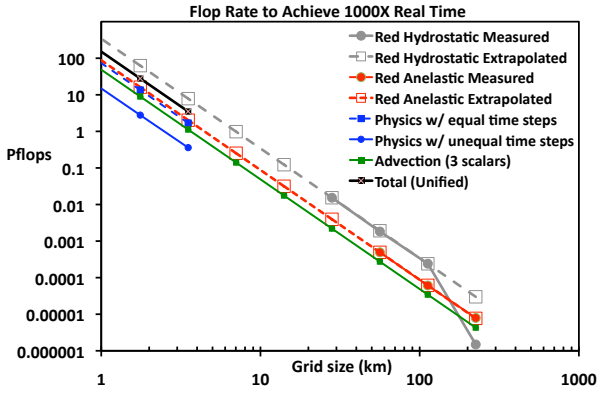


Figure 2: The sustained computational rate (Pflops) required to integrate the hydrostatic and anelastic dynamical cores of the red versions of the CSU GCSRMs with 128 vertical levels. Also shown are similar results for the physics parameterizations and advection routines minimally necessary for a GCSRM.

dicted by our extrapolation fits the measured data very well with the exception of the coarsest resolution hydrostatic result. This is a result of a coding artifact which can be ignored. (This particular code was designed in a way such that at higher resolutions the local process generates only its portion of the grid while at coarser resolutions all processes contain global grid information to offset the communication costs.) Note that the anelastic code, although it involves more prognostic variables, requires fewer operations than the hydrostatic code. We interpret this as a real world phenomena when working with production codes. The newer anelastic code is simply more efficient due to better software engineering. These raw operations counts can be converted into the execution rate to integrate at a throughput of 1000 times faster than real time. The measured CSU codes were configured with 25 vertical levels. This is not enough to adequately resolve cloud systems. A value of 128 for the vertical dimension is a more reasonable choice and will prove useful later for domain decomposition purposes. As changing vertical resolution does not alter the time step, it suffices to multiply the measured operations counts by the fractional increase (5.12) in vertical levels to revise them to higher resolution. Figure 2 shows the sustained floating-point operations rate required to integrate this portion of the CSU red codes at a rate of 1000 times faster than real time expressed in Pflops (10^{15} flops). At the 12th level of discretization, the ~2km target resolution of this study, this sustained computational rate for the red anelastic dynamics code is approximately 16 Pflops.

We also measured the total memory required by the dynamical cores of the red CSU codes. Table 4 and Figure 4 summarize these results for the 128 vertical level versions of the codes. Again, the expected scaling behavior predicted by linear extrapolation provides a good fit to the measured results. In this case, the difference between the hydrostatic and anelastic code requirements is negligible. At the ~2km target resolution, the total memory requirement for this portion of the anelastic code is about 1.8 PB. Finally as expected, our preliminary analysis of the full "unified" model reveals no measurable differences in either memory or operations rate requirements than for the anelastic model.

3.2. Multigrid Elliptic Solver

Like many implicit dynamics schemes, the CSU dynamical cores require the solution of elliptic equations. The current versions of the codes use multigrid (MG) solvers to achieve this. The multigrid solver uses iterative techniques and may require multiple passes to achieve convergence. Since the number of iterations required to reach convergence is data and problem dependent, this adds some uncertainty to the computational requirements for the dycore. In this section, we will describe the parameterized approach to extrapolating the computational cost of the elliptic solve. In Section 5.5 we will show how multigrid computations may be mapped effectively onto a massively parallel computing architecture, and why multigrid efficiency further motivates a hardware design consisting of a smaller number of more powerful many-core chips (processor containing arrays of hundreds or thousands of computational elements per chip) rather than a larger number of computational nodes containing fewer processing elements each.

Multigrid methods provide a powerful technique to accelerate iterative algebraic solvers that are commonly required in computational fluid dynamics problems, but pose challenges to sustained application performance and scalability. A conventional iterative solver operating on a full resolution grid would take many iterations to converge globally due to the slow propagation of error information across the entire grid. Multigrid techniques iterate towards convergence on the full-resolution grid and a hierarchy of lower-resolution (coarsened) representations of the grid data. The coarsened grids accelerate information propagation by damping out errors at large spatial frequencies while the fine grids efficiently damp out high-frequency errors, thus improving the convergence rate for the iterative solver.

MG methods consist of the following primary phases:

- *relaxation:* This is a conventional Jacobi, SOR, or Gauss-Seidel solver iteration involving a stencil sweep over the computational grid and a subsequent halo-exchange to provide updated values for the ghost-cells
- *restriction:* This operator interpolates the fine grid data into a coarsened grid.
- *prolongation:* This operator is the inverse of restriction, where the information on the coarse grid is interpolated back into the fine grid.

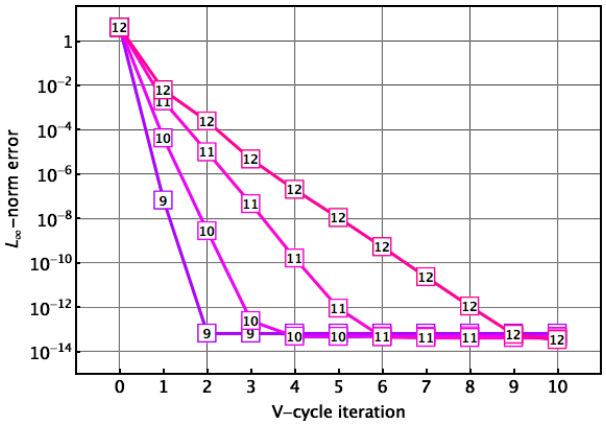


Figure 3: The convergence rate for the multigrid elliptic solver starting from a worst case (identically zero) initial guess and 16 relaxation steps for the bottom-solver on the coarsest grid level for resolutions ranging from the Level 9 grid bisection (∼14km) to the Level 12 grid bisection (∼1.8km) . The horizontal axis shows the number of multigrid V-cycles required to achieve convergence. At the GCSRM scale (level 12), convergence can be achieved with fewer than 9 cycles.

The multigrid method operates in what is called a V-cycle where the restriction operator is used to copy information from the finest grid to progressively coarser grids. In the CSU codes, the coarser grids are half the resolution in each dimension as the fine grid from which they are derived. There is a relaxation step at each level of the V-cycle to squeeze out errors at each level. Current experience running kilometer scale models on existing petascale architectures suggests convergence can typically be achieved in 7-9 V-cycles as shown in Figure 3 given worst-case initial guess. However, we believe convergence can be achieved much faster in practice (fewer than 7 cycles) given the input data to the elliptic solve is seeded with a preconditioned solution using data from the previous timestep. This property would prove useful, for in million-way parallel systems it may not be practical to go all the way to the bottom of the multigrid V-cycle under any circumstances. Rather, early termination of V-cycle and the application of a bottom-solver is far more likely to be the most efficient method.

There are two additional optimizations that are important to deal with explicitly parallel computing systems.

- *repartitioning:* If a coarsened grid becomes so small that it can no longer be domain-decomposed across existing processors, the grid must be re-partitioned amongst a smaller subset of processors so that the V-cycle can continue to lower levels. The theoretical limit case is when the grid is composed of a single cell. Normally a practical parallel implementation would switch to a bottom-solver before we get to this extreme case.
- *bottom-solve:* The coarsening stages of the V-cycle can be cut-off early and an iterative relaxation (bottom-solve) performed until convergence is achieved. This mitigates the worsening exploitable parallelism and communication performance, but may slow the convergence rate because many iterations must be performed to damp out errors in the largest spatial frequencies.

Repartitioning within a multicore chip is relatively inexpensive (nearly free), but repartitioning between nodes in a distributed memory system is extremely expensive. Therefore, we must carefully balance the trade-off between the computational cost of repartitioning against lost convergence efficiency from shifting to the bottom solver too early. In the specific case of the CSU red hydrostatic code, the MG portion of the code consists of layered 2-dimensional solvers where each level of the atmosphere constitutes a separate solver that wraps around the globe using the icosahedral domain decomposition. This simple implementation performs N_L independent multigrid solves, where N_L is the number of vertical levels. The more sophisticated implementation in the Red Anelastic code implements a 2.5 dimensional solver where each layer of the atmosphere is solved in tandem. The latter implementation enables more effective messaging aggregation, which enables the code to send larger messages (above the latency limit) for the halo exchange at each relaxation step. These communication issues are deferred until Section 5.5. Table 3 shows that computational cost of the 2.5 dimensional multigrid package with 5 steps in the V-cycle and 20 steps in the bottom solver is negligible compared to the other major components of the GCSRM.

