Experiences from Leadership Computing in Simulations of Turbulent Fluid Flows

Myoungkyu Lee, Rhys Ulerich, Nicholas Malaya, and Robert D. Moser | University of Texas at Austin

PoongBack is a turbulence simulation code that helps create realistic simulations of turbulent flows. PoongBack includes a new parallel 3D FFT kernel and shows excellent scalability up to 786,432 cores. Using Mira at the Argonne Leadership Computing Facility, PoongBack achieved direct numerical simulation at $Re_\tau = 5{,}200$, and generated approximately 140 Tbytes of data.

Engineering efforts to reduce turbulent drag and thus energy consumption are hindered by the poor predictive capabilities of state-of-the-art turbulence models. Improving these models' effectiveness requires advancing our fundamental understanding of wall-bounded turbulence. Direct numerical simulation (DNS) provide the necessary insights for model improvement, and they open new possibilities for manipulating turbulent phenomena. Unfortunately, the utility of DNS is limited by its extreme expense with memory and compute requirements scaling like the third and fourth power of the nondimensional Reynolds number, *Re*, respectively. Most turbulent flows of engineering interest possess *Re* high enough to require supercomputing.

The significant algorithmic and numerical efforts required to enable massively parallel DNS are well documented.[1–3] Further, the process of running the simulation requires substantial effort, over many months. Here, we document our experiences preparing and running a leadership computing simulation, and detail various aspects of the workflow. The majority of the benchmark data presented here was measured on Mira, an IBM BlueGene/Q system at Argonne National Laboratory, on which the production run was conducted.

Petascale Performance

To simulate the flow of an incompressible fluid governed by the Navier-Stokes equations, we developed a new turbulence simulation code, *PoongBack*. The flow between two infinite parallel planes is simulated by imposing periodic boundary conditions on a finite domain in the streamwise (x) and spanwise (z) directions. Also, zero-velocity boundary conditions are imposed at the walls. We used the well-known formulation of John Kim and his colleagues[4] to solve the Navier-Stokes equations in the channel geometry. The full Navier-Stokes equations with continuity can be re-expressed as follows:

$$\frac{\partial \omega_y}{\partial t} = h_g + \nu \nabla^2 \omega_y, \quad \frac{\partial \phi}{\partial t} = h_v + \nu \nabla^2 \phi, \tag{1}$$

where v is the kinematic viscosity, h_g and h_v are combinations of derivatives of the nonlinear terms,[4] ω_y is the vorticity component in the wall-normal (y) direction, and ϕ is the Laplacian of velocity component in the y direction. We used a Fourier-Galerkin method in the x and z directions, so the aforementioned equations become a set of time-dependent 1D partial differential equations (PDEs) in Fourier space. We used a B-spline collocation method to evaluate derivatives in the wall-normal direction. The nonlinear terms, h_g and h_v, include quadratic products of velocity components. Evaluating quadratic products in Fourier space is extremely expensive, so they're computed in physical space instead. This process requires a number of forward and inverse Fourier transforms on the entire dataset with global data transposes. For time advancement, we used a third-order low-storage Runge-Kutta scheme to treat the nonlinear terms explicitly, while the linear terms are treated

 1521-9615/14/$31.00 © 2014 IEEE Copublished by the IEEE CS and the AIP

implicitly with a Crank-Nicholson-type scheme. As a result, a number of linear systems need to be solved for each time step. In summary, the code is composed of three major parts: global data transpose, fast Fourier transform (FFT), and solving Equation 1 in Fourier space.

For data distribution, we used 2D partitioning, the so-called pencil decomposition, as Figure 1 shows. In Figure 1, small blocks depict the smallest data unit, and large blocks with thick boundaries depict each message passing interface (MPI) task. The operations, such as FFT and solving linear equations, perform best when all required data are in the same MPI task. In our code, FFTs are performed in the x and z directions and linear equations are solved in the y direction. As a result, four global data communications need to be performed, x-pencil to z-pencil, z-pencil to y-pencil, and vice versa. The communication pattern requires two MPI subcommunicators, which are created with `MPI_cart_create` and `MPI_cart_sub` functions. For each communication stage, the data transpose can be performed with `MPI_alltoall` calls. However, `MPI_alltoall` doesn't always yield the best communication performance. We created MPI subcommunicators with Cartesian topology. On Mira, when MPI tasks for one subcommunicator are topologically close, MPI tasks for the other subcommunicator are topologically distant. It's analogous to operating with rows or columns in a matrix with continuous data storage. Because of this, `MPI_sendrecv` or other MPI communication tools may perform better than `MPI_alltoall` for the subcommunicator with distantly located MPI tasks. For these reasons, it is beneficial to use the global communication library in FFT in the West (FFTW) 3.3, because it will select among several different MPI call sequences by measuring their performance on the target problem during an initiation phase. The FFTW 3.3 communication library actually only supports a 1D data decomposition, the so-called planar decomposition. We used it for our 2D decomposition by applying it to each communicator separately.

