Performance Analysis of the Parallel CFD Code for Turbulent Mixing Simulations

Tulin Kaman*
Department of Mathematical Sciences
University of Arkansas
Fayetteville, AR
tkaman@uark.edu

Alaina Edwards
Department of Mathematical Sciences
University of Arkansas
Fayetteville, AR
aje004@uark.edu

John McGarigal
Department of Mechanical Engr.
University of Arkansas
Fayetteville, AR
jamcgari@uark.edu

ABSTRACT

Understanding turbulence and mixing due to the hydrodynamic instabilities plays an important role in a wide range of science and engineering applications. Numerical simulations of three dimensional turbulent mixing help us to predict the dynamics of two fluids of different densities, one over the other. The focus of this work is to optimize and improve the computational performance of the numerical simulations for the compressible turbulent mixing on Blue Waters, the petascale supercomputer at the National Center for Supercomputing Applications. In this paper, we study the effect of the programming models on time to solution. The hybrid programming model, which is a combination of parallel programming models, becomes a dominant approach. The most preferable hybrid model is the one that involves the Message Passing Interface (MPI), such as MPI + Pthreads, MPI + OpenMP, MPI + MPI-3 shared memory programming, and others with accelerator support. Among all choices, we choose the hybrid programming model that is based on MPI + OpenMP. We extend the purely MPI parallelized code with OpenMP parallelism and develop the hybrid version of the code. This new hybrid implementation of the code is set up in a way that multiple MPI processes handle the interface propagation, whereas multiple OpenMP threads handle the high order weighted essentially non-oscillatory numerical scheme.

KEYWORDS

Turbulent Mixing, Numerical Simulations, Performance Analysis, Distributed/Shared memory programming

1 INTRODUCTION

Turbulent mixing flows arise in a wide range of science and engineering applications, from climate studies to all forms of fusion, whether the confinement is inertial, gravitational or magnetic. The numerical simulations help us to understand the dynamics of turbulent mixing. Turbulent mixing due to Rayleigh-Taylor (RT) instability arises at the interface between two fluids of different densities whenever the pressure gradient opposes the density gradient.

These problems are deeply multiscale. The level of scales that are desired to be resolved identifies the characteristic properties of the numerical approach, such as Direct Numerical Simulations (DNS), Large Eddy Simulations (LES), and Reynolds Averaged Navier Stokes (RANS). With the power of today's HPC systems, resolving all turbulence length scales is handled by DNS [14]. RANS resolves length scales sufficient to specify the problem geometry. Among these three approaches, DNS has the highest computational cost, and RANS has the lowest. The use of LES reduces the computational cost of the DNS and resolves some but not all length scales. LES was first proposed by Smagorinsky [21] for the study of the dynamics of the atmosphere's general circulation. In LES, the unresolved smaller-scale motions are modeled by subgrid scale (SGS). The multi-species compressible Navier-Stokes equation, filtered at a grid level, is solved, so that the LES defines SGS terms (such as the Reynolds stress) as a source. These source terms are modeled as gradient diffusion terms, and otherwise coefficients such as turbulent viscosity, mass, and thermal diffusivity are recovered in a dynamic manner from the solution itself. This is called a dynamic SGS model. The missing coefficients are computed locally in the simulation [2].

Front tracking (FT) is the technique of storing and dynamically evolving a meshed front that partitions a simulation domain into two or more regions, each representing a different material, or physics model. Front tracking is the unique method presently demonstrated to avoid systematic errors in an important class of problems revolving around turbulent mixing [5]. The sharp resolution of interfaces and steep gradients occurs in a variety of applications, such as primary breakup of a liquid jet [1], forecasting of cloud boundaries [7], target design of muon collider in high energy particle accelerators [4], and electrocardia [23]. The FT/LES/SGS combination has previously been validated for macro, meso, and micro scale observables. By this, This means that diagnostics with comparison to experimental data have been applied to assess the solution accuracy of the turbulent mixing flow at the macro/meso/micro length scales. Thus, the simulations have achieved agreement with experimental measurements in the overall size of the mixing zone (macro), the coherent bubble structure within it (meso), and molecular level mixing (micro) as recorded by chemical reactions [5, 12].

The simulation package, FronTier, has the implementation of all these algorithms. FronTier supports a range of physics, including compressible and incompressible flow, turbulence models, fluid-structure interactions, phase transitions, and crystal growth, each with its own validation and verification studies [3]. It is parallelized with a tensor product domain decomposition. MPI is used to pass states and interface data from one processor to another. FronTier

*Corresponding Author

https://doi.org/10.22369/issn.2153-4136/12/1/7

has adopted object oriented programming. Many major front tracking functionalities have been modularized to allow users to call them with a minimal knowledge of internal operation and coding. An API to modularize the front tracking and to make it available to other simulation codes is constructed [13]. We extend the purely MPI parallelized code with OpenMP parallelism and develop the hybrid version of the code. The focus is to optimize and improve the computational performance and increase the scalability to perform high resolution numerical simulations efficiently on high performance computing systems.

The organization of this paper is as follows. In Section 2, we describe the model problem, Rayleigh-Taylor instability. In Section 3, we present the strong and weak scaling analysis of the purely MPI parallel version of the code and introduce the profiling tool Tuning and Analysis Tool (TAU) that is used to analyze the runtime behavior of the program. We show how to instrument the FronTier code using TAU and present performance results. In Section 4, we introduce the hybrid programming model, which is a combination MPI + OpenMP parallel programming model and compare the purely MPI and hybrid models. Section 5 presents reflections on how the research activities as Blue Waters Interns influence the students' future careers.

2 PROBLEM DESCRIPTION

The study of the turbulent mixing problem in Rayleigh-Taylor aims for macro level validation and to predict the overall dimensionless growth rate of the mixing zone. The growth rate is defined by the formula $h_b = \alpha_b A g t^2$ for the penetration distance h_b of the light fluid into the heavy fluid, g being the acceleration force that defines the instability, and $A = (\rho_1 - \rho_2)/(\rho_1 + \rho_2)$ the dimensionless buoyancy correction to gravity, depending on the density difference between the two fluids. Here, ρ_1 denotes the heavy fluid density, and ρ_2 denotes the light fluid density. The determination of the growth rate α_b has been the source of considerable interest. The numerical simulations are conducted to predict the quantity of interest on the growth rate of the mixing zone. The uncertainty quantification analysis associated with initial conditions, the sensitivity analysis to the parameters such as the initial mass diffusion layer thickness, and the effect of initial conditions and parameters to the quantity of interest α_b were studied in comparison with experimental data and presented in [9–12, 24]. The problem with single-mode, multi-mode, and random initial perturbations is presented in Figure 1. Here, the validation and verification studies are out of the scope of this paper. The focus is to optimize and improve the computational performance of the FronTier software package for the Rayleigh-Taylor turbulent mixing problem on the Blue Waters petascale supercomputer.