3.3. Advection Equations

Our estimates of the computational cost of dynamics include advection of momentum (a vector) and temperature (a scalar). A full climate model requires the advection of other scalars necessary for the subgrid scale physics parameterizations. At a minimum, advection of moisture related quantities must be included in any realistic estimate of the total computational burden. The cloud microphysical parameterizations in the System for Atmospheric Modeling code (SAM), a limited area cloud system resolving model, are a good candidate for use in a GCSRM (Khairoutdinov and Randall 2003). An investigation of the advected variables in this code reveals three additional prognostic scalars. These are the total precipitating water, total non-precipitating water and turbulent kinetic energy. The computational requirement to advect three additional scalars in the red anelastic dynamical core is shown in Table 3 and Figure 2. At the ∼2km target resolution with 128 vertical levels, the required sustained computational rate to achieve an integration rate 1000 times faster than actual time is 9 Pflops. We note that modern climate models including sophisticated treatments of aerosol forcings can require many more prognostic scalar

Level	Grid Spacing	Total (Unified)	Red Hydrostatic Measured	Red Hydrostatic Extrapolated	Red Anelastic Measured	Red Anelastic Extrapolated	Physics w/ Equal Time Steps	Physics w/ Unequal Time Steps	Advection	Multigrid
5	225.0		1.3E+11	2.6E+12	6.8E+11	6.7E+11			3.7E+11	3.26E+11
6	112.5		2.1E+13	2.1E+13	5.3E+12	5.3E+12			3.0E+12	1.30E+12
7	56.3		1.6E+14	1.7E+14	4.3E+13	4.3E+13			2.4E+13	5.22E+12
8	28.1		1.3E+15	1.3E+15		3.4E+14			1.9E+14	2.09E+13
9	14.1			1.1E+16		2.7E+15			1.5E+15	8.35E+13
10	7.0			8.5E+16		2.2E+16			1.2E+16	3.34E+14
11	3.5	3.0E+17		6.8E+17		1.7E+17	1.5E+17	3.1E+16	9.7E+16	1.34E+15
12	1.8	2.4E+18		5.4E+18		1.4E+18	1.2E+18	2.4E+17	7.7E+17	5.34E+15
13	0.9	1.9E+19		4.3E+19		1.1E+19	9.4E+18	1.9E+18	6.2E+18	2.14E+16

Table 3: The number of operations required to integrate for a single day the momentum and continuity equations for the CSU red codes with 128 vertical levels. The "measured" values were obtained from a 25 vertical level code and scaled to 128 levels. Multigrid performance was measured using 81 levels, and re-projected for 128 levels with 5 steps in the V-cycle and 20 steps in the bottom solver.

variables. In the Eulerian advection scheme used in the CSU models, the number of operations scales linearly with the number of advected variables. For very large numbers of prognostic variables, semi-Lagrangian or pure Lagrangian techniques are likely to exhibit significant advantages in computational cost.

3.4. Subgrid Physics Parameterizations

An estimate of the computational requirements of the subgrid scale physics parameterizations portion of the model can be directly obtained by diagnosing an existing limited area cloud system resolving model such as the SAM code described in Section 3.3. A critical design choice for this portion of the model is the time step. The Courant condition, necessary for stability of the dynamics, does not apply to the parameterizations. Rather, the physics time step is chosen from accuracy considerations. Although there have been some limited studies on the effects of the choice of time step in coarse resolution global atmospheric models (Williamson 2008, Li et al. 2010), this is an unresearched area in terms of global cloud system resolving models. Limited area models such as SAM, run some portion of the subgrid scale physics at similar time steps to the dynamics. However, at these resolutions, the Courant time step is so small, it may be possible to divide the physics into "fast" and "slow" portions by relaxing the time step in the "slow" parameterizations if changes in their output vary slowly enough.

A prime candidate for such a relaxation is the radiation transport parameterizations as they are very CPU intensive and the solar intensity is controlled by the diurnal cycle, which obviously does not vary with resolution. Arguably, at the high resolutions considered in this study, existing radiation parameterizations in cloud system resolving models do not capture multi-angle scattering adequately. Inclusion of such effects would increase the computational cost even more. Consider then, that the time step for radiation transport and the planetary boundary layer turbulence (another rather slowly varying yet costly parameterization) can be increased to two minutes. Also consider that all the cloud microphysical and other moist physics parameterizations are calculated at the Courant time step. The resulting computational burdens to run the are shown in Figure 2 and Table 3. We show only results for resolutions under 4 km, as these parameterizations are not applicable to coarser models. At the ∼2km target resolution with 128 vertical levels, the required sustained computational rate to achieve an integration rate 1000 times faster than actual time is 2.8 Pflops. We note that separating this portion into fast and slow portions affords considerable speedup. If all components of the physics parameterizations are calculated at the Courant limit, the required sustained computational rate is 13 Pflops, almost as much as the dynamics. At high resolutions, the stability and accuracy of the model can be dependent on the relationship of the dynamics and physics timesteps. These two estimates bracket the range of possibilities. We did not measure the additional memory required by the physics parameterizations; however, as each additional prognostic variable requires only an additional 173 GB at ∼2km target resolution, we are comfortable with asserting that the addition of temperature and moisture variables will not be a significant increase in total memory requirements.

3.5. Total Computational Burden

Our estimates for the total throughput rate required to integrate a complete GCSRM at 1000 times faster than real time are obtained by adding together our estimated requirements for the red anelastic dynamics, the multigrid solver, the advection of three additional prognostic scalars and the physical parameterizations and are shown in Figure 2 and Table 3. We assume that the additional cost of modifying the dynamics to solve the

Level	Grid Space	Red Hydrostatic Measured	Red Hydrostatic Extrapolate	Red Anelastic Measured	Red Anelastic Extrapolate
5	225.0	0.10	0.12	0.11	0.11
6	112.5	0.39	0.47	0.43	0.43
7	56.25	1.68	1.89	1.73	1.73
8	28.13	7.54	7.54		6.94
9	14.06		30.18		27.75
10	7.03		120.71		110.99
11	3.52		482.85		443.97
12	1.76		1931.39		1775.87
13	0.88		7725.54		7103.47

Table 4: The total memory (TB) required for the momentum and continuity equations portion of the CSU red codes with 128 vertical levels.

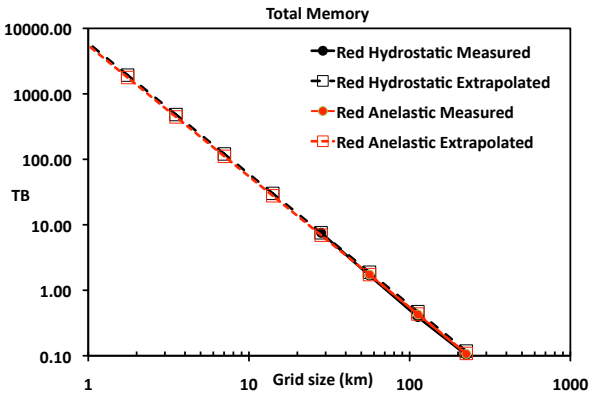


Figure 4: The total memory (TB) required for the momentum and continuity equations portion of the CSU red codes with 128 vertical levels.

unified set of dynamics equations is negligible based on our preliminary measurements. At the ~2km target resolution with 128 vertical levels, a machine containing at least 1.8 PB of addressable memory and capable of sustaining at least 28 Pflops is minimally required for this task. These requirements are machine architecture independent and provide a starting point for a hardware/software co-design. Interprocessor communications requirements are machine and implementation dependent and a critical element of co-design. One possible set of requirements is presented in following sections of this study.

Uncertainty in these estimates of machine requirements is difficult to quantify. Clearly, analysis of different approaches to the solutions of the dynamical equations, including different grids and different solution techniques, would yield different estimates. Also, other radiation and cloud microphysics parameterizations could require more or less computation. However, for the particular codes analyzed here, our estimates may be on the high side as professional software analysts have not attempted any optimization. Automated approaches to optimization will prove critical to our view of hardware/software co-design and some aspects of this are discussed in later sections of this paper.

3.6. Climate model co-design

As stated in the introduction, one of the principal purposes of this study is to use a snapshot of a GCSRM to demonstrate that hardware/software co-design provides a route to a rapid power and cost efficient integration of these ultra-high resolution models. As described in the following sections, many of the hardware and software engineering issues can be explored via automated tools. This is not true for algorithmic aspects of climate model design where changes to solution techniques must be exhaustively analyzed as to their validity in the regimes of interest. It is beyond the scope of this paper to provide quantitative rationales for one algorithm or another. However, we will speculate on which aspects of GCSRM algorithms may provide leverage.

Model intercomparison projects (Taylor et al. 2009) have shown that confidence in future climate projection is increased by the usage of multiple models. Hence, it is vital that any actual machine developed by a hardware/software co-design strategy be able to efficiently execute multiple codes with very different underlying algorithms. As discussed in Section 2, the underlying mesh is fundamental to design of the dynamics algorithm and credible GCSRMs will exploit multiple variations. Time stepping algorithms also provide multiple opportunities for optimization. Fully-implicit, semi-implicit and explicit methods in the dynamics equations all have different computational properties. We examine the multi-grid method of solving the elliptic equation dictated by implicit techniques in details in Sections 3.2 and 5.5. Other methods of elliptic equation solution may be more or less suitable to co-design. Additionally, explicit time stepping may be found to be superior or inferior to implicit time stepping in such a context. The time-splitting algorithm between the dynamics and subgrid scale physics was analyzed briefly in Section 3.4. More computationally expensive parameterizations may prove to be necessary for both moist physics and radiation transport. The explicit inclusion of multi-angle shortwave radiation scattering may prove to be necessary as grid cells approach the same scales in the horizontal and vertical dimensions. Modern climate models are including more radiatively active trace components in the atmosphere including as aerosols and other gaseous compounds. These components must be advected which could lead to this portion of the model dominating the computational expense. It is clear that every portion of climate model algorithms must be examined in a co-design spirit if execution is to

be optimally efficient.