If someone were to choose to use a planar decomposition instead of the pencil decomposition used here, the number of global data communications would be reduced by half. However, the maximum number of MPI tasks in the planar decomposition is bounded by the grid size in one direction. For example, the maximum number of MPI tasks is 1,024 when the grid size is $1{,}024^3$. Also, it's more difficult to achieve good data load balance among MPI tasks with planar decomposition. Hence, we chose to use a pencil decomposition for flexibility and adaptability in currently available leadership computing systems.

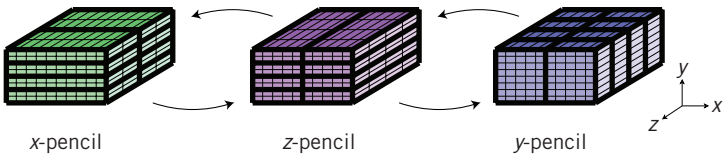

Figure 1. Pencil decomposition and transpose. Small blocks depict the smallest data unit, and large blocks with thick boundaries depict each message passing interface (MPI) task.

In the mixed implicit-explicit time advance scheme used here, a number of linear systems, $\boldsymbol{A}\mathbf{x} = \mathbf{b}$, need to be solved in Fourier space. With the B-spline representation, $\boldsymbol{A}$ is a banded matrix with additional nonzero entries in the first and last few rows. Also, $\boldsymbol{A}$ is real while $\mathbf{x}$ and $\mathbf{b}$ are complex vectors, since the equations are formulated in Fourier space. Such systems can be solved using the conventional Linear Algebra Package (LAPACK) library, but it isn't optimal. It would require data reshuffling and, because of the additional nonzero elements in the first and last few rows, many unnecessary computations would be performed. Instead, we developed a customized linear algebra solver to treat this system's specific structure. Our benchmark data shows that the customized linear algebra solver is four times faster than LAPACK implementations, such as those in Intel Math Kernel Library (MKL) and IBM Engineering and Scientific Subroutine Library (ESSL).

We also implemented hybrid parallelism using OpenMP threads. Linear algebra for time advancement and the FFTs only requires one contiguous data line, which is independent of other data lines. This makes threaded parallelism straightforward to implement in OpenMP. Our benchmark results on Mira, which have 16 physical cores per node, show that parallel efficiency of OpenMP is almost 100 percent, with up to 16 OpenMP threads. Additionally, performance is more than doubled using four hardware threads per core. We benchmarked performance of the 3D FFT kernel at the heart of the code against Parallel Three-Dimensional FFT (P3DFFT)-2.5.1, which is one of most widely used parallel 3D FFT libraries. The parallel 3D FFT is the combination of two global data transpose and three FFTs in each direction. In this benchmark, we disabled the last FFT in the y direction in P3DFFT, because the linear equation in Fourier

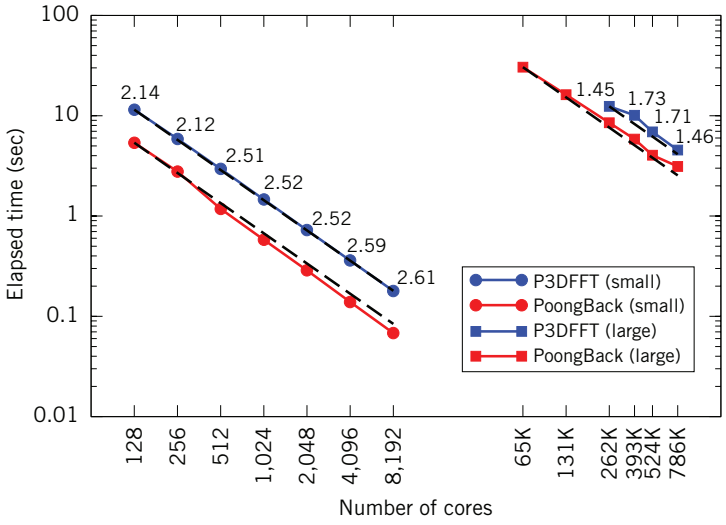

Figure 2. Parallel fast Fourier transform (FFT) performance. Small is 2,048 × $1{,}024^2$; large is 18,432 × $12{,}288^2$. The numbers shown on the blue points are the ratio of Parallel Three-Dimensional FFT (P3DFFT) execution times to that of PoongBack.