For the numerical simulations of multiphase flows, one of the advantages of the front tracking method is in dealing with the contact discontinuities. The front tracking method is used to solve the conservation laws with discontinuities between fluids. The mathematical formulation is based on the filtered Navier-Stokes equations for the multiphase flows [10]. These equations are the governing equations of LES simulations. In the equations of continuity, momentum, energy, and concentration (1) – (4), the variables that have been filtered on the grid scale are denoted by the overbar. There

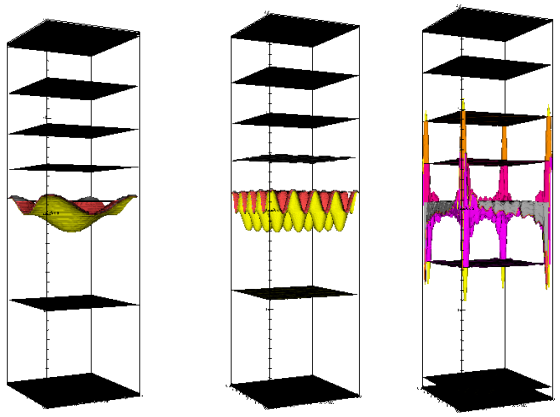

Figure 1: Images of the single-mode, uniform and random multi-mode Rayleigh-Taylor instabilities.

is also a density-weighted filtering operation, which is denoted by the tilde. The Favre-filtered continuity equation (1) is obtained by first applying the grid scale onto the continuity equation

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial \overline{\rho v_i}}{\partial x_i} = 0,$$

then the density-weighted filtering $\widetilde{v_i} = \dfrac{\overline{\rho v_i}}{\overline{\rho}}$.

For the compressible flows, the Favre-filtered continuity, momentum, energy, and concentration equations are obtained as

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial \overline{\rho}\widetilde{v_i}}{\partial x_i} = 0 \tag{1}$$

$$\frac{\partial \overline{\rho}\widetilde{v_j}}{\partial t} + \frac{\partial (\overline{\rho}\widetilde{v_i}\widetilde{v_j} + \overline{p}\delta_{ij})}{\partial x_i} = \frac{\partial \overline{d_{ij}}}{\partial x_i} \tag{2}$$

$$\begin{aligned}\frac{\partial \overline{E}}{\partial t} + \frac{\partial (\overline{E} + \overline{p})\widetilde{v_i}}{\partial x_i} &= \frac{\partial \overline{d_{ij}}\widetilde{v_j}}{\partial x_i} + \frac{\partial}{\partial x_i}\left(\overline{\kappa}\frac{\partial \widetilde{T}}{\partial x_i}\right) \\ &\quad + \frac{\partial}{\partial x_i}\left((\widetilde{H_h} - \widetilde{H_l})\overline{\rho}\widetilde{D}\frac{\partial \widetilde{\Psi}}{\partial x_i}\right),\end{aligned} \tag{3}$$

$$\frac{\partial \overline{\rho}\widetilde{\Psi}}{\partial t} + \frac{\partial \overline{\rho}\widetilde{\Psi}\widetilde{v_i}}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\overline{\rho}\widetilde{D}\frac{\partial \widetilde{\Psi}}{\partial x_i}\right). \tag{4}$$

where $\overline{\rho}, \widetilde{v_i}, \overline{E}, \overline{p}$, and $\widetilde{\Psi}$ are the filtered variables for total mass density, the velocity, the total specific energy, the pressure, and the mass fraction. The total specific energy is

$$\overline{E} = \overline{\rho e} + \overline{\rho v_k}^2/2.$$

$\widetilde{H_h}$ and $\widetilde{H_l}$ are the partial specific enthalpy of each species defined by

$$\widetilde{H_h} = \widetilde{e_h} + \frac{\overline{p}}{\overline{\rho}}, \qquad \widetilde{H_l} = \widetilde{e_l} + \frac{\overline{p}}{\overline{\rho}},$$

where $\widetilde{e_h}$ and $\widetilde{e_l}$ are the specific internal energy of each species. $\widetilde{T}, \overline{\kappa}$, and $\widetilde{D}$ are the filtered temperature, the heat conductivity, and the kinematic mass diffusivity. The viscous stress tensor, d_{ij}, in momentum and energy equations is expressed as

$$\overline{d_{ij}} = \overline{\nu_d}\left(\left(\frac{\partial \widetilde{v_i}}{\partial x_j} + \frac{\partial \widetilde{v_j}}{\partial x_i}\right) - \frac{2}{3}\frac{\partial \widetilde{v_k}}{\partial x_k}\delta_{ij}\right),$$

where $\overline{\nu_d} = \overline{\rho\nu_k}$ is the filtered dynamic viscosity.

The stable and higher order WENO (Weighted Essentially Non-Oscillatory) scheme [19] is used for solving the Favre-averaged Navier-Stokes equations. In this section, the WENO scheme is briefly explained. The main features of the WENO finite difference methods, finite volume methods, and the discontinuous Galerkin finite element methods for computational fluid dynamics can be found in Shu's paper [20]. The flux-averaged WENO method uses local Lax-Friedrichs flux-splitting and a characteristic decomposition of the variables and fluxes. The fluxes in x, y, and z are calculated separately. High order accurate and non-oscillatory scheme flux reconstruction uses a convex combination of k candidate stencils [8]. For $k = 3$, the fifth $(2k - 1)$ order finite difference WENO scheme approximates the derivative $\mathbf{F}(\mathbf{U})_x$ at a point x_i,

$$\mathbf{F}(\mathbf{U})_x|_{x=x_i} \approx \frac{1}{\Delta x}(\hat{F}_{i+1/2} - \hat{F}_{i-1/2}) \tag{5}$$

where $\mathbf{U}$ is the state vector, $\mathbf{F}(\mathbf{U})$ is the flux, and $\hat{F}_{i+1/2}$ and $\hat{F}_{i-1/2}$ are the right and left fluxes. The fifth order WENO scheme uses three stencils,

$$\hat{F}_{i+1/2} = \sum_{j=1}^{3} \omega_i \hat{F}^{(j)}_{i+1/2}$$

with three third order fluxes $\hat{F}^{(j)}_{i+1/2}$ and the nonlinear weights ω_i.

