CROSS-SITE COMPUTATIONS ON THE TERAGRID

The TeraGrid's collective computing resources can help researchers perform very-large-scale simulations in computational fluid dynamics (CFD) applications, but doing so requires tightly coupled communications among different sites. The authors examine a scaled-down turbulent flow problem, investigating the feasibility and scalability of cross-site simulation paradigms, targeting grand challenges such as blood flow in the entire human arterial tree.

Launched by the US National Science Foundation in August 2001, the TeraGrid (www.teragrid.org) integrates the most powerful open resources in the US, providing 50 Tflops in processing power and 1.5 Pbytes of online storage connected via a 40-Gbits-per-second network. Although the TeraGrid offers potentially unlimited scalability, the key question facing computational scientists is how to effectively adapt their applications to such a complex and heterogeneous network. Indeed, parallel scientific computing is currently at a crossroads. The emergence of the message-passing interface (MPI) as well as domain decomposition algorithms and their corresponding freeware (such as ParMETIS[1]) has made parallel computing available to the wider scientific community, allowing first-principles simulations of turbulence at very fine scales,[2] blood flow in the human heart,[3] and global climate change.[4] Unfortunately, simulations designed to capture detailed physicochemical, mechanical, or biological processes have also demonstrated widely varying characteristics.[5,6] Some applications are computation intensive, requiring extremely powerful computing systems, whereas others are data intensive,[7] necessitating the creation or mining of multiterabyte data archives.

Efficiently and effectively harnessing grid computing's power requires applications that can exploit ensembles of supercomputers, which in turn requires the ability to match application requirements and characteristics with grid resources. The challenges in developing grid-enabled applications lie primarily in the high degree of system heterogeneity and the grid environment's dynamic behavior. For example, a grid can have a highly heterogeneous and unbalanced communication network whose bandwidth and latency characteristics vary widely over time and space. Computers in grid environments can also have radically different operating systems and utilities.

Grid technology—primarily in the form of Globus-family services (www.globus.org)—has largely overcome the difficulties in managing such a heterogeneous environment. With these services' uniform mechanisms for user authentication, accounting, resource access, and data transfer, users and applications can discover and utilize disparate resources in coordinated ways. In particular, the emergence of scientific-application-oriented grid

1521-9615/05/$20.00 © 2005 IEEE
Copublished by the IEEE CS and the AIP

SUCHUAN DONG AND GEORGE EM KARNIADAKIS
Brown University
NICHOLAS T. KARONIS
Northern Illinois University and Argonne National Laboratory

middleware, such as MPICH-G2,[8] has significantly spared computational scientists from low-level details about communication handling, network topology, resource allocation, and management. However, in spite of these advancements, devising efficient algorithms for computational fluid dynamics (CFD) applications that can exploit the TeraGrid's scalability remains an enormously challenging problem. In this article, we'll present computations performed on the TeraGrid machines across a continent and show that grid computing can pave the way for the solution of future grand-challenge problems in biological and physical sciences.

Grand Challenges

Our work is motivated by two grand-challenge problems in biological and physical sciences: the simulation of blood flow in the entire human arterial tree, and the direct numerical simulation (DNS) of bluff-body turbulent wake flows. Both problems are significant from both fundamental and application viewpoints, and their resolution will have profound scientific and societal impacts.

The human arterial tree simulation problem originates from the widely accepted causal relationship between blood flow and the formation of arterial diseases such as atherosclerotic plaques.[13–15] These disease conditions seem to preferentially develop in separated and recirculating flow regions such as arterial branches and bifurcations. Blood-flow interaction in the human arterial system can occur on widely different scales; it can also occur on similar scales in different regions of the vascular system. At the largest scale, the human arterial system is coupled via the wave-like nature of the pulse information traveling from the heart through the arteries. Surgical interventions, such as bypass grafts, can block the system and alter the wave reflections, which in turn can modify the flow waveforms at seemingly remote locations. Subsequently, the modification of a local waveform can lead to the onset of undesirable stresses on the arterial walls, possibly starting another pathological event.