space is solved in *y* direction in our simulation instead of an FFT. Figure 2 shows the benchmark result. The new parallel FFT routine performs better than P3DFFT in all benchmark cases.

Figure 3 shows overall code performance for a full time step on Mira. Two cases are tested; 4 OpenMP threads per MPI task and 16 MPI tasks per node; 64 OpenMP threads per MPI task and 1 MPI task per node. Sixteen MPI tasks per node result in almost perfect strong scaling up to 786,432 cores. However, 1 MPI task per node has better performance, though it doesn't scale quite as well. The weak scaling benchmarks are similar, in that more MPI tasks result in better scaling but poorer performance. This is mainly because of communication contention that occurs when using many MPI tasks. This implies that reducing the total number of MPI tasks by using more OpenMP threads can be beneficial, provided that OpenMP threading scales well. It is well-known that the FFT computations don't scale linearly. As a result, in the weak scaling results, the parallel efficiency decreases with increasing core count because the corresponding data size also increases. Such performance decreases become worse when the data size for one FFT overflows cache memory.

Petascale I/O

The ExaScale I/O (ESIO) library provides simple, high-throughput I/O of turbulence simulation restart files using parallel HDF5.[5] The ESIO library was developed after an earlier study concluded that the increased flexibility HDF5 provides outweighed its slight performance penalty on petascale systems.[6] In addition to easing near-term tasks, including post-processing and visualization, using HDF5 for long-lived, public datasets increases their long-term value to the turbulence research community.[7] In addition to lowering the barrier for DNS practitioners to use HDF5, ESIO is designed to encourage current best practices in turbulence simulation data management.

Our turbulence research group has used ESIO as the I/O subsystem for several Fortran and C++ codes, including for a homogeneous, isotropic turbulence code based on *PSDNS*;[8] for channel flow codes like PoongBack; and most recently for reacting, multispecies boundary-layer simulations.[9] Consolidating I/O handling in ESIO eases and standardizes I/O management in each of these applications. For example, PoongBack's restart handling and in situ analysis outputs comprise less than 4 percent of its source code. The library builds on a variety of leadership class systems including Mira, Stampede, Titan, and Blue Waters. Open development is ongoing at https://github.com/RhysU/ESIO.

Table 1 shows the strong scaling benchmarks for ESIO, mimicking previous parallel FFT benchmarks[1] and selecting hybrid parallelism with 1 rank per 16 cores (that is, per node). Each read and write benchmark was divided into four distinct phases—opening a file, processing PoongBack-like metadata consisting of time advance and linear operator details, writing simulation state, and finally closing the file. The file open and metadata phases don't scale, but their wall time is small enough to be of no concern. The file-opening phase within the Table 1 write benchmarks was curiously slow and didn't scale with the number of MPI ranks used. It did, however, show a linear dependence on the problem sizes chosen (5.06 versus 1.76 Tbytes). After investigation, the root cause was identified to be the benchmark creating a "new" file on each iteration by repeatedly opening the same path in overwrite mode. In this circumstance, the ESIO implementation supplied the `H5F_ACC_TRUNK` flag when invoking the HDF5 `H5Fcreate` method, causing the clobbered file to be truncated at approximately 60 Gbytes/s. Writing to a wholly new path showed no such delay. By first attempting to unlink a path on overwrite, future versions of ESIO should be more performant on this use case.

Further improvements in ESIO performance are possible and will be pursued as needed.

