SAND2015-2679C

Aeras: A Next Generation Global Atmosphere Model

William F. Spotz[1], Thomas M. Smith[2], Irina P. Demeshko[3], and Jeffrey A. Fike[4]

[1] Sandia National Laboratories, Albuquerque, NM, U.S.A., wfspotz@sandia.gov
[2] Sandia National Laboratories, Albuquerque, NM, U.S.A., tmsmith@sandia.gov
[3] Sandia National Laboratories, Albuquerque, NM, U.S.A., ipdemes@sandia.gov
[4] Sandia National Laboratories, Albuquerque, NM, U.S.A., jafike@sandia.gov

Abstract
Sandia National Laboratories is developing a new global atmosphere model named Aeras that is performance portable and supports the quantification of uncertainties. These next-generation capabilities are enabled by building Aeras on top of Albany, a code base that supports the rapid development of scientific application codes while leveraging Sandia's foundational mathematics and computer science packages in Trilinos and Dakota. Embedded uncertainty quantification (UQ) is an original design capability of Albany, and performance portability is a recent upgrade. Other required features, such as shell-type elements, spectral elements, efficient explicit and semi-implicit time-stepping, transient sensitivity analysis, and concurrent ensembles, were not components of Albany as the project began, and have been (or are being) added by the Aeras team. We present early UQ and performance portability results for the shallow water equations.

Keywords: Global atmosphere model, performance portability, uncertainty quantification

1 Introduction

Today's climate system models, and their atmospheric components specifically, are the products of decades of research and development. They boast high-order accuracy, long-term stability, and a high degree of efficiency on today's computer architectures [1]. As we look at future climate modeling requirements, two capabilities stand out as both necessary and difficult to back-port to existing codes. The first capability is performance portability, or the ability of a code to perform with a reasonable degree of efficiency on newly emerging architectures, without having to be re-written. The second capability is uncertainty quantification (UQ), the ability of a code to propagate uncertainties in problem parameters through a model to determine their effect on output quantities of interest.

The Albany project at Sandia National Laboratories seeks to provide a software foundation for agile development of scientific application codes that support these capabilities as well as many others. Built upon Sandia's Trilinos [6] and Dakota [4] suites of scientific and optimization software packages, respectively, Albany is a parallel, implicit, unstructured-grid finite element

code that enables advanced analysis techniques spanning template-based generic programming, optimization, UQ, adaptivity, and model order reduction. Albany has already served as the foundation for a continuum mechanics code, a quantum device simulator, an ice sheet model, and other scientific applications.

The Aeras Project ("aeras" is Greek for "air") seeks to use Albany (and the technologies it leverages) to develop a next-generation global atmosphere model, with all of the features one would expect in a modern atmosphere code, plus UQ and performance portability.

2 Albany: A Foundation for Scientific Application Codes

Within Albany, systems of PDEs are conveniently discretized by the finite element method. The developer is responsible only for writing element-level integral kernels for the differential operators and source terms, and defining degrees-of-freedom and equations as a collection of evaluators. Time integration, nonlinear and linear solvers, parametric sensitivities, and file I/O are available to the developer with minimal programming effort.

Albany makes extensive use of the Trilinos package Sacado [13] for automatic differentiation (AD). For example, implicit solution techniques for nonlinear equations require a Jacobian (or Hessian) matrix, consisting of the derivatives of the governing equations with respect to degrees of freedom of the problem. Through AD, this matrix can be provided automatically and exactly, absent both human-induced errors in derivation and numerical errors due to approximation. If the developer changes the governing equations by updating the Albany evaluators, these changes automatically propagate to the evaluation of the Jacobian (or Hessian) matrix. A similar approach is employed in Albany for computing tangents, adjoints, and polynomial chaos expansions (for UQ) as needed [12].

3 The Aeras Models

Aeras describes a collection of atmosphere models, ranging from 2D shallow water and vertical plane models, to a 3D hydrostatic model. In the future, this collection will include 2D and 3D nonhydrostatic models. We describe here the governing equations for the shallow water model, the 3D hydrostatic model, and the 2D hydrostatic (vertical plane) model.