The challenge of modeling these interactions lies in the demand for supercomputing to model the three-dimensional (3D) unsteady fluid dynamics within the arterial branches. What makes this type of application amenable to TeraGrid computing is that we can reasonably model the waveform coupling between the sites of interest with a reduced set of 1D equations that capture the cross-sectional area and sectional velocity properties.[9] We can thus simulate the entire arterial tree by using a hybrid approach based on a reduced set of 1D equations for the overall system and detailed 3D Navier-Stokes equations for arterial branches and bifurcations.

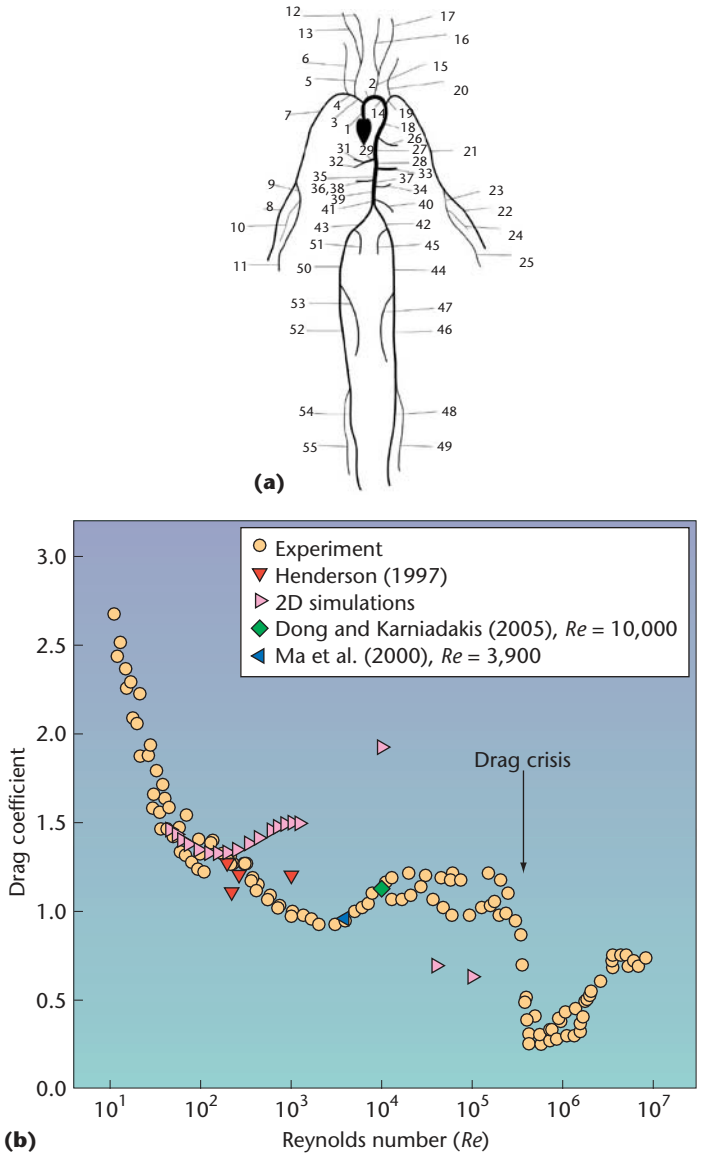

Figure 1. Two grand-challenge problems. (a) Sketch of the arterial tree containing the largest 55 arteries in the human body along with 27 artery bifurcations; (b) drag coefficient as a function of Reynolds number in cylinder flow.

To capture the flow dynamics in an artery bifurcation reasonably well, the grid resolution typically requires a mesh of 70,000 to 200,000 high-order finite elements (spectral elements with a polynomial order of 10 to 12 on each element).[10] The human arterial tree model in Figure 1a contains the largest 55 arteries in the human body with 27 artery bifurcations. The inclusion of all 27 bifurcations in the simulation with the grid resolution we just de-

scried requires a total memory of 3 to 7 Tbytes, which is beyond any single supercomputer's current capacity. The TeraGrid is the only possible way to accommodate such a simulation.