For hyperbolic conservation equations, the nonlinear part of WENO is carried out in local characteristics fields. The implementation starts by computing the average state $\bar{U}_{i+1/2,j,k}$ using the average, then the left and right eigenvectors and eigenvalues of the Jacobian $F'(\bar{U}_{i+1/2,j,k})$ at the average state. One can project the conservative fields and the fluxes onto the local characteristic fields using the left eigenvectors matrix and compute the left and right fluxes in characteristic field. Then, project back the numerical fluxes in the physical space using the right eigenvector matrix. We perform the same steps for the other two directions in y and z using the average states $\bar{U}_{i,j+1/2,k}$ and $\bar{U}_{i,j,k+1/2}$. In the next section, we will observe that WENO flux computation is the most time consuming in our simulations.

3 PERFORMANCE STUDIES

Our primary goal is to achieve performance improvements of the numerical simulations for hydrodynamic instabilities. We start with identifying the parts of the code that are time consuming. For this, we take advantage of the available performance analysis tools on Blue Waters. The first tool used for profiling is the Cray Performance and Analysis Tools (CPMAT). CPMAT is equipped with several components used in preparing any project for performance analysis. CPMAT is able to gather data during the program execution, and the data can be processed and analyzed for presentation to the user on its own graphical user interface, Cray Apprentice2. The details of how the code was prepared and how the analysis was carried out can be found in Blue Waters' user guide, but, in essence, any performance analysis process has three main steps: code instrumentation, execution, and data analysis. Apprentice2 is used for visualizing and exploring the data for analysis.

There are other profiling tools on the system, such as PAPI, PerfSuite, and TAU [15]. Among these profiling tools, we choose TAU, which is a comprehensive code profile tool with additional features

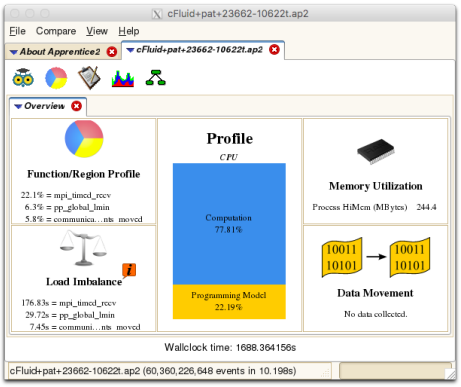

Figure 2: CPMAT Apprentice2 visualization.

for our performance analysis. In Section 3.3, the portable, robust, and parallel scalable TAU tool [18] is introduced for the performance instrumentation, measurement, analysis, and visualization.

The Blue Waters system is a Cray XE/XK hybrid machine composed of AMD 6276 "Interlagos" processors and NVIDIA GK110 (K20X) "Kepler" accelerators, all connected by the Cray Gemini torus interconnect [15]. The XE node has 2 Interlagos processors, and each processor has 16 bulldozer cores, as shown in Figure 3. Each bulldozer core's memory is 4GB, and the total node memory is 128GB. In the distributed memory parallel programming model with MPI, we observe that the memory footprint per integer core is enough to fit in memory, and we could use all 32 integer cores available on an XE node. We vary the number of MPI processes per node by setting the "-N" parameter in a job script file to investigate the effect of the processors per node (ppn) on the runtime. In Table 1, the time to solution on problem size $64 \times 64 \times 256$ with different processors per node is presented. The system default task placement for MPI processes is used in pure MPI runs. The efficiency per MPI process is virtually unaffected when changing the processors per node from 32 to 8 integer cores, and we observe a 6% loss of efficiency using 4 processors on 8 nodes. See Table 1.

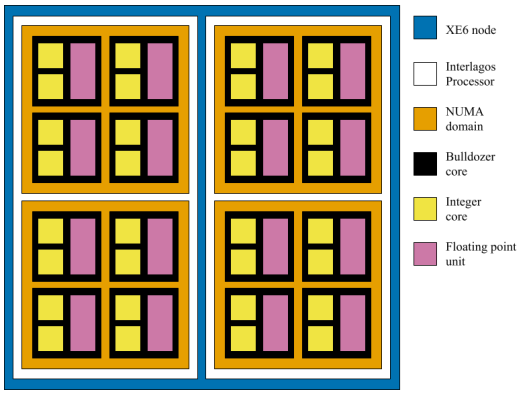

Figure 3: Cray XE6 node type on Blue Waters. Courtesy of Aaron Weeden, Blue Waters Petascale Institute notes.

Table 1: Time to solution on 32 MPI processes. #PPN is the processes per node.

#Node	#PPN	Time (seconds)
1	32	254
2	16	254
4	8	252
8	4	237

The runtime behavior is investigated by running weak and strong scaling analyses. The simulations are performed on a domain $1 \times 1 \times 4$ cm with a single grid spacing $\Delta x = \Delta y = \Delta z$. The number of grids in the z direction is four times the number of the grids in the x and y directions. The weak and strong scaling analyses are all performed using 32 processors per node in pure MPI jobs.

3.1 Weak Scaling

We conduct a weak scaling study and run simulations on four different grids: $64 \times 64 \times 256$, $128 \times 128 \times 512$, $256 \times 256 \times 1{,}024$, and $512 \times 512 \times 2{,}048$, using 32, 256, 2,048, and 16,384 cores, respectively, so that the amount of computation remains constant per core. The problem size on each MPI process is fixed and has 32 grid points in each direction. The runtimes for these problems are reported in Table 2, and they include both computation and communication. The explicit nature of the algorithm described in Section 2 contributes to the very good weak scaling as shown in Figure 4. The results for weak scaling indicate that the amount of communication increases 5% from 32 cores to 16,384 cores due to the communication overhead.

Table 2: Weak scaling for RT simulations. The grid resolution per MPI process is $32 \times 32 \times 32$.

Grid	#Processes	Actual Time to Solution	Ideal Time to Solution
$64 \times 64 \times 256$	32	254	254
$128 \times 128 \times 512$	256	263	254
$256 \times 256 \times 1{,}024$	2,048	266	254
$512 \times 512 \times 2{,}048$	16,384	269	254

3.2 Strong Scaling

To do a strong scaling analysis, we fix the total problem size while the resources are increased. The resolution of the computational grids $64 \times 64 \times 256$ (coarse) and $256 \times 256 \times 1{,}024$ (medium) run with a number of processes from 32 to 256 and from 1,024 to 8,192, respectively. Table 3 shows that the efficiency results drop to below 65% and 50% for the coarse and medium meshes. The simulations of Rayleigh-Taylor instability on the coarse and medium grid resolution are performed on processes with 2 threads (hybrid).