3.1 Shallow Water

The Williamson standard test set paper [15] gives eight different categories of the shallow water equations, and several different forms within each category. The preferred form for our purposes here is the advective primitive variable form, given by

$$\frac{\partial \mathbf{u}}{\partial t} = -\omega \hat{k} \times \mathbf{u} - \nabla \left(\frac{1}{2}\mathbf{u} + gH \right), \tag{1}$$

$$\frac{\partial h}{\partial t} = -\nabla \cdot h\mathbf{u}, \tag{2}$$

where $\mathbf{u} = (u, v)$ is the horizontal velocity vector, ∇ is the horizontal gradient operator (that excludes the vertical component), $\omega = \hat{k} \cdot (\nabla \times \mathbf{u} + 2\mathbf{\Omega})$ is the absolute vorticity, $\hat{k}$ is the outward radial unit vector, $\mathbf{\Omega}$ is the earth's rotation, g is gravity, and $H = h + h_s$ with H the fluid-surface height, h the fluid thickness, and h_s the bottom surface elevation.

3.2 3D Hydrostatic

The hydrostatic equations make several assumptions about the magnitude of vertical velocities and derivatives, and so consider horizontal and vertical derivatives separately. To this end, the ∇ operator denotes a 2D, horizontal operator, and vertical (radial) derivatives are denoted with $\partial/\partial\eta$. Similarly, $\mathbf{u} = (u, v)$ is the 2D horizontal velocity. The vertical velocity is $\dot{\eta}$.

We proceed from the discussion in [14]. The prognostic momentum equation is

$$\frac{\partial \mathbf{u}}{\partial t} + (\zeta + f)\,\hat{\mathbf{k}} \times \mathbf{u} + \nabla\left(\frac{1}{2}u^2 + \phi\right) + \dot{\eta}\frac{\partial \mathbf{u}}{\partial \eta} + \frac{RT_v}{p}\nabla p = 0, \tag{3}$$

where $\zeta = \nabla \times \mathbf{u}$ is the vorticity, f is the Coriolis parameter, and $\hat{\mathbf{k}}$ is the unit vector in the vertical (radial) direction. Three quantities are obtained from diagnostic equations, starting with ϕ, the geopotential, given by the following:

$$\phi = \phi_s + \int_{\eta_s}^{\eta} \frac{RT}{p}\,\mathrm{d}\eta'. \tag{4}$$

The vertical velocity $\dot{\eta}$ is obtained from

$$\dot{\eta}\frac{\partial p}{\partial \eta} = -\frac{\partial p}{\partial t} - \int_0^{\eta} \nabla \cdot \left(\frac{\partial p}{\partial \eta'}\right)\,\mathrm{d}\eta', \tag{5}$$

and the virtual temperature T_v is given by

$$RT_v = (c_p - qc_v)T. \tag{6}$$

$\partial p/\partial\eta$ serves as a density-like quantity, and the governing prognostic continuity equation is

$$\frac{\partial}{\partial t}\left(\frac{\partial p}{\partial \eta}\right) + \nabla \cdot \left(\mathbf{u}\frac{\partial p}{\partial \eta}\right) + \frac{\partial}{\partial \eta}\left(\dot{\eta}\frac{\partial p}{\partial \eta}\right) = 0. \tag{7}$$

The final prognostic equation is the energy equation,

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T + \dot{\eta}\frac{\partial T}{\partial \eta} - \frac{RT_v}{c_p p}\omega = 0, \tag{8}$$

where ω is given by

$$\omega = \frac{\mathrm{D}p}{\mathrm{D}t} = \frac{\partial p}{\partial t} + \mathbf{u} \cdot \nabla p. \tag{9}$$

3.3 2D Hydrostatic (X-Z Plane)

The hydrostatic X-Z equations are simply a 2D version of the 3D hydrostatic equations expressed in a vertical plane. Thus we can take the preceding equations and replace vector velocity $\mathbf{u}$ with scalar velocity u, and 2D derivative operator ∇ with $\partial/\partial x$. This gives momentum equation

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{1}{2}u^2 + \phi\right) + \dot{\eta}\frac{\partial u}{\partial \eta} + \frac{RT_v}{p}\frac{\partial p}{\partial x} = 0. \tag{10}$$

The geopotential and virtual temperature diagnostic equations remain unchanged, and the vertical velocity diagnostic equation becomes

$$\dot{\eta}\frac{\partial p}{\partial \eta} = -\frac{\partial p}{\partial t} - \int_0^{\eta} \frac{\partial}{\partial x}\left(\frac{\partial p}{\partial \eta'}\right)\,\mathrm{d}\eta'. \tag{11}$$