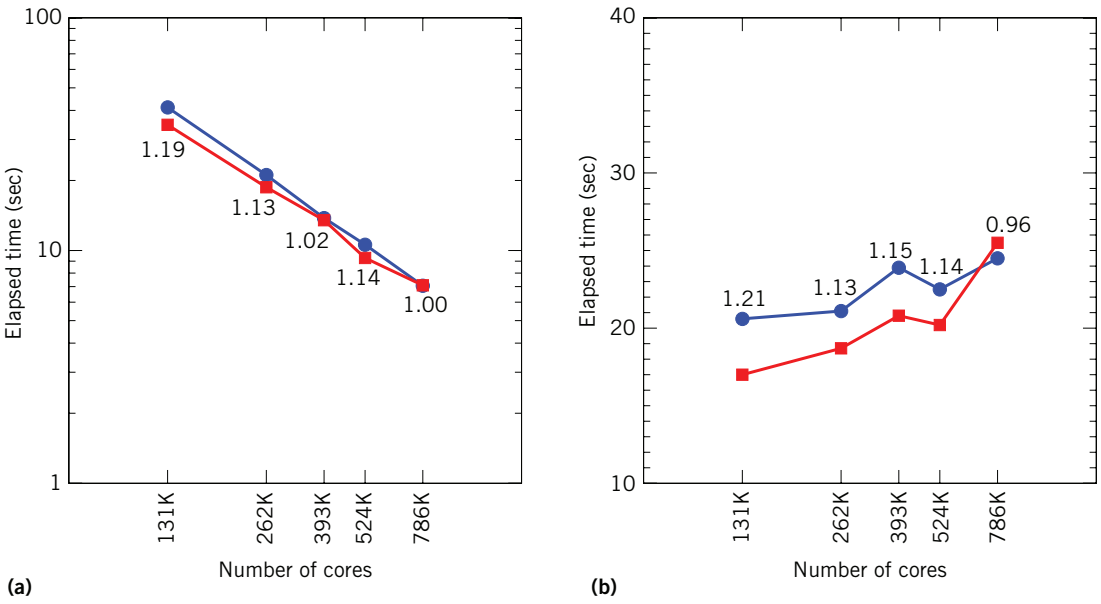

Figure 3. Overall code performance. Blue: 16 MPI tasks/node; Red: 1 MPI task/node. Numbers shown by the data points are the ratios of execution time for 16 MPI tasks/node to that for 1 MPI task/node.

Table 1. ExaScale I/O (ESIO) read and write time measured in seconds on Mira using one MPI process per 16 cores.*

Cores	MPI ranks	File open	Read meta	Read state	File close	Total read	File open	Write meta	Write state	File close	Total write
32,768	2,048	9.52	6.99	282.25	0.83	299.6	94.2	0.92	647.8	3.66	746.58
65,536	4,096	11.63	6.56	155.37	1.33	174.89	95.31	1.39	730.8	3.6	831.1
131,072	8,192	10.16	7.04	110.38	1.99	129.57	96.21	2.27	741.39	3.99	843.86
262,144	16,384	11.6	7.57	52.59	2.48	74.24	95.14	3.91	439.15	5.21	543.42
524,288	32,768	11.2	8.38	59.71	4.23	83.52	96.55	6.88	476.51	10.18	590.12
524,288	32,768	14.76	7.88	25.22	2.81	50.67	35.43	6.86	84.57	13.43	140.29

*Two 18,432 × 1,536 × 12,288 state fields were processed for a total of 5.06 Tbytes. Observed standard deviations in read and write time across 3–5 samples were 1.8–12.5 percent and 2.5–30.5 percent, respectively. The highlighted row benchmarks two smaller 10,240 × 1,536 × 7,680 fields totaling 1.76 Tbytes, which correspond to production PoongBack jobs.

However, the current performance has been adequate for production PoongBack work. The final line of Table 1 is representative of the $Re_\tau = 5{,}200$ channel flow. At that problem size, reading one restart file on job startup and defensively checkpointing simulation state five times in a 12-hour run requires less than 2 percent of the wall-clock time.

Code Verification

The PoongBack code underwent comprehensive testing and verification before initiation of production simulations. Furthermore, when an unexpected anomaly was observed in the simulation results, an even more comprehensive and detailed set of verification tests were conducted to confirm that it wasn't a consequence of a software error. One of the most powerful verification tools employed for the verification of PoongBack is the so-called method of manufactured solutions. In this approach, a "solution" to the equations being solved is posited, and a forcing term, which is just the residual in the equations when evaluated for the manufactured solution, is added to the equations. This makes the manufactured solution a

solution to the modified equations. The code is then used to solve the forced equations, and the numerical solution compared to the exact manufactured solution.