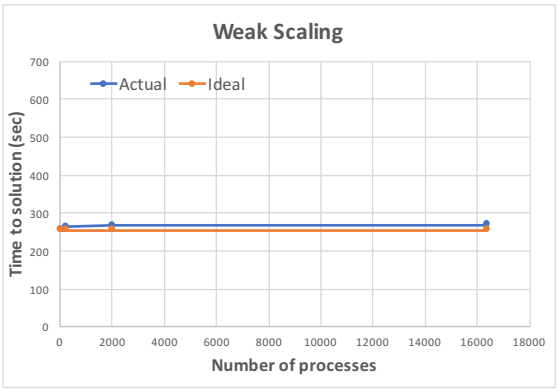

Figure 4: Runtimes under weak scaling.

Table 3: Strong scaling for RT simulations at grid resolutions of the coarse (C) and the medium (M) meshes.

#Processes		Actual Time to Solution		Ideal Time to Solution		Efficiency	
C	M	C	M	C	M	C	M
32	1,024	254	424	254	424	100%	100%
64	2,048	153	266	127	212	83%	80%
128	4,096	81	200	63.5	106	79%	53%
256	8,192	51	113	32.75	53	63%	47%

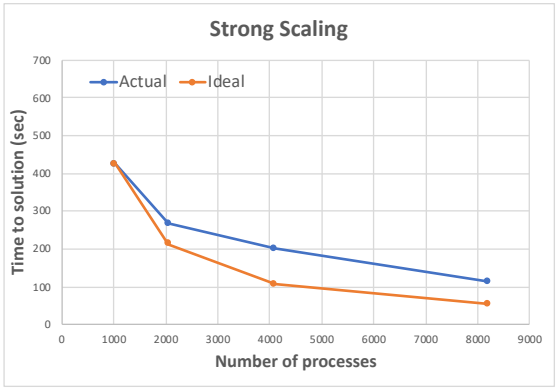

Figure 5: Runtimes under strong scaling for the medium mesh.

3.3 Profiling and Performance Analysis

In this section, we introduce the portable, robust, and parallel scalable TAU tool [18], which is used for performance instrumentation, measurement, analysis, and visualization. There are several options for instrumentation to observe the performance measurement, such as source-based, preprocessor-based, compiler-based, wrapper library-based, binary, interpreter-based, component-based, virtual machine-based, multi-level selective instrumentation, and TAU_COMPILER. The details of the instrumentation options are presented in [18].

Profiling consists of three stages: (i) compiling and linking the program with profiling enabled, (ii) executing the program to generate a profile data file, and (iii) running the profiler to analyze the data. Without any code modification, by compiling the program with a debug symbol ("-g "), a code developer can extract performance data measurements with minimal effort. TAU supports parallel profiling. Automatic instrumentation of the code using TAU's compilers (tau_cxx, tau_cc, tau_f90) and the visualization tool (ParaProf) help users to collect, analyze and visualize the performance data on thousands of processes. The TAU measurement system provides a profile data structure for each node/context/thread. Once compiling and building the executable with TAU compilers is done, we execute the new program to generate the profile data for each MPI process. The production of parallel profiles for thousands of processes requires an analysis tool to handle the performance information. TAU's scalable parallel performance profile analysis tool is called *ParaProf*. ParaProf provides a graphical interface to display all performance analysis results. ParaProf's 3D visualization option shows the spread of performance data across routines and processes. Figure 6 shows the profile data for the numerical simulation of Rayleigh-Taylor Instability. It helps in interpreting the performance data and presenting the time spent for each routine and process at once. TAU supports several types of performance profiles, such as flat, callpath, callsite, and phase profiles. The flat profile helps us to learn more about the time spent in an event, exclusive/inclusive, number of calls, and number of child calls. The profiling shows how much total time was spent in each routine. The exclusive and inclusive times show the statistics for each function. The exclusive time is the amount of time spent within that function, excluding the time spent in all child functions called from that function. The inclusive time is the amount of time spent within that function and all its child functions. The mean inclusive and exclusive times for a parallel test are presented in Figure 7. The bar graphs show the spread of performance data across routines on each process. In Figure 7, it is observed that the most time-consuming (blue bar graph, 34%) is the WENO flux computation. Each bar shows the mean exclusive time for routines and gives us an idea of how much time was spent in different routines. In the Comparison Window, Figure 8, we compare four coarse grid runs with a number of processes from 32 to 256. We have taken several steps to optimize the part of the code and work on the enhancement of the computational performance of the WENO flux computation in the `weno5_get_flux` routine.

4 HYBRID PARALLEL SCHEME

Code developers investigate the fastest programming model to handle computationally expensive simulations on clusters. An efficient programming model of the clusters of shared memory parallelism (SMP) is needed to handle large scale simulations. The parallel programming model could be purely MPI parallel or hybrid, depending on the application code. Hybrid programming with the MPI parallel scheme for internode communications and a shared memory programming for intranode communication is a preferable approach [6]. Available programming options for the shared memory parallelism are MPI-3, OpenMP, and OpenMP 4.0 / OpenACC for accelerator support. We investigate the usage of shared memory parallelism within the MPI processes for the best performance. Among the shared programming model options, we choose OpenMP for the intranode communication. OpenMP's application programming interface (API) supports the shared memory multiprocessing programming in C, C++, and Fortran. It is available since 1997 and is being actively developed to standardize directive-based, multi-language, high-level parallelism that is highly scalable and portable.

We first assess the performance of the code on the Blue Waters HPC platform using four different compilers: Cray, PGI, GNU, and Intel. The compiling and linking is performed using wrappers such as "ftn" for Fortran, "cc" for C, and "CC" for C++. To invoke the compiler, we set the programming environment corresponding to the specific compiler suite. Using the wrapper scripts and the compilers' default options provided on the system, a difference in times is observed. We observe that the PGI (Portland Group) compiler performed the best among all the compiler suites, as shown in Figure 4. There are many compiler options that can be used in the compilation process, and we use the default options that come with the wrappers. For the OpenMP directives and pragmas, "-h omp" for Cray, "-mp=nonuma" for PGI, and "-fopenmp" for GNU and Intel are provided additionally to the wrapper scripts for compiling the newly developed OpenMP implementation.

Table 4: Compiler performance on the application code.