Continuity can be expressed

$$\frac{\partial}{\partial t}\left(\frac{\partial p}{\partial \eta}\right)+\frac{\partial}{\partial x}\left(u\frac{\partial p}{\partial \eta}\right)+\frac{\partial}{\partial \eta}\left(\dot{\eta}\frac{\partial p}{\partial \eta}\right)=0, \tag{12}$$

and the energy equation becomes

$$\frac{\partial T}{\partial t}+u\frac{\partial T}{\partial x}+\dot{\eta}\frac{\partial T}{\partial \eta}-\frac{RT_v}{c_p p}\omega=0, \tag{13}$$

with

$$\omega=\frac{\partial p}{\partial t}+u\frac{\partial p}{\partial x}. \tag{14}$$

Addition of the X-Z model into the Aeras collection allows us to focus on a simpler model when potentially studying the effects of different vertical coordinate systems.

3.4 Numerical Methods

These shallow water equations and the 3D hydrostatic equations are solved by discretizing the surface of the sphere with quadrilaterals (for the 2D hydrostatic equations, the x-coordinate is discretized with 1D elements). Formally, these elements are shells, i.e. elements that can be parameterized with two dimensions, that lie on a two-dimensional manifold (the surface of the sphere) that exists within a three dimensional space. As the Aeras project began, shell elements were not supported in Albany, and had to be added by the Aeras team.

Albany was originally designed to support low-order finite elements with linear and quadratic (tensor) basis functions. It is known that these elements are sub-optimal for the governing equations presented here [5]. Nevertheless, preliminary experiments have been performed on these low-order elements, while Albany is being extended to support spectral elements.

Both 2D and 3D hydrostatic models presented in sections 3.3 and 3.2 discretize the vertical transport operators using central-difference approximations to the differential operators [14]. Vertical transport using finite-differences is supported by column data structures. At a given surface element node, the entire vertical column is referenced to that particular node. Since the entire column for each surface node is contained on-processor, finite-differencing is straight forward and requires no parallel data communication.

Column data structures were implemented by redefining element level “gather” and “scatter” routines. Gather routines collect all data necessary to compute element integration kernels from a global vector and provide it in an element centric pattern. Scatter routines distribute the resulting data into a global residual vector. With these specialized gather and scatter routines, Albany evaluators are used to discretize the PDEs. Most of the evaluators are shared between 2D and 3D hydrostatic models. The Jacobian matrix terms are automatically generated through the use of automatic differentiation. Integrals in the vertical direction are conveniently cast as summations on columns [14].

4 Next Generation Capabilities

Aeras is intended to be a collection of modern global atmosphere models, with all of the features one would expect of such models, including massive parallelism and a variety of efficiently-implemented numerical discretizations and time-stepping techniques. The features that we deem “next generation” are support for UQ and performance portability, which we detail here.

4.1 Uncertainty Quantification

Broadly speaking, UQ is the determination of how uncertainties in input parameters propagate through a model and influence output quantities. Uncertainties can be due to a lack of knowledge (epistemic uncertainties) or variability (aleatory uncertainties). This distinction determines how the uncertainties will be represented mathematically. In either case, a method is then required that can propagate the uncertainties through the model. Many methods exist for doing this, but most involve ensembles of simulations, which can become prohibitively expensive, due to the curse of dimensionality.

The simplest approaches are categorized as non-intrusive methods. These methods treat the model as a "black box," choosing input parameters, running the model, and storing output quantities of interest. The only way to reduce the number of samples required in an ensemble is to bring additional information into the problem, and this fact has led to so-called "embedded UQ" methods, such as the stochastic Galerkin approach. In embedded UQ problems, a new set of equations is generated, requiring a new Jacobian. These quantities can be computed exactly using the automatic differentiation techniques described in section 2.

Albany, and by extension Aeras, have software hooks for employing both non-intrusive and intrusive UQ methodologies. In addition to the built-in Albany UQ capabilities, we are implementing a new capability into Albany, that we call concurrent ensembles. The idea is that data for a single simulation can be extended to support data for multiple simulations. The data types in Albany are already abstract to support automatic differentiation, so an ensemble type is trivial in theory. One drawback of the method occurs for iterative methods: if one sample converges in a small number of iterations relative to another sample, then the first process has to wait while the second sample converges. For a transient problem, this mismatch can potentially occur at every time step. The global atmosphere is an attractive candidate for concurrent ensembles, because spectral elements can be (and typically are) run fully explicit, without employing iterative methods.