The second problem, DNS of the "drag crisis" in turbulent bluff-body flows, is a fundamental grand-challenge problem in fluid dynamics. (The drag crisis is the sudden drop of drag force near Reynolds number Re = 300,000; see Figure 1b) The need to resolve all the energetic scales in the DNS down to the Kolmogorov scale dictates that the number of grid points should be on the order of $Re^{9/4}$, or roughly a trillion grid points at drag-crisis conditions. Concentration of turbulence in the bluff-body wake and nonuniform meshing effectively reduces the required number of grid points to a few billion. The appropriate mesh now consists of approximately 512 to 768 Fourier modes along the cylinder axis and 50,000 to 80,000 spectral elements in the nonhomogeneous planes, with a spectral polynomial order of 6 to 10 on each element. A monolithic simulation with such resolutions requires more than 4 Tbytes of memory, exceeding any open supercomputer's current capacity. Just as in the human arterial tree simulation, the TeraGrid is the only viable option.

These extremely large biological and physical simulations share a common characteristic: the solution process requires tightly coupled communications among different TeraGrid sites. This is in sharp contrast to other grid application scenarios in which a monolithic application runs on one grid site while the data it produces is moved to another site for visualization or postprocessing (for example, the TeraGyroid project, www.realitygrid.org/TeraGyroid.html). A big question here is the scalability of an application involving multiple TeraGrid sites and the slowdown factor of cross-site runs compared to single-site runs under otherwise identical conditions.

Using Code

To investigate these issues and the feasibility of cross-site runs on the TeraGrid, we'll look at a scaled-down version of the drag-crisis problem—specifically, we'll use the simulation of turbulent flow past a circular cylinder at lower Reynolds numbers (Re = 3,900 and 10,000) as a prototype. We'll employ Fourier spectral expansions in the homogeneous direction and a spectral element discretization in the nonhomogeneous planes to efficiently handle a multiconnected computational domain.[10]

To conduct such a DNS, we use a high-order CFD code called Nektar (www.nektar.info/2nd_edition/) in our computations. It employs a spectral/hp element method to discretize in space and a semi-implicit scheme to discretize in time.[10] The mesh consists of structured or unstructured grids (or a combination of both) similar to those used in standard finite element and finite volume methods. We employ Jacobi polynomial expansions to represent flow variables; these expansions provide *multiresolution*, a way to hierarchically refine numerical solutions by increasing the order of the expansion (p-refinement) within each element without needing to regenerate the mesh, thus avoiding a significant overhead cost.

We can use MPICH-G2 for the cross-site communication: it's a Globus-based MPI library that extends MPICH to use the Globus Toolkit's services.[8] During the computation, MPICH-G2 selects the most efficient communication method possible between two processes, using vendor-supplied MPI whenever available. MPICH-G2 uses information in the Globus Resource Specification Language (RSL) script to create multilevel clustering of the processes based on the underlying network topology; it stores this information as attributes in the MPI communicators for applications to retrieve and exploit.

Computation Algorithms

Two cross-site parallel algorithms based on different data distribution strategies can both minimize the number of cross-site communications and overlap cross-site communications with in-site computations and communications. Both algorithms are designed based on a two-level parallelization strategy.[11] Let's consider the turbulent flow past a circular cylinder. The following incompressible Navier-Stokes equations govern the flow:

$$\frac{\partial u_i}{\partial t} + \sum_{j=1}^{3} u_j \frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{Re} \sum_{j=1}^{3} \frac{\partial^2 u_i}{\partial x_j^2},$$

$$i = 1, 2, 3$$

$$\sum_{i=1}^{3} \frac{\partial u_i}{\partial x_i} = 0,$$

where $x_i (i = 1, 2, 3)$ are the interchangeable coordinates used with x, y, z; $u_i (i = 1, 2, 3)$ are the three velocity components; and p is pressure. We normalize all length scales with the cylinder diameter D, all velocity components with the free-stream velocity U_0, and the pressure with ρU_0^2 (where ρ is the fluid density). The Reynolds number Re is expressed by

$$Re = \frac{U_0 D}{\nu},$$

where ν is the kinematic viscosity of the fluid. We refer to the term

$$u_j \frac{\partial u_i}{\partial x_j}$$

as the nonlinear term in the following discussions. Because the flow is homogeneous along the cylinder axis (assumed to be the z-direction), we perform the following Fourier expansion for the three velocity components and the pressure (letting f denote one of these variables):

$$f(x, y, z, t) = \sum_k \hat{f}_k(x, y, t) e^{ikz},$$

where $\hat{f}_k$ ($k = -N/2, \ldots, N/2 - 1$) are the Fourier modes, and N is the number of Fourier planes in the homogeneous direction.