4. Embedded Processor Supercomputing and Hardware/Software Co-design

The described computational rates require machines far faster than what is available today. In previous studies (Wehner et al. 2008; 2009, Donofrio et al. 2009), we discussed how the power requirements of exascale machines extrapolating from architectures based on personal computer or server technologies could be prohibitively expensive. Consistent with other studies (Simon et al. 2008, Kogge et al. 2008, Shalf et al. 2011), the electric bill of facilities drawing in excess of 150MW would be a major, if not the largest, portion of the total cost of ownership. Also related to power concerns, we discussed how the trend in CPU design is not towards faster clock speeds but towards multiprocessor chips, as the number of transistors per unit area continues to grow by Moore's Law. Hence the overall computing rate of a single chip will continue to increase but greater parallelism is required of applications to exploit these increases.

Our previous studies thus presented a new approach to scientific computing. Two decades ago, high performance scientific computing changed from using highly customized vector based machines with a few processors to massively parallel processor machines based on personal computer and server technologies driven by the consumer market. Presently, we are witnessing another significant change in the consumer market that can impact high performance scientific computing, namely the widespread usage of portable consumer electronics devices such as smart phones, e-books and digital cameras. This segment of the hardware market has already surpassed the personal computer segment in a financial sense and is expected to continue to grow for the foreseeable future (Shalf 2007).

In our previous papers, we argued that the "embedded processor" technology behind this current market shift is ideal for the construction of specialized supercomputers at much lower acquisition and power costs. Embedded processor chip designers use pre-verified functional units to rapidly produce semi-custom designs. In this new model for high performance computer design, the commodity product basis is not the hardware components themselves, but rather the embedded processor design toolset. Just as consumer electronics chip designers choose a set of processor characteristics appropriate to the device at hand, high performance computer designers can choose processor characteristics appropriate to a specific application or class of applications, such as GCSRMs. The resulting machine, while remaining fully programmable, would achieve maximum performance on the limited target set of applications. In that sense, such a machine is less general purpose than the typical supercomputers of today. However, the savings appear to be so favorable that multiple machines, each targeted to one of the few truly exascale applications areas, would be more cost effective than a single general purpose exascale machine.

The rapid chip design cycle enabled by embedded processor toolsets enables a true co-design process of hardware and software. In our previous work we described three technologies that enable efficient co-design for scientific applications. The first of these, called "auto-tuning", allows for automated code optimization for the myriad of possible chip designs. The second, known as co-design, iterates between the optimal auto-tuned code and the chip designs themselves. Finally, these steps can be greatly accelerated by simulating the chip performance on Field Programmable Gate Arrays (FPGAs) rather than using chip emulators written in software. Examples of these technologies are discussed later in this paper.

5. A Strawman Machine Design

We now present an overview of our proposed co-designed system specifically designed to run ultra-high resolution, cloud-resolving simulations. We have named our climate computer concept design "Green Flash" after the well known atmospheric phenomenon. More details of the Green Flash project and architecture can be found in recent publications (Donofrio et al. 2009, Wehner et al. 2009, GreenFlash 2010).

5.1. Multi-million Way Parallelism

Our fundamental premise behind the use of embedded processor chips in a targeted supercomputer is that there is enough parallelism in the applications to scale to the requisite overall processing speed. GCSRM codes are good candidates for this hardware/software co-design as the number of three dimensional grid points is in excess of twenty billion. Furthermore, there are opportunities to exploit the nested levels of parallelism afforded by multi-processor chips.

From Table 1, the icosahedral discretization of surface of the globe contains 167,772,162 vertices at our ∼2km target resolution. A logically rectangular two-dimensional domain-decomposition strategy can be used to partition this geodesic grid among processors. A regular decomposition into 64 grid points arranged in an $8X8$ configuration results in 2,621,440 horizontal domains. The vertical dimension offers additional parallelism. Assuming that we could decompose the 128 layers into eight separate vertical domains, the total number of physical subdomains could be 20,971,520. If the communications demands in this decomposition are practical, 20-million way parallelism could be obtained on a GCSRM of this scope by assigning a single processor core to each distinct subdomain. The generation of this domain decomposition can be visualized similar to the generation of the geodesic grid in Figure 1. The first panel of Figure 5 shows the icosahedron projected onto a sphere, while the middle panel has the border between certain of the original triangular faces of the icosahedron removed resulting in parallelograms. These parallelograms, projected onto the sphere, can be successively bisected until the desired decomposition is achieved.

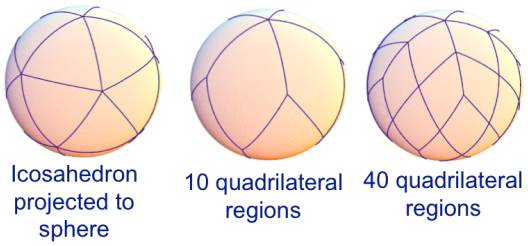


Figure 5: A visualization of the generation of a regular domain decomposition of the geodesic grid.

5.2. Processor Requirements

Putting aside the rather complex communication issues for the moment, it is straightforward to estimate minimum requirements for the most basic of the individual processor characteristics. In Section 3.5, we determined that a total *sustained* computational throughput of 28 Pflops is required to integrate a ~2km/128 level version of the CSU GCSRM. This translates to an averaged sustained computational rate of 1.3 Gflops for each computational core in the strawman machine design described above. From Table 4, the total memory footprint for such a code is less than 2 PB leading to a requirement that each processor core have about 1 GB of accessible memory. Additionally, further code optimization could reduce both of these requirements. Methodologies to optimize other processor characteristics, such as cache size, energy requirements, and chip area are discussed in Section 6.

The number of processors (cores) on a single chip is an important consideration with ramification to both machine and code design. Using current 45nm fabrication techniques, 128 processing cores on a single chip is feasible. With this technology, the strawman machine design described above would require 163,840 chips. Future conventional fabrication techniques of 22nm and beyond (anticipated in the 2016-2018 timeframe) would enable 512 or more processing cores on a single chip, requiring only 40,960 chips for the full system — which is fewer than the number of nodes in contemporary Petascale supercomputers such as BlueGene and Cray XT systems. Although the individual processing core requirements would remain the same in these two scenarios, significant differences in the overall machine design would be affected. On the one hand, mean time to failure would likely decrease with fewer components of the 512-core configuration and the network topology would be simpler. However, as discussed in the next section, demands on that network would be more severe.

5.3. Cache Size Requirements

One of the major consumers of chip resources, including area and energy, for processors is on-chip cache, which is essential for performance. Caches attempt to mitigate large wait times to read and write from main memory by exploiting temporal and spatial locality. Ideally, caches should be sized to guarantee that computation occurs at close to peak rates (that is, ensure the code is not memory bandwidth-bound) while not being so large as to result in wasted resources and power. One possible way to ensure this happens is to study the code at the granularity of each computational loop, and ensure that each loop has enough cache to store all data items accessed; this way, caches are only flushed after each loop, and the data used by the loop that follows can be brought into cache simultaneously or double-buffered to overlap with computation.

In order to understand caching requirements for GCSRM codes, we designed a simple experiment using the functional simulator described in Section 6.3. First, we ran the code using the expected subdomain size and recorded the addresses accessed for each computational loop. Then, these addresses were analyzed to determine the number of unique cache lines for each loop; this gives a reasonable estimate for the cache size required for the loop. Using this methodology, the maximum cache footprint of the hydrostatic red code is approximately 256 KB, before tuning. This is the upper bound on the per-core cache memory that should be included on-chip. Autotuning techniques can further reduce this (see Section 6).

This on-chip cache is sized to capture all temporal recurrences for the stencils in the code. Hence, for any given loop in the code where the same piece of data is referenced more than once, the cache is chosen large enough so that the data will not need to be fetched again from main memory (it would be resident in-cache). The amount of on-chip memory required to capture these temporal recurrences is far smaller than the total amount of memory for all of the model information. It also ensures that the ratio of execution rate to the rate memory reads would remain the same (thus the bytes/flop ratio remains constant).

A previous analysis of a regional version of the blue anelastic CSU dynamical core (Donofrio et al. 2009) using autotuning (see Section 6.1) and local store programming (Williams et al. 2007) shows that 0.1 bytes per flop is sufficient for achieving the target sustained computation rate per core and permitting the memory bandwidth requirement to be to be well within the capabilities of existing technology.