To demonstrate meaningful UQ, we need a problem with meaningful uncertainties. This is not necessarily the case with the dynamical core models presented in section 3. To address this, we are developing a simple cloud model to couple to the 2D and 3D dynamical cores, because clouds represent the greatest source of uncertainties in an operational atmosphere model.

4.2 Performance Portability

The high performance computing community envisions a programming model for future architectures described as "MPI + x," [7] where MPI is the Message Passing Interface for handling distributed computing, and x refers to a shared-memory model for programming high-concurrency nodes. (Some refer to an "MPI + x + y" model where y refers to other approaches, such as code generation, dynamic runtime, or just-in-time compilation.) The problem is that x refers to several different current (and possibly future) programming models that have been designed for specific architectures and often have very different optimal data layouts and looping constructs. Fortunately, the MPI portion of the new programming model is well-understood and represents the approach for the majority of parallel scientific applications today. The difficulty is that different architectures require different "x" implementations and therefore adoption of a new machine may institute a new "x" that forces an expensive code refactoring effort. For example, a GPU version of the code would be based on CUDA [9] or OpenACC [10] programming models, while a multicore-CPU version might use OpenMP [11] or pthreads [8]. These implementations have different algorithms to achieve optimal performance, and to make matters worse, newer programming models might be on the horizon.

The Sandia strategic path to achieving performance portability on next generation architectures is to develop a programming model that supports both parallel dispatching methods and a data abstraction layer, separating algorithms and their data access from actual memory layout. The advantage of this approach is that application developers can write numerical kernels once and get them to perform efficiently on a wide variety of current and future architectures. Support for these capabilities are collected in a Trilinos software package called Kokkos [2, 3]. Kokkos provides a single portable application programming interface (API) without exposing the programmer to device-specific programming models such as CUDA, pthreads and OpenMP.

A Kokkos `ExecutionSpace` is an abstraction implemented as a C++ template parameter that specifies where code will be executed and which memory space to use, and can be `CUDA` or `CUDAUVM` for NVIDIA GPUs, `OpenMP` or `Threads` for multicore CPUs and Intel Xeon Phi, etc. As an example of a data layout abstraction, the Kokkos `View` class defines data buffers that can exist on the host or on a device such as a GPU, or both, and whose copy operations are always explicit (unless the user relies on CUDA UVM). Kokkos `View`s are written in such a way that the compiler can choose the data layout striding automatically (allowing for both "arrays of structs" or "structs of arrays," depending on context) and optimized for the current hardware.

To support abstracted concurrent algorithms, Kokkos provides `parallel_for` and `parallel_reduce` operations that are templated on the Kokkos `ExecutionSpace`. These parallel operations are specialized by Kokkos developers for each `ExecutionSpace`. The user provides a functor, which is a generic name for an instantiation of any class object that can be called like a function. Each specialization of the templated operations will perform optimized loops according to the `ExecutionSpace` and call the user's functor to execute the numerical kernel. The user must ensure that the functor/kernel is thread safe, e.g. that it avoids race conditions. In order to avoid thread collisions, Kokkos provides "Atomic" operations: when a thread performs an atomic operation, the other threads see it as happening instantaneously and avoid accessing the same data.

At the time Albany was originally developed, Kokkos was not available for adoption by full-fledged applications. Albany has been recently refactored to use Kokkos and distributed linear and nonlinear algebra packages that take advantage of Kokkos. Aeras is a leading application for leveraging these new capabilities.

5 Preliminary Results

We have implemented evaluators for the shallow water, 2D hydrostatic x-z plane, and 3D hydrostatic equations. For the shallow water equations, we have implemented the full Williamson suite of test cases [15]. The discretization method is low order quadrilateral elements (linear and quadratic), as these elements were readily available in Albany. However, it is known that low-order elements are sub-optimal for atmospheric modeling, and current results are consistent with predicted results. For this reason, we are currently engaged in upgrading Albany to support spectral elements.

What is novel about Aeras, of course, is not the discretization method, but rather the UQ support and performance portability. We do have some preliminary results in these areas, presented below.