Fourier Modal-Based Algorithm

The Fourier modal-based algorithm distributes different groups of Fourier modes onto different TeraGrid sites. It's based on the observation that a physical variable's different Fourier modes (three velocity components and the pressure) are decoupled except when evaluating the fast Fourier transform (FFT) as we compute the nonlinear terms in the Navier-Stokes equations. At each site, we compute a subset of the Fourier modes for all physical variables. As a result, solutions of any physical variable on different sites are largely independent. Coupling among different sites (hence cross-site communication) only occurs in the transposition of distributed matrices—an all-to-all type of communication—when evaluating the FFT during nonlinear term calculation.

The application takes special care to minimize the cross-site latency impact and improve cross-site bandwidth utilization. It uses the network topology information in MPICH-G2's initial communicator to enforce the data distribution strategy and ensure that, in the two-level parallelization, computations within non-homogeneous planes involve processors from the same site only.[11] A special implementation avoids unnecessary Transmission Control Protocol (TCP) polling for `MPI_ANY_SOURCE` exchanges on communicators involving in-site communications only.[8] We also agglomerate the data of different physical variables such that we perform a single cross-site matrix transposition instead of several separate transpositions for different variables. Therefore, we only need two cross-site communications (one forward and one backward transform) when computing nonlinear terms. Compared to the usual approach of performing the FFTs of different physical variables separately, data agglomeration minimizes the number of cross-site communications and increases each message's size, thus reducing the latency effect. The larger message size also improves cross-site bandwidth utilization.

Physical Variable-Based Algorithm

The physical variable-based algorithm computes physical variables on different TeraGrid sites and exploits the coupling characteristics among them in the Navier-Stokes equations. Computations of various velocity components are independent except for their interdependence in the nonlinear term. A mutual dependence exists between the velocity and the pressure, for example: computation of pressure depends on both the velocity divergence and the nonlinear terms and velocity gradients on the boundaries; computation of velocity depends on the pressure gradient.

For simplicity, let's assume we're using three TeraGrid sites; in the physical variable-based algorithm, we compute all of a velocity component's Fourier modes along with a third of the pressure Fourier modes on a different site. The computation will involve three cross-site communications, with the first occurring prior to the nonlinear solve. (Here, the nonlinear solve—and the pressure and velocity solves in subsequent discussions—refers to a three-step time-integration scheme.[10]) Each site must communicate its own velocity component to—and receive other velocity components from—the other sites for nonlinear term calculation. With Nektar's two-level parallelization,[11] a processor at one site communicates only with the other two sites' corresponding processors, and thus cross-site communication involves three processors. Different processors at the same site participate in parallel independent communications.

The second cross-site communication, a SUM reduction for velocity divergence and pressure boundary conditions, occurs prior to the pressure solve. Again, a processor participates in the reduction only with corresponding processors at the other two sites (different processors at the same site participate in parallel independent reductions). The third cross-site communication occurs prior to the velocity solve: it distributes the pressure gradient data to the other sites and receives the pressure gradient component that it computes from the other sites.