#Processes	MPI Distribution	Time (seconds)			
		Cray	*PGI*	*GNU*	*Intel*
2,048	$8 \times 8 \times 32$	285	250	266	263
4,096	$8 \times 8 \times 64$	208	192	200	198
8,192	$16 \times 16 \times 32$	104	88	113	90

On Blue Waters, the maximum number of threads per node is 32. When running a hybrid (MPI+OpenMP) program, we first set the number of threads per process using the environment variable OMP_NUM_THREADS. In addition, the number of threads per process should be set using the depth parameter ("-d") in the run command. The depth parameter sets the number of OpenMP threads per MPI task, and it should have the exact same value as the environment variable OMP_NUM_THREADS. We specify the total number of MPI tasks for the job using the "-N" parameter, and the value for -N multiplied by the value for -d should not exceed 32 on Blue Waters.

The implementation of the WENO scheme described in Section 2 has a parallel region where we calculate the local eigenvalues, average state, right eigenvector matrix R_{mid}, and its left counterpart $L_{mid} = R^{-1}_{mid}$ at the mid-points. We transform the conservative fields, its differences, and flux differences to local characteristic field by multiplying them with L_{mid}, and we compute the numerical fluxes in each characteristic field. The last step is to project back the numerical fluxes in the physical space by multiplying them with the right eigenvector matrix R_{mid}. In this part of the code, the variables are scoped by using private and shared OpenMP data scope attribute clauses with the `parallel` directive.

In Figure 9, the time to solution using purely MPI and hybrid parallel schemes are compared. In the hybrid parallel scheme, the

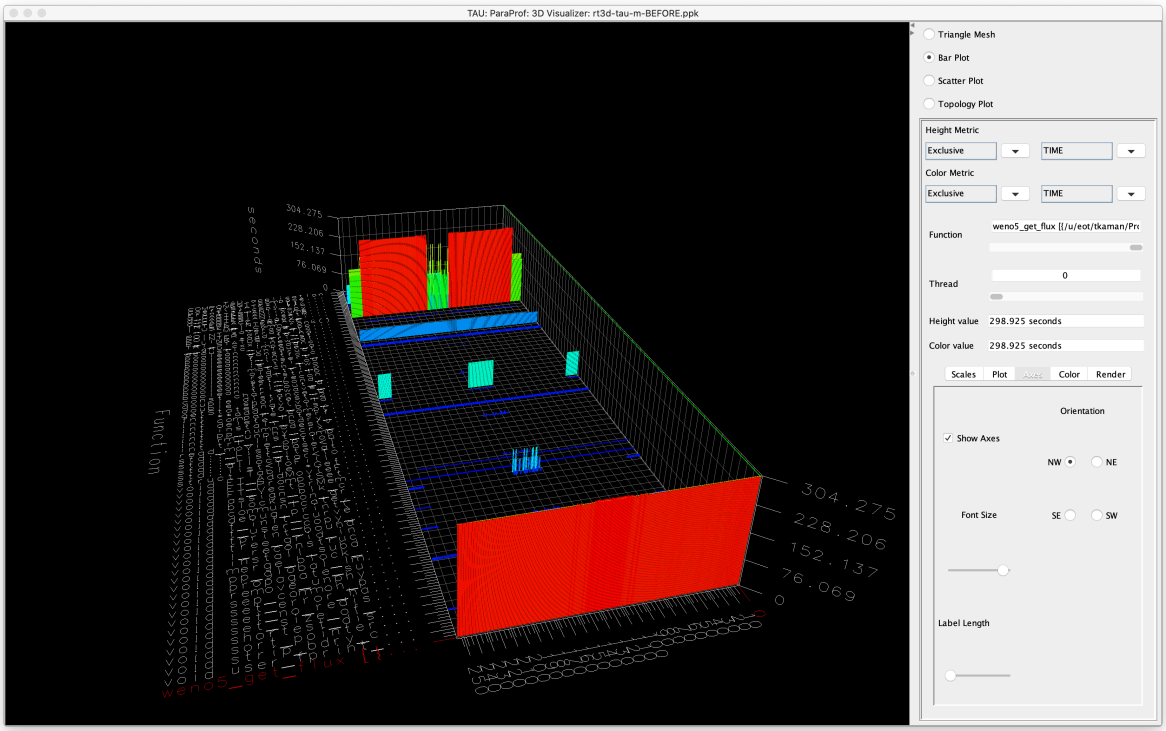

Figure 6: TAU's ParaProf 3D visualization shows the spread of performance data across routines and processes.

number of threads in OpenMP parallelism is set to two. The number of threads is controlled by setting the OMP_NUM_THREADS environment variable and giving the depth with "-d" option in running. For the hybrid case, the jobs run with 16 MPI processes per node and 2 OpenMP threads per process ("-N 16 -d 2 "). Figure 10 shows the performance improvement in the hybrid case with the inclusive time on the coarse grid. The mean exclusive time spent in the `weno5_get_flux` routine dropped from 302 seconds to 242 seconds, and the mean inclusive time is reduced by 20%. On the medium grid, we observe the pure MPI model is faster than the hybrid model. The newly developed version of the program with two OpenMP threads inside of each MPI task is slower than the pure MPI version as shown in Figure 11.

5 INTERNSHIP REFLECTION

The goal of the project is not only to investigate the effect of the parallel programming models on time to solution for the numerical simulations of compressible turbulent mixing, but also to engage the undergraduate students in petascale computing research in the area of computational fluid dynamics. The Blue Waters interns, Edwards and McGarigal, had little-to-no experience in Unix, programming in C, or parallel computing before starting this research project. A two-week intensive Petascale Institute at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign helped them to develop the basic skills needed to start this research. Within one year, they gained experience in the usage of Blue Waters, distributed/shared parallel programming models, visualization, and performance tools. During their internship program, they were selected to attend the International Conference for High Performance Computing, Networking, Storage, and Analysis as student volunteers. There, they were able to make many significant connections to help propel them into their future careers. Edwards and McGarigal presented their first poster at the American Physics Society Conference for Undergraduate Women in Physics, which was held at the Texas A&M University at Corpus Christi and at the 2019 Annual Meeting of the Arkansas Academy of Science (AAS), which was held at Fort Collins, respectively. Their poster at AAS received the first place undergraduate poster in computer science and was also selected to be presented at the 2019 Blue Waters Symposium.