5.4. Nearest Neighbor Communication

Except for the elliptic solver necessary for implicit time stepping schemes, all horizontal communications required for this domain decomposition involve each subdomain's nearest neighbors. In domain decomposition strategies such as this one, information is often communicated through "border" cells, a set of additional cells surrounding the subdomain representing the edge cells in the nearest neighbor subdomains. The number of messages in the dynamical core is fixed for each time step as is the distribution of message sizes. For

a logically rectangular mesh, these types of communication costs can be reduced by carefully ordering the messages. The first set, communicating to the neighboring domains above and below (in logical space), is larger and are individually $8(i+2n_b)n_b(k+1)n_v$ bytes in size. Here, i is one of the horizontal dimensions of the subdomain, k is its vertical dimension of the subdomain, n_b is the number of border cells and n_v is the number of variables communicated.

The second set of messages, communicating to the right and left of the subdomain are individually $8jn_b(k+1)n_v$ bytes long, where j is the other horizontal dimension. We have measured the communication patterns in the CSU global anelastic model using the Integrated Performance Monitor (IPM) message profiling tool (Skinner June 21-24, 2005) and determined that there remains ample opportunity for optimization of communications. These opportunities include removal of corner messages by the method just stated as well as by consolidation of messages. However, these measurements can serve as an upper bound on the specifications for the strawman machine design considered here.

From the IPM tool, we find that in the CSU anelastic global momentum equations code each subdomain (processor) performs 199 messages per time step in the horizontal dimension when configured with 64 horizontal cells in an $8X8$ configuration. Associated with this is a total volume of data communicated per time step of less than $k18,000$ (bytes), where k, the vertical subdomain dimension is determined by the vertical decomposition and the total number of levels. In the 20,971,520 domain strawman design, there are eight vertical domains dividing up 128 levels. This results in a value of 16 for this vertical dimension and hence an upper limit of 288 KB for the amount of data communicated by each subdomain per time step.

The dynamical time step, even in implicit schemes, scales as the Courant condition. Table 5 shows the time steps at various resolutions for the CSU dynamical cores. To reach the 1000 times faster than real time performance metric, each dynamics cycle must be executed 1000 times faster than the time step values in Table 5. At level 12, the target resolution of this design study, the dynamical time step is about 1 second indicating that each dynamics cycle must be completed in 10^{-3} seconds.

At this point in our analysis, we consider that the amount of time spent in communications routines is a hardware/software design constraint. An aggressive target is to spend no more than 10% of the total execution time communicating. Under this constraint, the communications must complete 10,000 times faster than the time steps of Table 5. In other words, at level 12, each processor must send and receive 2 million messages per second at a total rate of about 3 MB per second in the horizontal directions.

We have not explicitly analyzed the vertical domain decomposition strategies in the existing CSU codes as they have not been designed to work in that manner. However, we can imagine two general strategies. The first would be to decompose the vertical domain into slabs and communicate the necessary data, some of which would contain the entire column. The second would be to alternate between domains containing slabs and the entire column as has been done in the Community Atmospheric Model (Mirin and Worley 2009). An optimal strategy might be either of these or even a combination of them and is left for future research. In any event, it is highly likely that communication volumes would be significantly larger for a vertical decomposition than we have measured for the horizontal decomposition.

Level	Grid Spacing	Time Step (sec)
5	225.00	120.00
6	112.50	60.00
7	56.25	30.00
8	28.13	15.00
9	14.06	7.50
10	7.03	3.75
11	3.52	1.88
12	1.76	0.94
13	0.88	0.47

Table 5: Time step (seconds) used in the CSU momentum equation (red) codes at various resolutions.

The number of processors contained on a single chip and the manner in which subdomains are distributed on that chip is another important hardware/software design constraint. On-chip communication is anticipated to be at least two orders of magnitude faster than off-chip communication through a network. Since the vertical communication requirements would be so large, it makes sense to assign an entire stack of vertical subdomains to processors contained on a single chip. In the 20,971,520 domain strawman design, each of these stacks would require 8 processors. Optimal arrangement of horizontal domains depends on the number of processors that can be placed on a single chip. The 45nm lithography scale available today readily enables 128 lightweight processing elements per chip accommodating 16 horizontal subdomains efficiently arranged in a $4X4$ configuration. The resulting "super-subdomain" shown in Figure 6 could exploit fast on-chip protocols for much of the required communication. Subdomain boundary information interior to the super-subdomain, shown in blue, resides entirely on-chip. The boundary information on the outer edge of the super-subdomain, shown in red, must be sent to the neighboring super-subdomains and requires off-chip communication protocols, such as message passing. The total number of off-chip messages can be significantly reduced if the outer boundary data is collected from the subdomains and the communications performed between super-subdomains rather than between individual off-chip subdomains. The sizes of the individual messages obviously must increase; however, this re-

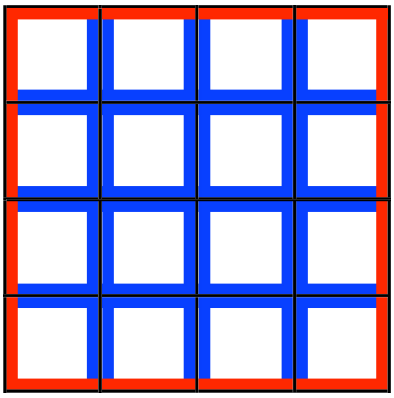


Figure 6: A "super-subdomain" consisting of 16 horizontal subdomains in a $4X4$ arrangement. If assigned entirely to processors on a single chip, the interior boundary information (represented in blue) would be communicated across subdomains utilizing fast on-chip protocols. The exterior boundary information (represented in red) would be communicated between super-subdomains using off-chip networking protocols.

duction in message latency demands comes with no increase in the total amount of off-chip data that must be sent and received.

Our measurements of the message traffic in the CSU anelastic global momentum equations code allow an estimate of the nearest neighbor off-chip communication requirements. By collecting the outer boundary information from all relevant subdomains, both in the horizontal and vertical directions, the off-chip message rate remains at 2,000,000 messages per second for each chip. The increase in the total amount of data to be communicated per dynamical time step depends on the number of subdomains than can be accommodated on a single chip. In the 128 processor per chip case, laid out in a 4x4x8 subdomain configuration, the off-chip communication volume is about 72,000 k_{total}, where $k_{total} = 128$ and is the total number of vertical levels. If the communication is to consume no more than 10% of the total execution time, each chip must communicate an aggregate of 9.2 GB of data to its four nearest neighboring chips every second. In the 512 processor case, laid out in an $8X8X8$ subdomain configuration, this message rate is increased by a factor of 4 to 37 GB/second. However, the topology of the off-chip communication network is simplified in this case as the total number of chips is reduced by a factor of four. Furthermore, large many-core chip system designs will more effectively scale with Little's Law (the maximum number of bytes in flight is equal to the bandwidth times the message latency) than few or single core chip system designs (Little 1961). This stems from an expectation that future improvements in latency are expected to be limited but bandwidth increases will continue to modestly benefit from technology scaling. The "super-subdomain" approach to aggregating nearest-neighbor communications described here is particularly well matched to these hardware constraints as the rate of messages remains fixed and the message sizes are proportional to the number of processing cores per chip. This solution to communication requirements furthers the case for a large many-core design approach to enhancing chip performance. Single (or few-core) chip designs could become severely latency limited as the number of processors approaches tens of millions. Finally, we note that the required communication rates described for the CSU model are well within the capabilities of today's technologies.

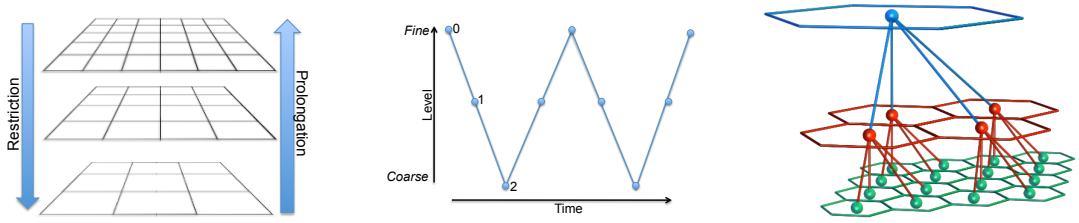
System Architecture	45nm	22nm
Cores per chip	128	512
Clock Frequency	650 MHz	650 MHz
GFlops / Core	1.3	1.3
Cache per core	256 KB	256 KB
GFlops / Chip	166	667
Subdomains per Chip	4x4x8	8x8x8
Inter-chip communication	9.2 GB/s	37 GB/s
Total cores	20,971,520	20,971,520
Total chip count	163,840	40,960

Table 6: Scaling of chip architecture with process technologies.

A summary of the chip requirements using two process technologies, for 128- and 512-cores per chip is shown in Table 6.