Simulation Results

We've obtained a large amount of simulation re-

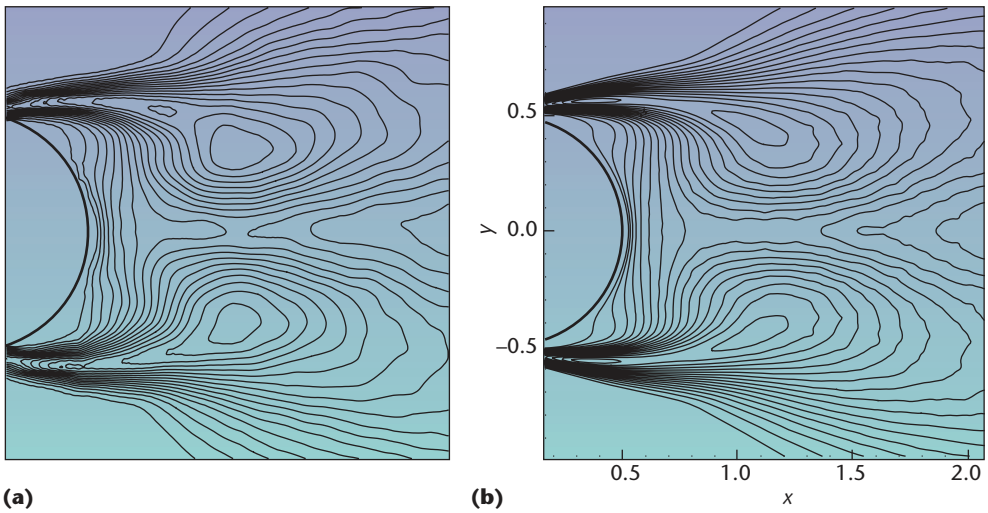

Figure 2. Turbulent flow past a circular cylinder. At Reynolds number Re = 10,000, we can compare streamwise root-mean-square (RMS) velocity u'/U_0 between (a) the particle-image-velocimetry (PIV) experiment and (b) current direct numerical simulation (DNS). Contours are plotted on the same levels for the experiment and the DNS: u'_{min}/U_0 = 0.1 and the incremental value between contour lines $\Delta u'/U_0$ = 0.025.

sults, but this article's emphasis is on the computing aspect (algorithm and performance), so we've chosen to include only one comparison with experiment here. Specifically, let's examine some simulation results for turbulent flows past bluff bodies obtained on the TeraGrid clusters. We can compare the statistical characteristics in the turbulent wake of a circular cylinder at Reynolds number Re = 10,000 between our 3D DNS and the particle-image-velocimetry (PIV) experiments.[12] This Reynolds number is the highest that the DNS has achieved for this flow so far.

Figure 2 shows a comparison of the normalized streamwise root-mean-square (RMS) velocity fluctuation u' normalized by the free-stream velocity U_0 between the experiment (Figure 2a) and the simulation (Figure 2b). We plotted experimental and DNS results on identical contour levels, with a minimum RMS value u'/U_0 = 0.1 and an incremental value of 0.025 between contour lines. The distribution patterns show strong fluctuations in the separating shear layers and two maxima associated with the vortex formation. The downstream locations of the RMS maxima are essentially the same for both experiment and simulation (at x = 1.14 for the experiment and at x = 1.13 for the simulation), and the respective peak values are also the same: u'_{max}/U_0 = 0.5.

At Reynolds numbers in the subcritical range (above 1,000 and below the "drag crisis" Reynolds number), the separating shear layers behind the cylinder become unstable and small-scale vortices develop in the shear layers (so-called shear-layer vortices). Our simulation data shows that the frequency of shear-layer vortices follows a scaling of $Re^{0.67}$ with respect to the Reynolds number, in agreement with experimental observations.[17] Furthermore, the values of shear-layer frequencies from our computation agree with the experimental values; for example, at Re = 10,000, our simulation predicts a value (normalized by Strouhal frequency) of 11.83 whereas it's 11.25 from experiment.[17]

Performance Results

To evaluate the efficiency of computation algorithms, we conducted single-site runs on three TeraGrid sites—the US National Center for Supercomputing Applications (NCSA), the San Diego Supercomputer Center (SDSC), and the Pittsburgh Supercomputing Center (PSC). We also ran cross-site runs between the SDSC and the NCSA, and between the two TeraGrid machines at the PSC. The NCSA and SDSC TeraGrid machines have Intel IA-64 processors (Itanium-2, 1.5 GHz) whereas those at the PSC have Compaq Alpha processors (Alpha EV68, 1 GHz).

Single-Site Performance

Figure 3a demonstrates Nektar's scalability, showing parallel speedup with respect to the number of processors on the PSC TeraGrid cluster for a fixed problem size with 300 million degrees of freedom. The parallel efficiency exceeds 95 percent on 1,024 processors and 85 percent on 1,536 processors. The test problem here is the turbulent cylinder