One of the challenges to using the method of manufactured solutions is reliably computing the required forcing term. This has been made straightforward by the Manufactured Analytical Solution Abstraction (MASA) library,[10] which determines the forcing term from a specification of the manufactured solution and the equations being solved, using automatic differentiation. The other challenge to verification with manufactured solutions is the definition of solutions that will provide a strong test of the code. A family of manufactured solutions for the incompressible Navier-Stokes equations in rectangular domains was developed for this purpose, and is described here. The first requirement for an incompressible manufactured solution is that it satisfies the continuity constraint; that is, that its divergence be zero. To this end, the velocity components u, v, and w, in the x, y, and z directions, respectively, are defined as:

$$u = af(x)g'(y)h'(z)$$
$$v = bf'(x)g(y)h'(z)$$
$$w = cf'(x)g'(y)h(z),$$

where f, g, and h are arbitrary functions having many continuous derivatives and satisfying required boundary conditions (as described below), and prime denotes the derivative. The coefficients a, b, and c can in general be functions of time. The divergence of the velocity will be zero, provided $a + b + c = 0$. For the channel flow simulations discussed here, f and h must satisfy periodic boundary conditions, which is accomplished by defining them in terms of sines and cosines. The function g and its first derivative must be zero at the walls at $y = \pm 1$, which is ensured by defining

$$g(y) = (1 - y^2)^2 \tilde{g}(y).$$

In precisely specifying the functions f, g, and h, and the coefficients, it's useful to choose functions that can be represented exactly by the numerical formulation in the code, and to choose functions that can't be exactly represented. In PoongBack, Fourier representation in x and z and B-splines in y are used. Exactly representable functions in x and z are therefore just sums of Fourier functions that are included in the expansion. To construct functions that aren't exactly represented, we use functions of the form

$$f(x) = \frac{1}{\beta_x + \sin(k_x x)}, \quad h(z) = \frac{1}{\beta_z + \sin(k_z z)},$$

where $\beta_x > 1$ and $\beta_z > 1$, and where clearly cosine could be used in place of sine. In the y-direction it's easiest to define

$$\tilde{g}(y) = P_n(y),$$

where P_n is a polynomial of degree n. Because B-splines of degree 7 are used in the production channel simulations, if $n \leq 3$ the $g(y)$ will be exactly represented, while if $n > 3$ it won't. Finally, since the time discretization scheme is formally second order, choosing the coefficients to be polynomials in time of degree 2 or less will result in an exact representation in time, while higher degrees won't be exact.

Using manufactured solutions of these forms, a comprehensive set of verification tests were conducted. First, solutions which are exactly represented by the numerics were used to confirm that the exact solutions were recovered, providing a basic check that the correct equations are being solved. Second, solutions that are approximately represented in only one of the spatial directions or time were used, and the rate of convergence with refinement in the approximated direction was measured to confirm that it's consistent with the theoretical convergence rate. Finally, solutions that aren't exactly represented in any direction or time were used to test the full numerical representation in the code. Having passed these and a wide variety of other more mundane verification tests, we have extremely high confidence that PoongBack correctly solves the incompressible Navier-Stokes equations, as designed.

Running the Simulation

Our simulation of channel flow at $Re_\tau = 5{,}200$ has a total of 242 billion degrees of freedom. This is significantly larger (by a factor of 15) than the previously largest channel simulation conducted by Sergio Hoyas and Javier Jiménez[3] in 2006 at $Re_\tau = 2{,}003$. This particular simulation was run on 524,288 cores of Mira, which is 67 percent of the entire Mira system. Each job consumes approximately 6 million core hours for a 12-hour run.

The statistical sampling period for DNS of turbulent channel flows are typically measured in

flowthrough times, where a flowthrough is the average time required for a fluid particle to traverse the simulation domain (in x). Previous DNS of channels of similar size have been performed for ten flowthrough times. Simulations were initialized with a turbulence dataset from a $Re_\tau = 2{,}003$ simulation, and flow parameters were adjusted to increase the Reynolds number. After simulation with coarse resolution for approximately two flowthroughs, we refined the grid by a factor of 2.5. A series of three such refinements were made until the final production resolution was reached. On Mira, a flowthrough time at full resolution in our simulation costs approximately 20 million core hours.