5.1 Uncertainty Quantification

True uncertainty quantification will have to wait until the cloud model has been implemented, but here we substitute sensitivity analysis as a demonstration that UQ-type calculations can be

performed. Sensitivity analysis is *not* UQ, but for problems with large numbers of parameters, it is a necessary precursor to actual UQ analysis. It determines which parameters the model is most sensitive to, and which parameters can be eliminated from the exploration space. For our purposes here, sensitivity analysis exploits much of the same templated code base that UQ does, and is an important demonstration in advance of meaningful UQ analysis.

We ran shallow water test case 5 [15], which is flow over an isolated mountain. In this simulation, however, we treated the height of the mountain as an uncertain parameter, and then calculated the sensitivities of a prognostic variable, in this case the latitudinal velocity component, with respect to this parameter.

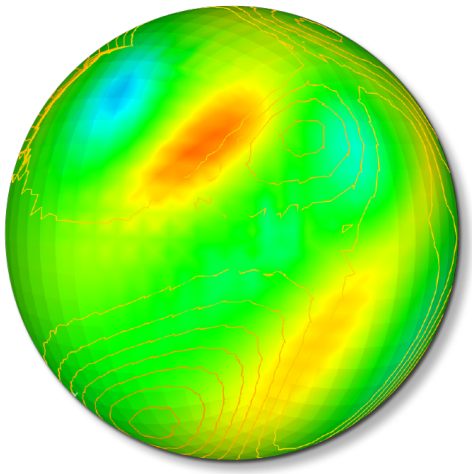

Figure 1: Aeras sensitivities in latitudinal velocities for shallow water test case 5. Sensitivities are with respect to the mountain height parameter.

Figure 1 is a projected plot of colored contours of this sensitivity field onto the sphere. The mountain is on the far side of the sphere and is inducing waves that are depicted by contour lines. In practice, we may be interested in the extrema of this field or the L^2 norm of this field relative to other sensitivities in the problem. Parameters with lower sensitivities will contribute less to the number of required ensembles.

As a preliminary demonstration of the concurrent ensemble approach described in 4.1, we use Dakota to request 32 samples from a uniform distribution over $\pm 50\%$ of the nominal mountain height. Figure 2 compares the effect of varying the number of concurrent ensembles with the time required for 32 sequential evaluations. As the number of concurrent ensembles is increased up to 32, there are competing effects that influence the overall run time. The cost of the one-time mass matrix calculation increases, but the cost per time step for the residual calculations decreases. Therefore, a benefit to using the concurrent ensemble approach can be demonstrated by running the simulation for a large enough number of time steps. For 3000 time steps, the evaluation of 32 concurrent ensembles shows a 40% decrease in the time required compared to 32 sequential evaluations. These are very preliminary results, and it is expected that improvements to the implementation, such as static allocation for a fixed number of concurrent ensembles, will significantly lower the times for the concurrent ensemble approach and demonstrate an increased benefit of the approach.

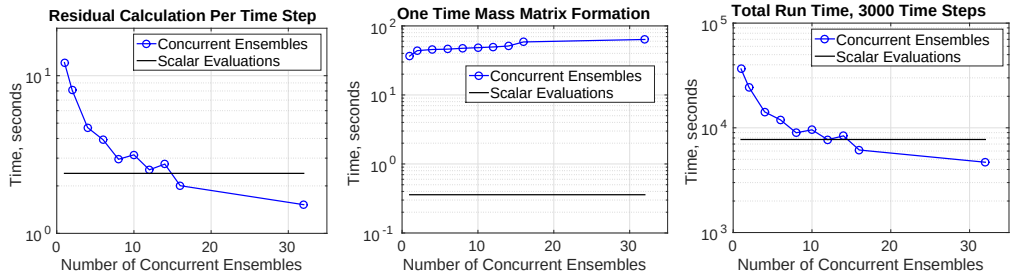

Figure 2: Comparing 32 sequential evaluations of the shallow water code with various numbers of concurrent ensembles.

5.2 Performance Portability

Albany has been recently refactored to use Kokkos for performance portability. It uses Kokkos data types at a base layer which provides portability across different architectures. To run Albany thread-parallel on multi-core architectures we need only replace Albany evaluators with Kokkos functors. We again choose Aeras Shallow Water TC5 code for our performance portability study. We have ported only the Finite Element Assembly part of the code since the linear solver package has not been ported to Kokkos yet. The code includes gathering data to local data structures, performing element integrations, and assembling element data into the global data structure. While the first two steps are inherently thread safe, the last one can have thread conflicts and to prevent this, Kokkos Atomic operations have been used. In our implementation CPU/GPU data flow is managed by CUDA 6.0 unified memory (UVM).