When the mentor created two Blue Waters internship positions for the two University of Arkansas students who were interested in developing skills in modeling, simulations, and high performance computing, she planned a weekly schedule for her directed reading course. This course was a one-on-one independent study to cover the topics from hydrodynamics instabilities to parallel performance systems. The directed readings were designed to help the students to see the big picture, provide an overview of the state of the project, and guide them. The papers of Zhou [25, 26] on the basic properties of the flow, turbulence, and mixing induced by hydrodynamic instabilities, Sameer [16, 18] on a TAU user's guide, and Shu [19, 20] on numerical schemes were read throughout the first semester. In the second semester, the interns' main duties were

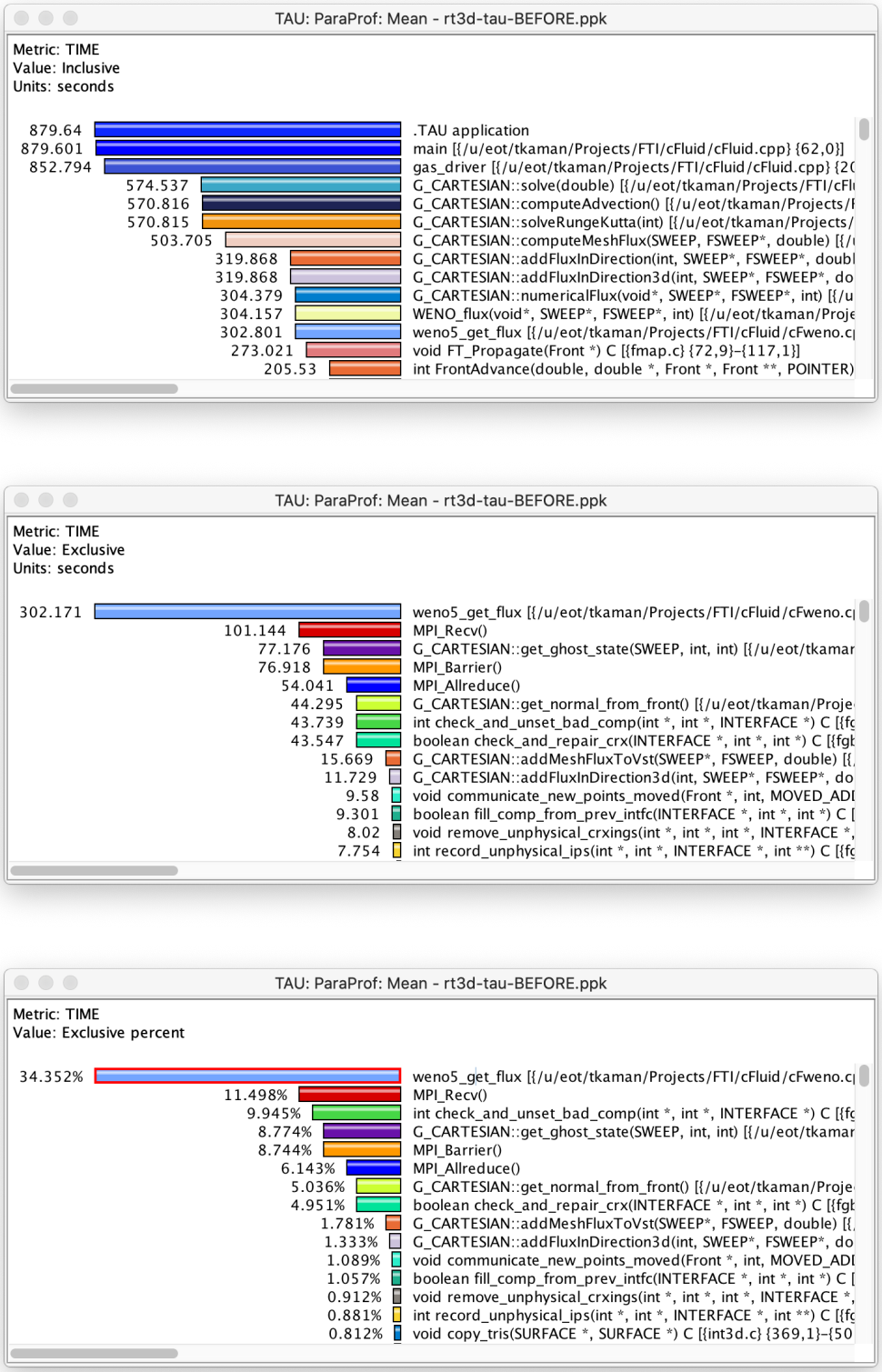

Figure 7: The mean inclusive time, exclusive time, and exclusive percent.

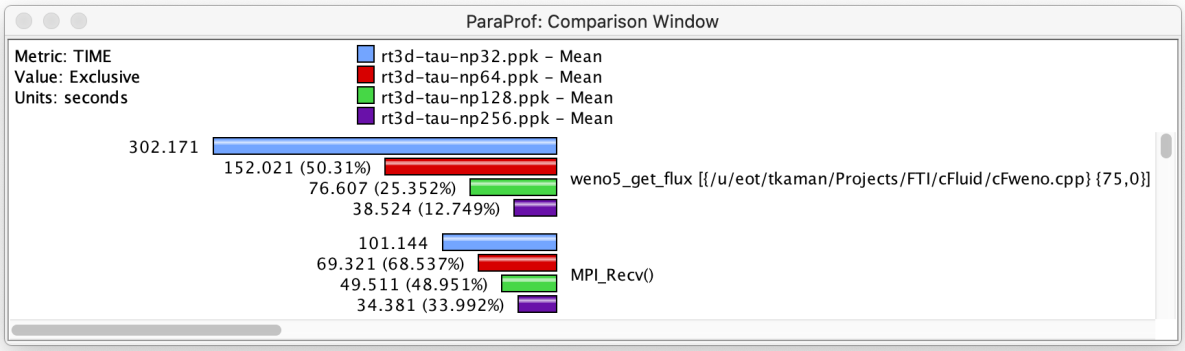

Figure 8: TAU's ParaProf Comparison Window shows the mean exclusive time.