5.5. Elliptic Solver Communication

The multigrid solutions to the elliptic equations in the CSU GCSRMs are introduced in Section 3.2. MG solvers typically iterate by successive aggregation to coarser meshes. After some specified number of aggregations, a "bottom solver" is called to finish the calculation (see Figure 7). As the data on the native mesh is distributed across processors, communications are required at each aggregation step. In the strawman design described above, the first several of these communications can be performed using fast on-chip protocols. In the 128 processor chip design, the super-subdomains contain 1024 horizontal cells and would not require repartitioning until the 3rd aggregation in a naive implementation. For the 512 processor chip design, the super-subdomains contain 4096 horizontal cells and would not require communication for such an expensive repartitioning until the 5th aggregation. The off-chip repartitioning is expensive because of the the large volume of off-chip communication required to move all of the grid data to a subset of the nodes. However, without repartitioning, the surface to volume ratio for the ghost cells can become very unfavorable and become ultimately limited by messaging latency. We have proposed a new approach to laying out multigrid computations on a manycore chip that reduces the pressure on costly off-chip



(a) (b) (c)

Figure 7: (a) Multigrid prolongation and restriction (b) V-cycle for a multigrid iterative solve. (c) Graphical representation of a multigrid hierarchy for an icosahedral domain decomposition.

repartitioning that uses functional-partitioning.

The functional-partitioning approach lays out successive levels of the multigrid hierarchy on different subsets of processors on the chip. The biggest challenge posed in the kind of massively parallel system we are considering is that the starting local subdomain size is quite small. Therefore, there is very little room for grid coarsening before arriving at the smallest usable cell size. One must either repartition or move to the bottom-solver and suffer slower convergence. However, a processor design that focuses on a large number of cores per chip remedies this situation. For a chip containing 128-cores, on each restriction (coarsening) phase of the V-cycle, the number of cores performing the relaxation can be reduced by a factor of 4x. Repartitioning the problem within the chip is comparatively inexpensive. In so doing, an efficient algorithm performs a maximum of 4 coarsening steps before repartitioning between nodes, with each stage using 1/4 as many processors per node. At 22nm, with 512 cores per chip, only 2 additional coarsening cycles are permitted before having to do an off-chip repartitioning.

For example, the finest-resolution grid (the top-level of the MG solve), could occupy the local memory of 64 processor cores on the 128-core chip design with a local subdomain size of 16x16x16 (performing the 2.5D elliptic solve on 16 layers of the atmosphere at the time). The coarsened level the next level down in the hierarchy would occupy 32 cores of the chip, and have a local subdomain size of 8x8x16 as its grid is half the resolution of the previous level. This process can be continued recursively for 7 levels, down to a patch of 2x2x16 running on a single core (a total of 127 cores occupied to concurrently represent 7 levels of the multigrid hierarchy as shown in Figure 8). The layout on chip and the partitioning process is very similar to that of a MIPmapping process used for optimizing multi-resolution texture maps for accelerated graphics cards (Williams 1983).

The prolongation and restriction operations can rely on direct communication using the high-speed on-chip interprocessor communication fabric to dramatically improve the energy efficiency of the first 7 levels of the multigrid V-cycle. However, the relaxation timesteps for interim grids would require interprocessor communication for each cycle. Communication can be delayed by increasing the number of ghost-cells to delay the first halo-exchange until we have descended 7 levels in the multigrid V-cycle, but at a cost of doubling the memory footprint for the multigrid solve. This is a design trade-off that we have not yet fully evaluated, but could enable dramatic improvements in performance if the increased memory requirements remain affordable. We note that the current CSU codes use a V-cycle that is 5-7 levels deep and achieves an acceptable convergence rate at the target resolution. Therefore, with a chip design containing at least 128 discrete cores, all of the repartitions can be performed on-chip, avoiding the costly repartitioning across nodes. Furthermore, we can use hardware support for direct inter-processor messaging to pipeline the iteration, restriction and prolongation steps to maximally support overlap of data movement and computation.

5.6. I/O requirements

Data input and output in machine containing in excess of 20 million processors is a daunting and highly speculative problem with practically no published literature. It is, however, likely, that the number of processors that can access disc drives will be a small fraction of the total. The amount of data input to climate models is generally far less than the amount of data output. This is likely to be the case for GCSRMs as well and we do not provide any further analysis of data input, although it will certainly be a challenging problem. Designation of a certain fraction of the machine as output processors and overlapping simultaneous output with the integration of the model is likely to be a favorable strategy to deal with what will be climate model datasets of unprecedented size. In this case, given an estimate of the volume of output data, a minimum output rate can be determined using the time required to integrate a single dynamical time step. This rate would include both the time required for the output processors to receive messages from the integration processors containing the output data and the time required to write the files to disc. Estimating the volume and frequency of

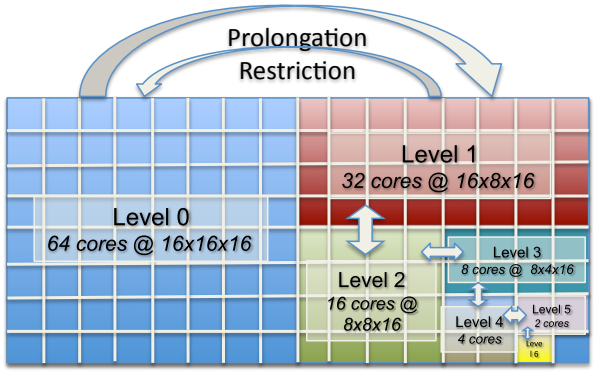


Figure 8: MIPmapping is a technique for storing multi-resolution images (texture maps) in a compact layout for graphics cards. This same layout and partitioning method can be used to store successively coarsened levels of a multigrid hierarchy on many-core chip. A 128 core chip containing the first 7 levels of a multigrid hierarchy in a MIPmap layout is depicted here along with the local subdomain size associated with each core. The fast on-chip interprocessor messaging network can be used to efficiently communication data for the prolongation and restriction operations that interpolate and coarsen data between neighboring layers of the multigrid hierarchy.

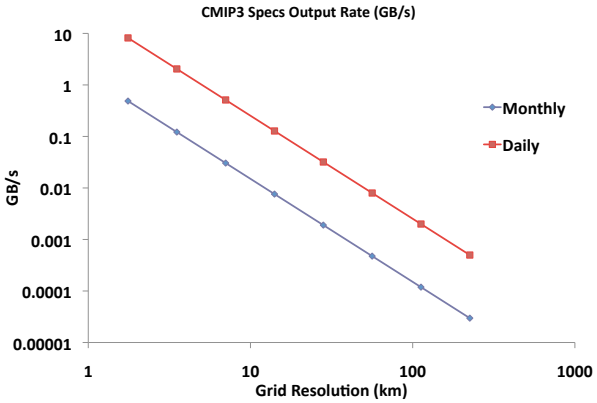


Figure 9: Minimum CMIP3 output rates to maintain a simulation rate of 1000 times faster than real time assuming that I/O processing is completely overlapped with the integration of the GCSRM.

data required to be output is difficult as this is usually determined at run time by the model user. In fact, in many cases, analysis of output at the model's grid resolution is unnecessary. At the targeted ~2km geodesic resolution discussed in this paper, global data sets contain more points than there are pixels on a screen (Prabhat 2010, personal communication). However, regional analyses may be able to exploit this full resolution, so it is likely that some fields will be saved at it.

One credible minimum estimate is to consider the required output for the Coupled Model Intercomparison Project (CMIP3), the standard model configuration for the Fourth Assessment Report (AR4) of the Intergovernmental Panel on Climate Change (IPCC). The CMIP3 protocol specifies output at both monthly and daily intervals. For the atmospheric component, the monthly interval specification is seven fields with all three spatial dimensions and 44 two-dimensional surface fields. The daily interval specification is for four full fields and fourteen surface fields. The rates necessary for the output processors to stay ahead of an integration running 1000 times faster than real time are shown in Figure 9. Note that the total volume of data required by the daily output significantly exceeds the monthly output. At the targeted ~2km geodesic resolution, the minimum CMIP3 output is a modest 8 GB/s. Storage requirements however are significant, the total monthly plus daily output would be 7 PB per simulated century at this resolution. Hence, a single calculation would be considerably larger than the entire multi-model CMIP3 database (Meehl et al. 2005). Higher temporal resolution data is useful for certain analyses at current climate model resolution. It is certain that will also be the case for GCSRMs, further compounding the storage issues.

5.7. Fault Resilience

An important question arises when proposing computing systems consisting of tens of millions of processors: How does one deal with fault resilience? Although the problem is certainly not trivial, neither is it unusual. As long as the total number of discrete chips is not dramatically different, any large-scale design faces the challenge of aggregating conventional server chips into large-scale systems. Across silicon design processes with the same design rules, hard failure rates (i.e. hardware failure) are proportional to the number of system sockets and typically stem from a mechanical failure. Soft error rates (error in signal or datum not cause by hardware defect) are proportional to the chip surface area, not how many cores are on a chip, and bit error rates tend to increase with clock rate. The Green Flash architecture concept is unremarkable in all these respects and should not pose challenges beyond those that a conventional approach faces. Simply stated, our strawman design contains no more parts than do the largest existing scientific computing systems.