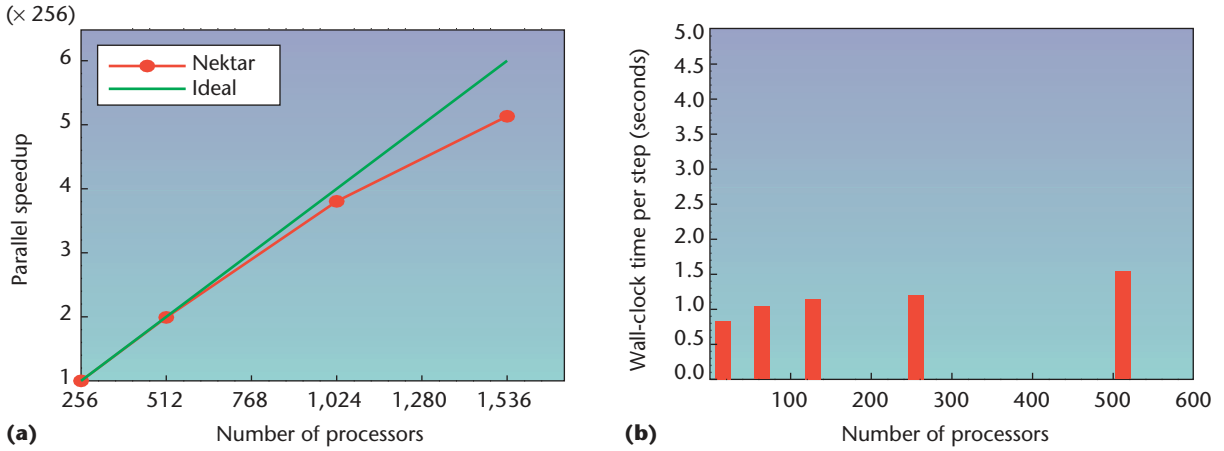

Figure 3. Nektar's performance on the Pittsburgh Supercomputing Center's TeraGrid cluster. (a) Parallel speedup with respect to the number of processors for a fixed problem size, and (b) wall-clock time per step (in seconds) as a function of the number of processors for a fixed workload per processor. As the problem size increases, the number of processors increases proportionally to keep the workload per processor constant.

flow at Reynolds number Re = 10,000 based on the free-stream velocity and cylinder diameter. We used a spectral element mesh with 9,272 triangular elements in the nonhomogeneous planes; the number of Fourier planes in the spanwise direction is 256 in this test.

Scalability for a fixed workload per processor is another important measure. In this set of tests, as the problem size increases, the number of processors increases proportionally such that the workload on each processor remains unchanged. The test problem is still the turbulent cylinder flow at Re = 10,000; in the nonhomogeneous plane, we use the same grid resolution, but vary the problem size by varying the number of Fourier planes in the spanwise direction. In this test, the number of Fourier planes increases from 8 to 128, and the number of processors increases proportionally from 32 to 512 to keep the workload per processor constant. For the case of 512 processors (128 Fourier planes), we use 64 processors for computations in the homogeneous direction and eight processors for computations in the two-level parallelization's nonhomogeneous planes. Figure 3b shows the wall-clock time per step (in seconds) as a function of the number of processors from the test. The ideal result would be a constant wall-clock time per step for any number of processors (flat curve), but we see (only) a slight increase in wall-clock time as the number of processors increases from 32 to 512, indicating good scalability.

We used native (vendor) MPI libraries in all these tests, but because cross-site communications are based on the MPICH-G2 library, we also want to test the performance differences between MPICH-G2 and the native MPI implementation in a single-site environment. MPICH-G2 hides low-level operational details from the application, including communication channel selection (vendor or TCP), data conversion, resource allocation, and computation management. MPICH-G2 has taken measures such as eliminating memory copies and unnecessary message polling to minimize the overhead cost.[8]

Figure 4 shows a performance comparison of Nektar compiled with MPICH-G2 and native MPI on the SDSC's TeraGrid cluster (we expect the results would be valid for all TeraGrid machines, particularly the ones that have identical compute and switch hardware). The test problem is the turbulent cylinder flow at Reynolds number Re = 3,900. We plot the wall-clock time per step as a function of the total number of processors for a fixed problem size, and we use a spectral element mesh with 902 triangular elements in the nonhomogeneous planes and 128 planes in the spanwise direction; the polynomial order is 8 on all elements. MPICH-G2 demonstrates a performance virtually identical to the native MPI, indicating a negligible overhead cost.