We evaluate the performance of the Aeras code on Sandia's Shannon testbed cluster. Shannon has Intel Xeon CPUs and NVIDIA Kepler (K40x) GPUs, 1 GPU per node. Performance evaluation results for the Aeras Shallow Water TC5 are presented in Figure 3. We compare execution time as a function of the number of elements per workset, total number of elements constant (weak scalability). Albany elements are stored in high-level worksets, whose sizes can be tuned to exploit optimal parallelism on a given platform. Experiments are run on different architectures (NVIDIA K20 GPU and Intel Xeon) along with a single CPU run. The key point here is that the same finite element code base was used for all runs, but each with a different configuration option for the Kokkos ExecutionSpace template parameter.

Figure 3 (a) provides performance results for the total Finite Element Assembly time which includes CPU-GPU communications for the CUDA implementation. The Kokkos OpenMP implementation (16 threads) is up to 9 times faster and Kokkos CUDA implementation is up to 17 times faster than single-CPU implementation. While OpenMP and Serial implementation results are almost flat, CUDA performance increases with higher number of elements per workset. This can be explained by the fact that there is not enough work for the GPU with smaller workset sizes and we are essentially measuring the kernel's call latency and CPU-GPU communication.

Note that we have only Finite Element Assembly part of the code ported to multi-core architecture today. When work on refactoring the linear solver packages to Kokkos is completed we expect improved performance results due to eliminating the CPU-GPU communication. Figure 3 (b) show performance results for the Finite Element Assembly part without Gather and Scatter evaluators where CPU+GPU communications are performed by UVM. Without communication overheads, the Kokkos CUDA implementation is up to 24 times faster that the Serial implementation.

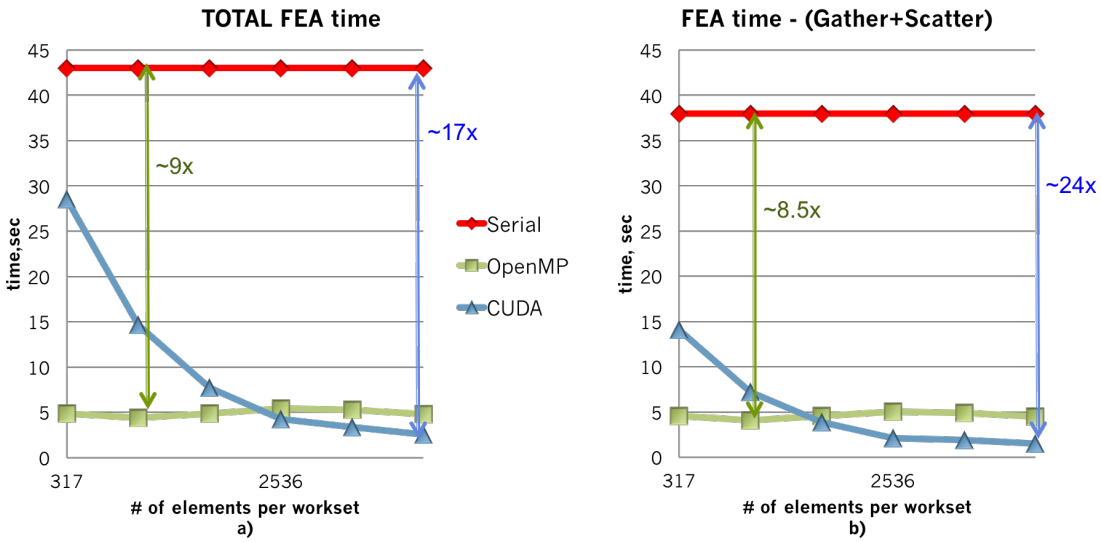

Figure 3: Aeras shallow water weak scalability results: (a) total time as a function of the number of elements per workset; (b) time without gather/scatter as a function of the number of elements per workset

6 Concluding Remarks

The Aeras project has met many milestones as it progresses towards development of a collection of 2D and 3D global atmosphere models with next generation capabilities. We have utilized Albany as a code base, which provides a wide variety of scientific simulation capabilities, by implementing evaluators for 2D shallow water and 2D and 3D hydrostatic models. General capabilities not initially supported by Albany, but added, or being added, by the Aeras team include shell elements, spectral elements, efficient explicit and semi-implicit time stepping, transient sensitivity analysis and concurrent ensembles.