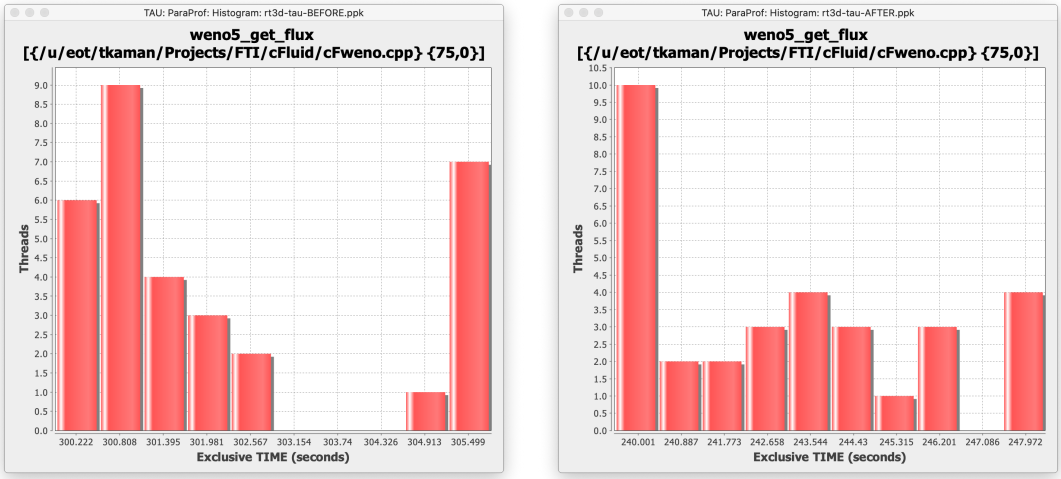

Figure 9: Comparing the performance of the `weno5_get_flux` routine using TAU's histogram function on the coarse grid.

to conduct simulations and analyze the results. To be eligible to complete the duties in a limited time, students must have met the requirements, such as being fluent in C/C++ programming, having experience with modeling and simulations, and having basic knowledge in parallel computing and computational fluid dynamics. The learning curve, the learner's performance on a task, and the number of attempts and time required for the task had taken more time than planned. In hindsight, it would have been better to add two levels of participation to the research program as learners and apprentices before their internships, as in the XSEDE EMPOWER (Expert Mentoring Producing Opportunities for Work, Education, and Research) program [17]. This way, the students would have first focused on strengthening their ability to handle challenges and taking steps on completing the assigned tasks. In a learner level, a student could have spend more time developing necessary skills to contribute to the work of Blue Waters through online tutorials, workshops, and self learning in programming. At the apprentice level, a student could have transformed the knowledge into skills and have the chance to apply the new skills with some additional trainings in debugging and performance tools to do the assigned tasks. After completing these two levels in two semesters, in their second year the students could have performed more independent work and became more fully engaged in research.

After their research experience in the computational and applied mathematics group of Kaman, the students pursue graduate studies and continue to work on computational science research projects. Edwards was one of the ten students accepted to the Oak Ridge National Laboratory Pathways to Computing Internship Program to learn and develop the next-generation explicit methods for radiation transport in astrophysics and explore programming models for GPUs supported on the fastest supercomputer in the world, Summit [22]. McGarigal started a new internship at HP as part of the test automation team, working on designing the robot framework for computers. The Blue Waters Student Internship Program helped

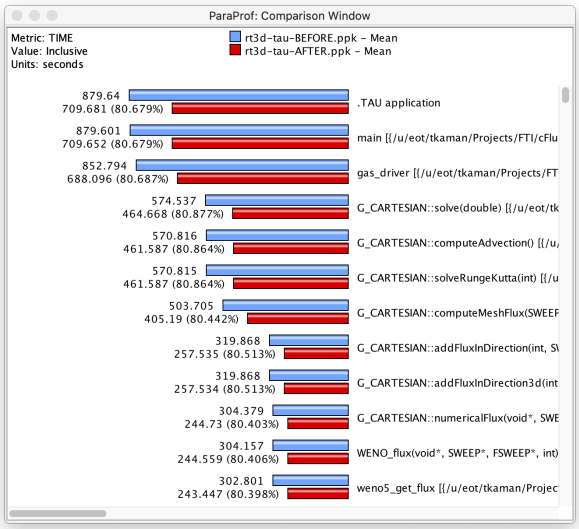

Figure 10: TAU's ParaProf Comparison Window shows the mean inclusive time before and after optimizing WENO flux on the coarse grid.

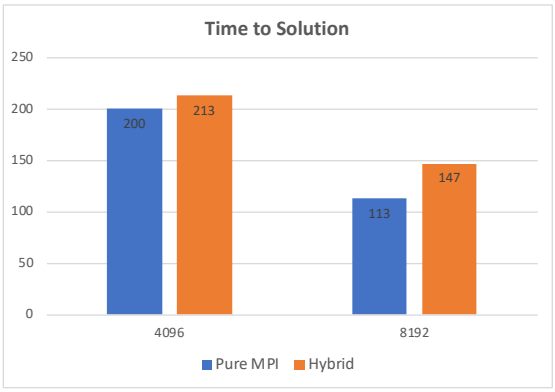

Figure 11: Comparison of purely MPI and hybrid models on the medium grid.

two University of Arkansas undergraduate students to develop strong computational skills in high performance computing and reflect their perspective of how they can advance their knowledge and skills for their future career.

6 CONCLUSIONS

Numerical simulations of turbulent mixing are computationally expensive and require efficient usage of high performance computing systems. The scalability of the purely MPI application code shows very good weak, and acceptable strong, scalability properties. We collect performance data to identify the most time consuming parts of the application code using the performance measurement and analysis tool TAU, whose performance system identifies that the high order accurate weighted essentially non-oscillatory numerical scheme is the computationally expensive part of the code. The flux computation starts with i) computing the average state, the left and right eigenvectors and eigenvalues of the Jacobian at the average state, ii) projecting the conservative fields and fluxes onto the local characteristic fields using the left eigenvectors matrix, iii) computing the left and right fluxes in characteristic field, and iv) projecting back the numerical fluxes in the physical space using the right eigenvector matrix. These computations are performed in a loop that is ideal for shared memory parallelism. In order to do that, we use the hybrid parallel model with MPI and OpenMP, where MPI is used for internode communication to pass states and interface data from one processor to another, and OpenMP is used for intranode communication to distribute the work equally to each thread. With the hybrid model, a performance improvement on the coarse grid is observed, and the total time to solution is reduced by 20%. However, the pure MPI implementation shows the best scalability on the medium grid on Blue Waters. The good weak and strong scalability of the pure MPI model is because of the optimized work distribution between processes. The problems with OpenMP performance could be due to the memory access and cache use. The use of "numactl", the core layout, plays an important role to achieve scalability. To avoid the bottlenecks with memory and cache, the task placement to distribute MPI processes and threads per processes will be investigated in the future.

ACKNOWLEDGMENTS

This work was supported by a grant from the Shodor Education Foundation through the Blue Waters Student Internship Program. This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

REFERENCES

[1] W. Bo, X. Liu, J. Glimm, and X. Li. 2011. A robust front tracking method: Verification and application to simulation of the primary breakup of a liquid jet. *SIAM J. Sci. Comput.* 33 (2011), 1505–1524.