Figure 5 compares the performances of the Fourier modal-based and physical variable-based algorithms discussed earlier—specifically, it compares the wall-clock time per step and the speedup factor with respect to the number of processors for a fixed problem size with 24 Fourier planes in the spanwise direction. Due to the configuration of the physical variable-based algorithm, the number of

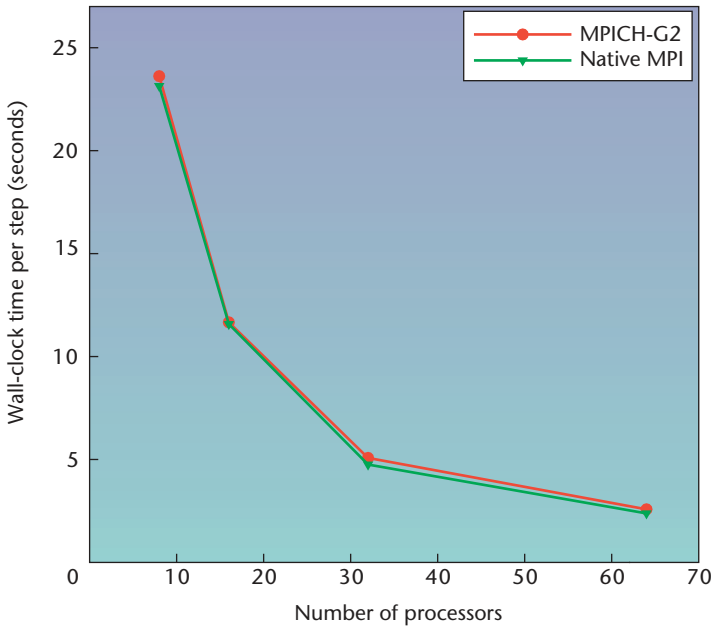

Figure 4. Nektar performance. Comparing MPICH-G2 and native MPI on the San Diego Supercomputer Center's TeraGrid cluster, we can plot wall-clock time per step versus the number of processors for a fixed problem size.

processors tested ranges from 3 to 96, and the parallel speedup is computed based on the wall-clock time on three processors. The Fourier modal-based algorithm consistently performs better, most likely due to the data agglomeration and reduced number of cross-site communications.

Cross-Site Performance

To assess the efficiency on cross-site platforms, we use the Fourier modal-based cross-site algorithm to conduct a series of cross-site runs with Nektar and MPICH-G2 on the SDSC and NCSA TeraGrid machines.

Our test problem was the turbulent flow past a cylinder at Reynolds number Re = 3,900. As we did earlier, we use spectral element mesh with 902 triangular elements in the nonhomogeneous planes (with a spectral element order of 8 on all elements); the number of Fourier planes in the spanwise direction varies from 16 to 128. We first investigate the scaling for a fixed problem size with 128 Fourier planes in the spanwise direction. Figure 6 shows the wall-clock time per step (in seconds) as a function of the total number of processors for cross-site runs between the NCSA and SDSC TeraGrid machines, together with results for single-site runs on the NCSA machine under identical configurations. The total number of processors varies from 16 to 256. For cross-site runs, the processors are split between the NCSA and SDSC TeraGrid machines—in a 256-CPU cross-site run, for example, 128 processors are from both the NCSA and the SDSC. We use MPICH-G2 in both single-site and cross-site runs; we performed at least three independent runs for each case. In single-site runs, the wall-clock time essentially shows a linear relationship with respect to the number of processors, indicative of a near-linear speedup.

In cross-site runs, the wall-clock time decreases significantly with the increasing number of processors. In fact, the wall-clock time–CPU curve shows a dramatic decrease, nearly an order of magnitude, as the number of processors increases from 32 to 64. To check if software errors cause this performance jump, we took special care to ensure that we obtained identical, correct computation results in all test cases, including those on 32 and 64 processors, and that we conducted several independent runs for each case (at least three for smaller CPU counts, at least five for larger ones). We conducted the tests at a special reserved time for both machines, with exclusive access to about a third of the TeraGrid machine at the NCSA and the whole TeraGrid machine at the SDSC. The performance results are repeatable, with only slight variation in exact values (Figure 6 shows the mean values). We're convinced that these aren't spurious data points. The exact reason for this performance jump isn't totally clear at this point. It can result from several factors such as the communication characteristics of the network connecting the NCSA and SDSC machines. As expected, a cross-site run is slower than the corresponding single-site run on the same total number of processors. The slowdown ratio, however, decreases dramatically as the number of processors increases. Beyond 32 processors, the slowdown ratio of the cross-site runs ranges from 1.5 to 2.0.