The next generation features of Aeras include uncertainty quantification (UQ) and performance portability. We plan meaningful UQ analysis in the context of a cloud model, currently under development. In anticipation of this, we have utilized Albany features to successfully perform automatically-generated sensitivity analysis, which exercises much of the generic programming that will also enable UQ. We have also demonstrated an initial implementation of concurrent ensemble computation, resulting in faster execution for larger ensemble sizes.

Performance portability in Albany is supported by upgrading to software packages specifically designed for this purpose. This upgrade has sufficiently progressed to demonstrate, with a single user implementation, shallow water tests that run efficiently on diverse high performance computing architectures, showing good performance on both multicore CPUs and manycore accelerator-based machines.

References

[1] John M. Dennis, Jim Edwards, Katherine J. Evans, Oksana Guba, Peter H. Lauritzen, Arthur A. Mirin, Amik St-Cyr, Mark A. Taylor, and Patrick J. Worley. CAM-SE: A scalable spectral element dynamical core for the Community Atmosphere Model. *Journal of High Performance Computing Applications*, 26(1):74–89, 2012.

[2] H. Carter Edwards, Christian R. Trott, and Daniel Sunderland. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing*, 74(12):32023216, 2014.

[3] Harold Carter Edwards, Daniel Sunderland, Vicki L Porter, Chris Amsler, and Sam Mish. Manycore performance-portability: Kokkos multidimensional array library. *Scientific Programming*, 20:89114, October 2012.

[4] M.S. Eldred, A.A. Giunta, B.G. van Bloemen Waanders, S.F. Wojtkiewicz, Jr. W.E. Hart, and M.P. Alleva. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis. Technical Report SAND2001-3514, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185-1110, April 2002.

[5] Katherine J. Evans, Mark A. Taylor, and John B. Drake. Accuracy analysis of a spectral element atmospheric model using a fully implicit solution framework. *Monthly Weather Review*, 138:3333–3341, 2010.

[6] Michael Heroux, Roscoe Bartlett, Vicki Howle, Robert Hoekstra, Jonathan Hu, Tamara Kolda, Kevin Long Richard Lehoucq, Roger Pawlowski, Eric Phipps, Heidi Thornquist Andrew Salinger, Ray Tuminaro, James Willenbring, and Alan Williams. An overview of trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185-1110, August 2003.

[7] Adolfy Hoise, Darren Kerbyson, Robert Lucas, Arun Rodrigues, John Shalf, and Jeffery Vetter. Report on the ASCR workshop on modeling and simulation of exascale systems and applications. Technical report, Department of Energy Office of Science, August 2012.

[8] IEEE Std 1003.1. Technical report, IEEE, 2004.

[9] NVIDIA CUDA Programming Guide version 3.0. Technical report, NVIDIA Corporation, 2010.

[10] The OpenACC Application Programming Interface. Technical report, OpenACC-Standard.org, 2013.

[11] OpenMP Application Program Interface. Technical report, OpenMP Architecture Review Board, 2013.

[12] Roger P. Pawlowski, Eric T. Phipps, and Andrew G. Salinger. Automating embedded analysis capabilities and managing software complexity in multiphysics simulation, Part I: Template-based generic programming. *Scientific Programming*, 20:197–219, 2012.

[13] Eric Phipps and Roger Pawlowski. Efficient expression templates for operator overloading-based automatic differentiation. In *Proceedings of the 6th International Conference on Automatic Differentiation*, July 2012.

[14] Mark A. Taylor. Conservation of mass and energy for the moist atmospheric primitive equations on unstrucutred grids. In Peter H. Lauritzen, Christiane Jablonowski, Mark A. Taylor, and Ramachandran D. Nair, editors, *Numerical Techniques for Global Atmospheric Models*, page Chapter 12. Springer, 2012.

[15] David L. Williamson, John B. Drake, James J. Hack, Rüdiger Jakob, and Paul N. Swarztrauber. A standard test set for numerical approximations of the shallow water equations in spherical geometry. *Journal of Computational Physics*, 102:211–224, 1992.