A DNS is essentially one long simulation of a turbulent flow, broken up into many smaller runs that execute for the maximum permitted time for a queued job, generally between 12 and 24 hours. The entire simulation is thus conducted over a period of many months (about nine months for the channel flow simulation discussed here). Managing and monitoring the simulation over such a long execution time is challenging. Although each job generally doesn't need to be monitored while it's running, it's important to monitor the results of jobs as they complete by computing a variety of statistical quantities. This allows any departures from the expected behavior that might have been caused by hardware faults or changes in the software stack to be detected.

Recent advances in both computing capabilities and experimental techniques have made it possible to generate scientific datasets of unprecedented size at an unprecedented rate. Because of these advances, 90 percent of the world's data was created in the last two years.[11] DNS is no exception to this trend, with the turbulent simulations performed here generating hundreds of petabytes of data.

This has generated a substantial data management requirement for this project. We stored about 75 snapshots of the velocity fields, and the size of each snapshot is 1.8 Tbytes. It's infeasible to maintain all the generated fields on disk, and as a result, they're all archived to tape using the High-Performance Storage System (HPSS) at Argonne National Laboratory. Postprocessing requires moving each file from tape to disk, performing analysis, and then moving to the next file. Sophisticated scripts to manage this process were developed. Also, for the security against possible loss of the data, and for accessibility by other researchers, the entire dataset was mirrored to another tape archival system, Ranch, at the Texas Advanced Computing Center at the University of Texas. The data transfer, including a checksum test via *Globus Online*, usually took one day for each 1.8-Tbyte snapshot.

Currently, the most complete sequence of cases for turbulent channels is in the friction Reynolds number range $Re_\tau = 180 - 2{,}000$, which now constitutes the standard reference data set in the field. Supplementing these data with $Re_\tau = 5{,}200$ will establish a reference dataset that will remain useful for the turbulence research community for many years. It's expected that this dataset will need to be accessed by us and the fluid dynamics community for several years. Generally, the community is provided access to statistical files and the raw field data.

For these turbulent fields to remain useful, it's important that each file be self-sufficient and self-documenting. To ensure this, PoongBack stores a plethora of metadata information, such as

- the physical problem definition, including Reynolds Number, viscosity, and pressure gradient;
- grid information such as the box size $\{L_x, L_y, L_z\}$ and number of points $\{N_x, N_y, N_z\}$;
- linear operators to compute derivatives {B-spline knots, matrices, and so on}; and
- instructions for how to read and manipulate the data.

The metadata is sufficiently rich to allow the simulation state to be completely reconstructed, with no additional outside information. Such rich metadata simplifies collaboration. It also naturally conforms to US National Science Foundation requirements for data management (see www.nsf.gov/eng/general/dmp.jsp).

We performed the largest direct numerical simulation of wall-bounded turbulent flows using Mira at Argonne National Lab. The results from this simulation, which aren't presented here, will be used to expand our knowledge of wall-bounded turbulence. This simulation wouldn't have been possible without petascale computing resources. Advances in direct numerical simulation of wall-bounded turbulence have followed the evolution of leadership computing systems. Hence, it's natural to look forward to the advent of exascale systems, which would allow simulations at a Reynolds number that's about six times larger.

In addition to excellent performance of our new code, PoongBack, the benchmark results showed several obstacles on the road to exascale DNS of wall-bounded turbulence. First, single-core performance was only 9 percent of theoretical peak, even though more than 98 percent of operations used data already in L1 cache. This was primarily because the double data rate (DDR) traffic was saturated. Second, there was communication contention when there were a very large number of MPI tasks. The communication contention can be alleviated somewhat by reducing the number of MPI tasks using hybrid parallelism. But this just puts off the problem. As node counts continue to increase, so will numbers of MPI tasks and the potential for network contention in applications similar to ours. In summary, at least for our simulation, memory bandwidth and global communication performance will be much more important than the theoretical maximum flop rate as systems evolve toward exascale. ■

Acknowledgments

The work described here was enabled by a generous grant of computer time, Early Science Program, INCITE 2013, and the Director's Discretionary Program from Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the US Department of Energy under contract DE-AC02-06CH11357. The financial support of the US National Science Foundation under PetaApps grant OCI-0749223 and PRAC grant 0832634 has also been critical, and is gratefully acknowledged. The resources at the Texas Advanced Computing Center (TACC), University of Texas at Austin, are used to develop ESIO.