To deal with hard errors, redundant cores can be added to the chips to cover defects. An old trick in the memory business, the strategy is apparent in designs such as the 188-core Cisco Metro chip (Eatherton 2005), and it is entirely feasible for hardware/software co-designs as well. Moreover, low power dissipation per chip (7 to 15W) will reduce the mechanical and thermal stresses that often result in a hard error. Soft

errors can be addressed by standard reliability and error recovery design practices in the memory subsystem, including full ECC (error correcting code) protection for all hierarchical levels. Low target clock frequency provides a lower signal-to-noise ratio for on-chip data transfers. Finally, to enable faster rollback if an error does occur, our design makes it possible to incorporate a nonvolatile RAM controller onto each SMP so that each node can perform a local rollback as needed. This strategy enables much faster rollback, relative to user-space checkpointing. These and other chip design parameters can be fully explored by the techniques described in Section 6.3 prior to any actual fabrication.

The Blue Gene system at Lawrence Livermore National Laboratory uses similar fault resilience strategies and contains a comparable number of sockets to our strawman proposal, yet its mean time between failures (MTBF) is 7 to 10 days, which is much longer than systems with far fewer processor cores. Because hardware/software co-design, as we interpret it, tailors the architecture to the application, a machine based on the criteria detailed in this study would deliver considerably more performance than a machine with a comparable number of sockets, thus reducing its exposure to both hard and soft errors. Applying well-known fault-resilience techniques together with novel mechanisms to extend fault resilience, such as localized nonvolatile RAM checkpoints, yields an acceptable MTBF for extreme-scale implementations.

6. Further Processor Design Refinements

As mentioned earlier, the opportunities to reduce the requisite flop rate and memory footprint of a GCSRM are somewhat limited and generally would require costly algorithmic changes. However, other processor design specifications are sensitive to the particular coding constructs. We detail some of these specifications and the techniques developed to optimize both the hardware and software simultaneously in this section.

6.1. Auto-Tuning

In a hardware/software co-design exercise such as described in this study, the number of different processor configurations is nearly limitless. Manual optimization of a large code such as a GCSRM is not feasible for each and every processor option. Even for a single loop on a single processor design, the number of coding possibilities is very large rendering the determination of the optimal result by hand uncertain. As the optimal coding construct rules are different for every loop and every processor design, only an automated search through the space of coding possibilities could feasibly produce a large optimized code. This technique, known as auto-tuning, parses loops written by humans, rewrites them in a large number of predefined ways, and tests each rewritten loop on the processor design in question. The optimal choice may be determined by examining the loops' performance in a high dimensional space of processor design relevant specifications. A principal advantage of this technique is that programmers can write loops in styles that make intuitive physical sense but the processor compiler sees a loop from which it can construct the most efficient executable code.

We present an example using a recently developed auto-tuning framework for stencil computations (Kamil et al. 2010). In this example, we study the buoyancy loop from the CSU blue anelastic vorticity equation model. The procedure begins by parsing the annotated loop, converting it into an intermediate representation. Then, the framework generates candidate versions of the loop with combinations of different optimizations, including loop unrolling and loop reordering. These candidates are run using a generated test harness and timed. Lastly, the best-performing version of the loop, selected on considerations relevant to the co-design, is included in a library to be called by the application. Figure 10 shows results obtained from analysis buoyancy loop from the CSU blue anelastic code, revealing that number of total operations is dramatically impacted by the choice of loop structure and cache configuration.

As a first exercise, consider the impact of a single optimization technique, loop reordering, on cache memory footprint. This exercise alters the order of the nested computational loops which changes the memory access pattern. By reordering loops properly, the cache footprint of the loop is reduced by over 100x from $\sim$100 KB to $\sim$1 KB. In fact, choosing the lowest cache footprint option also changes the mix of total number of operations required to compute the loop. In Figure 11, the mix of operations is shown in the original and optimized loops. Prior to optimization, a little less than half of the total operations are floating point operations, the remainder being integer and control operations necessary to set up the arithmetic. After optimization, these integer and control operations are substantially reduced. The number of floating point operations are not reduced by the auto-tuning, hence dominate the remaining operations.

6.2. Co-Tuning

Conventional approaches to hardware design use benchmarks to search for an efficient hardware architecture. However, the success of software auto-tuning has shown that untuned codes like benchmarks are unable to utilize the full performance potential of the underlying hardware. Thus, this benchmark-based approach to hardware design can lead to sub-optimal hardware designs. As a solution, we propose hardware/software co-design as an approach of using auto-tuned software instead of benchmark codes in the process of hardware design. Since the software is auto-tuned, hardware/software co-design enables the automatic exploration of the optimal hardware and software configurations in a systematic manner. Hardware/software co-design enables scientific application developers to directly influence the design of supercomputers in a coordinated way. Fast emulation platforms using field programmable gate arrays

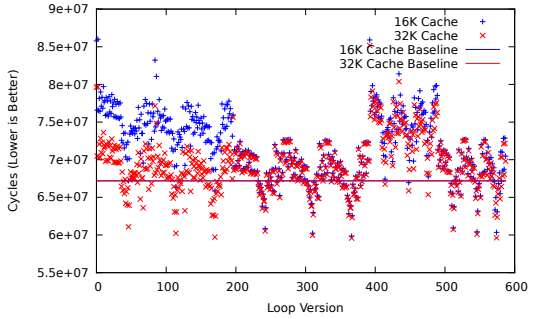


Figure 10: Total cycle count to execute a variety of automatically generated versions of anelastic vector vorticity model's buoyancy loop.

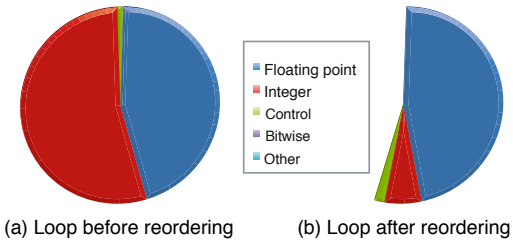


Figure 11: The mix of operations in the anelastic vector vorticity model's buoyancy loop before and after auto-tuning to reduce cache size.

(FPGAs) can accelerate this the design exploration process by orders of magnitude and make this approach practical.

As a demonstration of the co-tuning approach, we co-tuned the buoyancy loop (mentioned in Section 6.1) along with the various cache parameters. We considered different hardware configurations by varying the total cache size, the associativity of the cache and the cache line size. For each hardware configuration, we auto-tuned the buoyancy loop and report the best performance. Figures 12 show the impact of the cache parameters on performance and energy consumption of the buoyancy loop computation respectively. Figure 12(a) shows that auto-tuning can significantly improve performance for small cache sizes. The impact of tuning is small for large cache sizes as the problem can completely fit in the cache. Furthermore, Figure 12(b) shows that although larger caches have the best performance, they have a higher energy consumption as each cache access consumes more energy.

6.3. Rapid design prototyping

Thus far we have discussed potential processor and architecture improvements based on requirements from analysis of individual GCSRM components as well as a methodology to simultaneously optimize hardware and software. The final step of our hardware/software co-design methodology is to develop a highly tuned architecture for specific applications (in our case GCSRMs) based on commodity components and existing intellectual property (IP). The large search space in hardware and software configuration requires the creation of a fast, accurate emulation environment. Traditionally this performance prediction would be done using software simulators. However, the relatively slow execution time of software simulators prohibits the iterative testing of the large number of possible configurations necessary to achieve optimal performance. Furthermore, complex codes such as GCSRMs are not well described by simple kernels leading to the necessity of co-designing with the entire code. To address this shortcoming, we have combined software-based simulation methods with hardware emulation using FPGAs.

The embedded processor toolchain we have chosen generates both a functional model and a performance modeling framework based on C++. While useful for initial software development and initial performance measurements, these simulators break down at real-world problem sizes such as an entire GCSRM. The same toolchain is also able to generate register transfer level (RTL) code that can be synthesized onto an FPGA. Leveraging the FPGA emulation provides a two-order of magnitude speed improvement over comparable software based methods and allows gigabytes of memory to be allocated per processor. The many-core design proposed here for the strawman concept cannot fit on a single FPGA, however, multi-FPGA boards such as available from BEECube or Convey allow tens of cores to be emulated in parallel. The accuracy of FPGAs is inherently better than software based methods, as it requires an actual representation of the hardware to be constructed. The increase in speed and memory available to the emulated cores allow the full application to be benchmarked with realistic problem sizes rather than relying on representative loops. The ability to benchmark full applications on an emulated embedded processor was first demonstrated by our team at Supercomputing 2008. The dynamics portion of a regional version of the blue anelastic CSU atmospheric model was ported to the Tensilica Xtensa LX2 processor and ran at 25MHz. The comparable software simulation of the LX2 runs at ~100 KHz, giving the FPGA emulation environment a 250x speed advantage. These performance modeling techniques are not limited to climate models and have been successfully applied to other applications.