[2] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot. 1991. A dynamic subgrid scale eddy viscosity model. *Phys. Fluids A* 3 (1991), 1760–1765.

[3] J. Glimm, B. Cheng, D. H. Sharp, and T. Kaman. 2020. A crisis for the verification and validation of turbulence simulations. *Physica D: Nonlinear Phenomena* 404 (2020), 132346. https://doi.org/10.1016/j.physd.2020.132346

[4] J. Glimm, H. Kirk, X. L. Li, J. Pinezich, R. Samulyak, and N. Simos. 2000. Simulation of 3D fluid jets with application to the Muon Collider target design. In *Advances in Fluid Mechanics III*, M. Rahman and C. A. Brebbia (Eds.). Vol. 26. WIT Press, Southampton, Boston, 191–200.

[5] J. Glimm, D. H. Sharp, T. Kaman, and H. Lim. 2013. New Directions for Rayleigh-Taylor Mixing. *Phil. Trans. R. Soc. A* 371 (2013), 20120183. https://doi.org/10.1098/rsta.2012.0183 Los Alamos National Laboratory Preprint LA-UR 11-00423 and Stony Brook University Preprint SUNYSB-AMS-11-01.

[6] Torsten Hoefler, James Dinan, Darius Buntinas, Pavan Balaji, Brian Barrett, Ron Brightwell, William D Gropp, Laxmikant V Kale, and Rajeev Thakur. 2013. MPI + MPI: A new hybrid approach to parallel programming with MPI plus shared memory. *Computing (Vienna/New York)* 95, 12 (2013), 1121–1136. https://doi.org/10.1007/s00607-013-0324-2

[7] Ya-Ting Huang and J. Glimm. 2017. A Novel methodology of stochastic short term forecasting of cloud boundaries. *J. Uncertainty Quantification* 5 (2017), 1279–1294.

[8] G. Jiang and C.-W. Shu. 1996. Efficient Implementation of Weighted ENO schemes. *J. Comput. Phys.* 126 (1996), 202–228.

[9] T. Kaman. 2019. Model calibration for turbulent mixing simulations. In *The 16th International Workshop on the Physics of Compressible Turbulent Mixing*, I Houas G. Jourdan and C. Mariani (Eds.). IUSTI UMR 7343, 129–132.

[10] T. Kaman, J. Glimm, and D. H. Sharp. 2010. Initial Conditions for Turbulent Mixing Simulations. *Condensed Matter Physics* 13 (2010), 43401. Stony Brook University Preprint number SUNYSB-AMS-10-03 and Los Alamos National Laboratory Preprint number LA-UR 10-03424.

[11] T. Kaman, R Kaufman, J. Glimm, and D Sharp. 2012. Uncertainty Quantification for Turbulent Mixing Flows: Rayleigh-Taylor Instability. *IFIP Advances in Information and Communication Technology* 377 (01 2012). https://doi.org/10.1007/978-3-642-32677-6_14

[12] T. Kaman, H. Lim, Y. Yu, D. Wang, Y. Hu, J.-D. Kim, Y. Li, L. Wu, J. Glimm, X. Jiao, X.-L. Li, and R. Samulyak. 2011. A Numerical Method for the Simulation of Turbulent Mixing and its Basis in Mathematical Theory. In *Lecture Notes on Numerical Methods for Hyperbolic Equations: Theory and Applications: Short Course Book*. CRC/Balkema, London, 105–129. Stony Brook University Preprint SUNYSB-AMS-11-02.

[13] R. Kaufman, H. Lim, and J. Glimm. 2016. Conservative front tracking: the algorithm, the rationale and the API. *Bulletin of the Institute of Mathematics, Academia Sinica New Series* 11 (2016), 115–130. Stony Brook University Preprint SUNYSB-AMS-15-01.

[14] Parviz Moin and Krishnan Mahesh. 1998. DIRECT NUMERICAL SIMULATION: A Tool in Turbulence Research. *Annual Review of Fluid Mechanics* 30, 1 (1998), 539–578. https://doi.org/10.1146/annurev.fluid.30.1.539 arXiv:https://doi.org/10.1146/annurev.fluid.30.1.539

[15] University of Illinois at Urbana-Champaign National Center for Supercomputing Applications. 2019. Blue Waters Blue Waters. (2019). https://bluewaters.ncsa.illinois.edu

[16] Department of Computer and University of Oregon Advanced Computing Laboratory Information Science. 2020. TAU User Guide. (2020). https://www.cs.uoregon.edu/research/tau/docs/newguide/bk01.html

[17] Extreme Science and Engineering Discovery Environment. 2020. XSEDE. (2020). http://computationalscience.org/xsede-empower

[18] Sameer S. Shende and Allen D. Malony. 2006. The Tau Parallel Performance System. *The International Journal of High Performance Computing Applications* 20, 2 (2006), 287–311. https://doi.org/10.1177/1094342006064482 arXiv:https://doi.org/10.1177/1094342006064482

[19] Chi-Wang Shu. 1998. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, A. Quarteroni (Ed.). Lecture Notes in Mathematics, Vol. 1697. Springer, 325–432.

[20] Chi-Wang Shu. 2003. High-order Finite Difference and Finite Volume WENO Schemes and Discontinuous Galerkin Methods for CFD. *International Journal of Computational Fluid Dynamics* 17, 2 (2003), 107–118. https://doi.org/10.1080/1061856031000104851 arXiv:https://doi.org/10.1080/1061856031000104851

[21] J. Smagorinsky. 1963. General Circulation Experiments with the Primitive Equations. *Mon. Weather Rev.* 91 (1963), 99–165.

[22] TOP500.org. 2019. TOP 500 list. (2019).

[23] Shuai Xue. 2015. *A Sharp Boundary Model for Electrocardiac Simulations*. Ph.D. Dissertation. State Univ. of New York at Stony Brook.

[24] H. Zhang, T. Kaman, D. She, B. Cheng, J. Glimm, and D. H. Sharp. 2018. V&V for turbulent mixing in the intermediate asymptotic regime. *Pure and Applied Mathematics Quarterly* 14 (2018), 193–222. Los Alamos National Laboratory preprint LA-UR-18-22134.

[25] Y. Zhou. 2017. Rayleigh-Taylor and Richtmyer-Meshkov instability induced flow, turbulence, and mixing I. *Physics Reports* 720–722 (2017), 1–136.

[26] Y. Zhou. 2017. Rayleigh-Taylor and Richtmyer-Meshkov instability induced flow, turbulence, and mixing II. *Physics Reports* 723–725 (2017), 1–160.