References

1. M. Lee, N. Malaya, and R.D. Moser, "Petascale Direct Numerical Simulation of Turbulent Channel Flow on up to 786k Cores," *Proc. SC13: Int'l Conf. for High-Performance Computing, Networking, Storage and Analysis*, 2013, pp. 61:1–61:11.
2. Y. Kaneda et al., "Energy Dissipation Rate and Energy Spectrum in High-Resolution Direct Numerical Simulations of Turbulence in a Periodic Box," *Physics of Fluids*, vol. 15, no. 2, 2003; http://dx.doi.org/10.1063/1.1539855.
3. S. Hoyas and J. Jiménez, "Scaling of the Velocity Fluctuations in Turbulent Channels up to $Re_\tau = 2003$," *Physics of Fluids*, vol. 18, no. 1, 2006; http://dx.doi.org/10.1063/1.2162185.
4. J. Kim, P. Moin, and R. Moser, "Turbulence Statistics in Fully Developed Channel Flow at Low Reynolds Number," *J. Fluid Mechanics*, vol. 177, 1987, pp. 133–166.
5. The HDF Group, *Hierarchical Data Format Version 5*, user manual, 2014; www.hdfgroup.org/HDF5.
6. N. Malaya, K.W. Schulz, and R. Moser, "Petascale I/O Using HDF-5," *Proc. 2010 TeraGrid Conf.*, 2010, article no. 11.
7. M. Folk and E. Pourmal, "Balancing Performance and Preservation Lessons Learned with HDF5," *Proc. 2010 Roadmap for Digital Preservation Interoperability Framework Workshop*, 2010.
8. D.A. Donzis, P.K. Yeung, and K.R. Sreenivasan, "Dissipation and Enstrophy in Isotropic Turbulence: Resolution Effects and Scaling in Direct Numerical Simulations," *Physics of Fluids*, vol. 20, no. 4, 2008; http://dx.doi.org/10.1063/1.2907227.
9. T.A. Oliver et al., "A Semi-Implicit, Fourier–Galerkin/B-Spline Collocation Approach for DNS of Compressible, Reacting, Wall-Bounded Flow," *Bull. Am. Physical Soc.*, vol. 58, no. 18, 2013; http://meetings.aps.org/link/BAPS.2013.DFD.H30.6.
10. N. Malaya et al., "Masa: A Library for Verification Using Manufactured and Analytical Solutions," *Eng. with Computers*, vol. 29, no. 4, 2013, pp. 487–496.
11. IBM, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, white paper*, 2012; www.01.ibm.com/software/data/bigdata.

Myoungkyu Lee is a doctoral student in the Department of Mechanical Engineering at the University of Texas at Austin. His research interests include numerical simulation of wall-bounded fluid flow and code optimization for high-performance computing in scientific applications. Lee has a BS in mechanical engineering from Hanyang University, South Korea. Contact him at mk@ices.utexas.edu.

Rhys Ulerich recently completed his PhD degree in computational science, engineering, and mathematics at the University of Texas at Austin. His dissertation focused on spectral, direct numerical simulation of compressible, turbulent fluid flow. Contact him at rhys@ices.utexas.edu.

Nicholas Malaya is a doctoral student in the Department of Mechanical Engineering at the University of Texas at Austin. His research interests include computer modeling, verification, validation, and uncertainty quantification. Malaya has an MS in mechanical engineering from the University of Texas at Austin. Contact him at nick@ices.utexas.edu.

Robert D. Moser holds the W.A. "Tex" Moncrief Jr. Chair in Computational Engineering and Sciences and

is a professor of mechanical engineering in thermal fluid systems at the University of Texas at Austin. He serves as the director of the Center for Predictive Engineering and Computational Sciences (PECOS) and as the deputy director of the Institute for Computational Engineering and Sciences (ICES). His research interests include turbulence physics; direct numerical simulation and spectral methods; large-eddy simulation; cardiovascular fluid mechanics; and verification, validation, and uncertainty quantification in computational science. Moser has a PhD in mechanical engineering from Stanford University. He serves as an associate editor for *Theoretical and Computational Fluid Dynamics and Physics of Fluids*. He is a Fellow of the American Physical Society, and was awarded the NASA Medal for Exceptional Scientific Achievement. Research. Contact him at rmoser@ices.utexas.edu