Drawing on a library of independently developed Verilog modules it is possible to quickly assemble multicore systems. We have demonstrated dynamically sizable caches and can extend the configurability to more architectural features, such as DRAM memory interfaces and on-chip network topology. A typical drawback to hardware based emulators is the lack

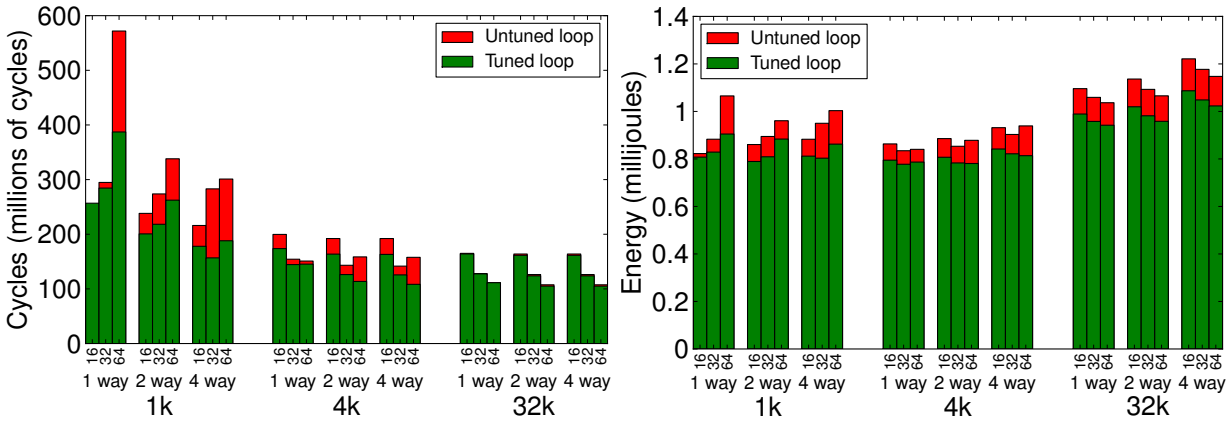


Figure 12: (a) Effect of auto-tuning the buoyancy loop on performance as a function of the cache parameters. The cache line size was varied as 16, 32, 64 bytes, associativity was varied as 1, 2, 4 and total size was varied as 1K, 4K, 32K. The impact of tuning is dramatic on small caches but performance saturates when the cache is large enough. (b) Impact of auto-tuning the buoyancy loop on energy consumed for the processor and cache. Because performance saturates when the cache is large, energy consumption increases for larger cache because they consume more energy per access.

of performance information available. However, the Tensilica processors provide a continuous stream of debug information fully describing the processor's activity. This data can be translated into performance statistics that can be collected enabling the necessary performance profiling for effective co-tuning. Dynamic reconfiguration and detailed performance information enable our FPGA based emulation platform to achieve similar flexibility and state visibility found within software based methods. The value in the credibility of an FPGA based emulation goes beyond providing accurate performance projections. From a practical standpoint, any co-design process must involve both hardware vendors and scientists that will, likely be from differing labs and industry partners. IP issues can become a barrier to innovation as hardware vendors are often reluctant to share low-level details of future designs. To further enhance vendor interaction, the flexible nature of the tools used to rapidly prototype designs can be extended to provide a proxy architecture for vendor specific technologies. The availability of a proxy architecture model that is highly credible is a powerful tool to influence industry as the hardware designers and architects are not constrained by products that must fit into a vendors existing product roadmap.

7. Discussion and Conclusion

We have estimated the computational requirements necessary to integrate a Global Cloud System Resolving Model (GCSRM) at rates appropriate for climate research by analyzing each of its major components separately. We find that a sustained computational rate of 28 Pflops and 1.8 PB of total memory would enable such a model to simulate the atmosphere one thousand times faster than real time. These requirements are machine architecture independent. We have also presented a strawman machine architecture tailored to the GCSRM as an example to illustrate our philosophy of hardware/software co-design. 20,971,520 processors could be mapped onto a global geodesic grid of ~2km resolution with 128 vertical levels using a three-dimensional domain decomposition. An on-chip cache-size of 256 Kb per core would be sufficient to accommodate the working set size of every loop of the dynamics kernel. A more modest 64 Kb L1 cache per core, would fit the working set for more than 80% of the dycore loops. Network requirements depend on the number of processing cores on a chip. In fact, fast on-chip interprocessor communications allows a grouping of subdomains into a "super-subdomain" to significantly simplify communication patterns. For instance, if a single chip contains 512 cores, the strawman architecture would contain 40,960 individual chips allowing an equivalent number of super-subdomains. 2,000,000 nearest neighbor off-chip messages must be sent per second at a bandwidth of 37 GB per second to maintain the desired throughput rate. These architectural specifications are well within other credible estimates for future exascale systems (Simon et al. 2008) (Shalf et al. 2011) (Stevens and White 2009).

The strawman sustained computational rate of 1.3 Gflop per core is a formidable barrier. Modern climate models rarely sustain better than five percent of the overall peak machine performance. The implied peak rate of 27 Gflop per core and

the corresponding overall machine peak rate of 560 Pflop are unlikely in the foreseeable future. Power, cost and heat dissipation conspire against such an architecture. An aggressive hardware/software co-design can increase code efficiency and enable designers to optimize the trade-offs between energy efficiency, cost, and delivered application performance. The increased efficiencies due to hardware/software co-design means that lower peak performances can achieve the required GCSRM integration rates sooner and at lower cost. We have demonstrated that auto-tuning can reduce the number of required non-floating point operations in selected loops from the dynamics, and expose opportunities for more efficient hardware designs. Improved automation of this process to optimize entire codes is an essential component of our co-design philosophy. An accurate census of the mix of instructions as well as other machine relevant quantities required by the GCSRM algorithm allows iteration of processor design tailoring chip characteristics to the GCSRM. This aspect of co-design increases efficiency of the system by reducing waste. Chip complexity and power requirements can be significantly reduced as a result of eliminating hardware resources that would otherwise not contribute to the performance of the application. We have demonstrated that the performance of any desired processor/chip configuration can be efficiently simulated by FPGAs, thus enabling rapid iterative prototyping of candidate co-designs.

One criticism of this approach is that it can result in a highly optimal design for a single code for a single snapshot of time. We agree that pursuing co-design methods to their logical extreme would result in such over-specialized machines. Hence, it is ill-advised to limit a co-designed architecture to a single code as it is well recognized that multi-model simulations are critical to the understanding of the climate system. However, our machine design is comprised of fully general purpose/fully-programmable cores, so the design is flexible enough to handle a broad range of climate code implementations (as demonstrated by our investigation of a broad range of codes in this study) so long as the discretization method is scalable. Our vision for a co-designed architecture must then be able to perform well for a variety of GCSRM algorithms. This necessitates the need to characterize the instruction mix and machine relevant quantities for each model of interest. Other credible kilometer-scale schemes, such as those based on the cubed sphere grid, undoubtedly require different processor characteristics. However, we expect that the differences in machine requirements between one climate code and another are very much less than that required for a full general purpose machine design and that appropriate compromises in co-design are possible.

Our strawman design pushes the limits of what can be accomplished by simple decomposition of the physical domain of the GCSRM. Additional computational burden, whether by increasing model complexity or resolution will push the single processor sustained computational rate past what we believe achievable. For example, to reach our target execution rate for this code, our strawman design required extremely small 8x8x10 subdomain sizes. Further refinement of the subdomains to attempt more parallelism are unlikely to be fruitful due to increased communication burdens. Also, increased single processor clock rates would not maintain optimal energy efficiency. Rather than continuing this SPMD (Single Program Multiple Data) type of parallelism to further improve performance, functional partitioning with different components of the model executing concurrently can overcome these granularity and processor speed limitations. Fast exchange of data via on-chip messaging between cores is a key machine architecture design feature required to permit this additional source of parallelism.

For example in the case of a GCSRM, we could concurrently schedule all or parts of the physics computations with the dynamic core of the climate code, thereby increasing the number of utilized processors and hence overall throughput. However, conventional C and Fortran coding techniques make it difficult to manage such functional partitioning, so there is a new thrust to explore new asymmetric and asynchronous approaches to achieving strong-scaling performance improvements from explicit parallelism (Song et al. 2009, Amato and An 2000). Techniques that resemble class static dataflow methods are garnering renewed interest because of their ability to flexibly schedule work and to accommodate state migration to correct load imbalances and failures for this kind of functional partitioning model. Successful demonstration of such parallelization procedures for a range of leading extreme-scale applications can then be utilized by other similar codes at smaller scales, which will accelerate development efforts for the entire field.

We stress that our results are not a complete estimate for a fully coupled climate model. Additional throughput and memory would be required to include ocean, sea ice, land surface, biogeochemistry and atmospheric chemistry processes. Many existing fully coupled climate models already exploit functional parallelism by assigning different processor sets to each major scientific component of the model (DeLuca et al. 2000). Due to differing computational loads, this is an effective strategy if the right combinations of processors can be determined to load balance the entire model. In a co-designed system, there is no requirement that all the processing chips be the same. If the circumstances dictate, it would be possible to co-design different parts of a machine to be optimal for different model components. Carrying the idea of this type of heterogeneity even further, there is no requirement that all processors on a single chip be identical. This would allow process parallelism within a model component. For instance, for chips assigned to the atmospheric model component, some of the processors could be tailored to advection, another group of them tailored to radiation transport, yet a third group tailored to dynamics and so on. Fast on-chip communication between these code portions would permit assignment of multiple processors to a single subdomain in this fashion. In a simpler sense, our strawman idea for vertical decomposition relies on the same basic concept by co-locating different vertical data at the same horizontal points

on the same chip.

References

N. Amato and P. An. Task scheduling and parallel mesh-sweeps in transport computations. Technical report, Parasol Laboratory, Texas A&M University, 2000.

A. Arakawa and C. S. Konor. Unification of the Anelastic and Quasi-Hydrostatic Systems of Equations. *Monthly Weather Review*, 137(2):710–726, February 2009. doi: 10.1175/2008MWR2520.1.

S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci. A Portable Programming Interface for Performance Evaluation on Modern Processors. *International Journal of High Performance Computing Applications*, 14:189–204, Fall 2000.

C. DeLuca, J. W. Larson, L. Buja, A. Craig, and J. Drake. Community Climate System Model Software Engineering Plan 2000-2005, November 2000. URL `http://www.cesm.ucar.edu/working_groups/Software/plan2000-2005`.

D. Donofrio, L. Oliker, J. Shalf, M. F. Wehner, C. Rowen, J. Kruger, S. Kamil, and M. Mohiyuddin. Energy-Efficient Computing for Extreme Scale Science. *IEEE Computer Magazine*, pages 52–60, October 2009.

W. Eatherton. The push of network processing to the top of the pyramid. `http://www.cesr.ncsu.edu/ancs/slides/eathertonKeynote.pdf`, 2005.

ECMWF, 2010. URL `http://www.ecmwf.int/products/data/operational_system/evolution/evolution_2010.html#24June2010`.

GreenFlash, 2010. URL `http://www.lbl.gov/cs/html/greenflash.html`.

R. P. Heikes and D. A. Randall. Numerical integration of the shallow water equations on a twisted icosahedral grid. part i: Basic design and results of tests. *Mon. Wea. Rev.*, 123: 1862–1880, 1995.

C. Jablonowski and D. L. Williamson. A Baroclinic Instability Test Case for Atmospheric Model Dynamical Cores. *Q. J. R. Meteorol. Soc.*, 132(621C):2943–2975, October 2006. doi: 10.1256/qj.06.12.

S. Kamil, C. Chan, L. Oliker, J. Shalf, and S. Williams. An Auto-Tuning Framework for Parallel Multicore Stencil Computations. In *IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, pages 1–12, April 2010.

M. F. Khairoutdinov and D. A. Randall. Cloud Resolving Modeling of the ARM summer 1997 IOP: Model Formulation, Results, Uncertainties and Sensitivities. *Journal of Atmospheric Sciences*, 60:607–625, 2003.

P. Kogge et al. Exascale computing study: Technology challenges in achieving exascale systems. `http://users.ece.gatech.edu/~mrichard/ExascaleComputingStudyReports/exascale_final_report_100208.pdf`, 2008.

F. Li, W. Collins, M. F. Wehner, D. L. Williamson, J. G. Olson, and C. Algieri. Impact of Horizontal Resolution on Simulation of Precipitation Extremes in an Aqua-Planet Version of Community Atmospheric Model (CAM3). *Submitted to Tellus*, 2010.

J. D. C. Little. A proof of the queuing formula: L=λw. *Op. Res.*, 9:383–387, 1961. doi: doi:10.1287/opre.9.3.383.

J. L. McGregor. Semi-lagrangian advection on conformal-cubic grids. *Mon. Wea. Rev.*, 124:13111322, 1996.

G. M. Meehl, C. Covey, B. McAvaney, M. Latif, and R. J. Stouffer. Overview of the Coupled Model Intercomparison Project. *Bulletin of American Meteorological Society*, 86: 89–93, 2005.

A. A. Mirin and P. H. Worley. Issues and Techniques for Performance Optimization at Scale: Recent Progress with the Community Atmosphere Model. LLNL Report LLNLPRES-411317, Oakridge National Laboratory, Oakridge, Tennessee, USA, 2009. Climate 2009: Eleventh International Specialist Meeting on Next Generation Models on Climate Change and Sustainability for Advanced High-Performance Computing.

W. Putnam and S.-J. Lin. Finite-volume transport on various cubed-sphere grids. *J. Comput. Phys.*, 227:55–78, 2007.

M. Rancic, R. Purser, and F. Mesinger. A global shallow-water model using an expanded spherical cube: Gnomonic versus conformal coordinates. *Quart. J. Roy. Meteor. Soc.*, 122: 959–982, 1996.

D. A. Randall, M. F. Khairoutdinov, A. Arakawa, and W. Grabowski. Breaking the Cloud Parameterization Deadlock. *Bulletin of the American Meteorological Society*, 84: 1547–1564, March 2003. doi: 10.1175/BAMS-84-11-1547.

R. Sadourny. Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids. *Mon. Wea. Rev.*, 100:136–144, 1972.

M. Satoh, T. Matsuno, H. Tomita, H. Miura, T. Nasuno, and S. Iga. Nonhydrostatic icosahedral atmospheric model (nicam) for global cloud resolving simulations. *J. Comp. Phys.*, 227:3846–3514, 2008.

J. Shalf. The new landscape of parallel computer architecture. *Journal of Physics: Conference Series*, 78(1), 2007.

J. Shalf, J. Morrison, and S. Dosanj. Exascale computing technology challenges. *Lecture Notes on Computer Science (LNCS 6449)*, 6449:1–25, 2011.

D. E. Shaw et al. Millisecond-Scale Molecular Dynamics Simulations on Anton. In *Conference on High Performance Computing, Networking, Storage and Analysis (SC09)*, 2009.

S. S. Shende and A. D. Malony. The Tau Parallel Performance System. *International Journal of High Performance Computing Applications*, 20:287–311, May 2006. ISSN 1094-3420. doi: 10.1177/1094342006064482. URL http://portal.acm.org/citation.cfm?id=1125980.1125982.

J. Shukla, B. Hoskins, J. Kinter, J. Marotzke, M. Miller, J. Slingo, and M. Beland. Workshop Report: World Modelling Summit for Climate Prediction, Reading, UK, 6-9 May 2008, January 2009. World Climate Research Program Report #131.

H. Simon, T. Zacharia, R. Stevens, et al. Modeling and Simulation at the Exascale for Energy and the Environment (E3). DOE ASCR Program Technical Report, DOE Office of Advanced Scientific Computing Research, 2008.

D. Skinner. Integrated Performance Monitoring: A portable profiling infrastructure for parallel applications. In *Proc. ISC2005: International Supercomputing Conference*, Heidelberg, Germany, June 21-24, 2005.

S. Solomon, D. Qin, M. Manning, M. Marquis, K. Averyt, M. M. B. Tignor, J. Henry Leroy Miller, and Z. Chen. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press, 2007.

F. Song, J. Dongarra, and S. Moore. A scalable non-blocking multicast scheme for distributed dag scheduling. In *Proceedings of the International Conference on Computational Science (ICCS)*, pages 195–204. Springer, 2009.

R. Stevens and A. White. Scientific grand challenges: Architectures and technology for extreme scale computing. http://extremecomputing.labworks.org/hardware/reports/FINAL_Arch&TechExtremeScale1-28-11.pdf, 2009.

K. E. Taylor, R. J. Stouffer, and G. A. Meehl. A Summary of the CMIP5 Experiment Design, 2009. URL \url{http://www-pcmdi.llnl.gov/}.

M. F. Wehner, L. Oliker, and J. Shalf. Toward Ultra-High Resolution Models of Climate and Weather. *International Journal of High Performance Computing Applications*, 22 (2):149–165, May 2008. doi: 10.1177/1094342007085023.

M. F. Wehner, L. Oliker, and J. Shalf. Low Power Supercomputers. *IEEE Spectrum*, October 2009.

L. Williams. Pyramidal parametrics. In *Computer Graphics (SIGGRAPH 83 Proceedings*, 1983.

S. Williams, J. Shalf, L. Oliker, S. Kamil, P. Husbands, and K. Yelick. Scientific Computing Kernels on the Cell Processor. *International Journal of Parallel Programming*, 35(3): 263–298, 2007.

D. L. Williamson. Convergence of aqua-planet simulations with increasing resolution in the Community Atmospheric Model, Version 3. *Tellus A*, 60:848–862, October 2008. doi: 10.1111/j.1600-0870.2008.00339.x.

Acknowledgments.All authors from LBNL were supported by the Office of Advanced Scientific Computing Research or the Regional and Global Climate Modeling Program of the Office of Biological and Environmental Research in the Department of Energy Office of Science under contract number DE-AC02-05CH11231. In addition, LBNL authors were supported by the LBNL Laboratory Directed Research Program project, Holistic Approach to Energy Efficient Computing Architecture. David Randall, Celal Konor, Ross Heikes and Hiroaki Miura were supported by Cooperative Agreement DE-FC02-06ER64302 from the U.S. Department of Energy. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.