Now let's look at the scaling for a fixed workload per processor. The problem size is varied by changing the number of Fourier planes in the spanwise direction. In this set of tests, we started with eight Fourier planes in the spanwise direction, and doubled the number for each test until we reached 128. Correspondingly, we increased the total number of processors proportionally, from eight to 128, such that the workload on each processor remained unchanged. Figure 7 plots the wall-clock time per step (in seconds) as a function of the total number of processors for cross-site runs between the NCSA and SDSC TeraGrid machines, as well as results for single-site runs on the NCSA machine under identical configurations. In

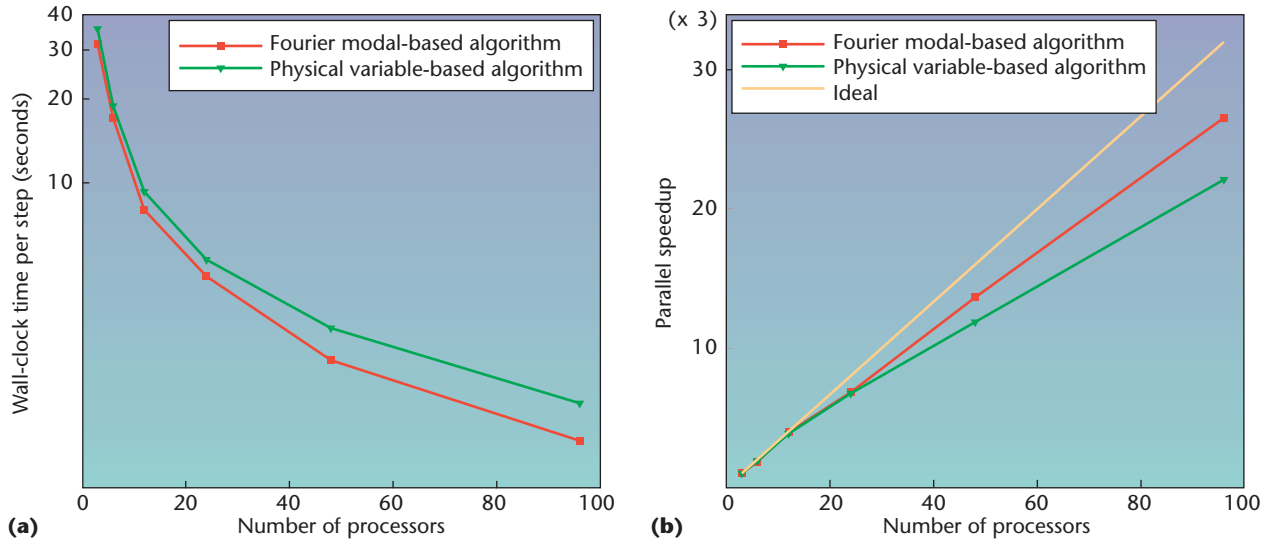

Figure 5. Algorithm performance. Comparing the Fourier modal-based and physical variable-based algorithms for a fixed problem size on the Pittsburgh Supercomputing Center's TeraGrid cluster, we can match up (a) wall-clock time and (b) parallel speedup as a function of the total number of processors. Speedup is calculated based on the wall-clock time on three processors.

cross-site runs, again, half the processors are from the NCSA and the other half are from the SDSC; we used MPICH-G2 in both cross- and single-site runs. In single-site runs, the wall-clock time increases very slightly as the number of processors increases from eight to 128, indicating excellent scalability. In cross-site runs, we observe a larger increase in wall-clock time as the number of processors increases from eight to 32. Again, we see a dramatic decrease in wall-clock time as the number of processors increases from 32 to 64. Compared to single-site runs on the same number of processors, the slowdown ratio of cross-site runs decreases significantly beyond 32 processors.

We also examined the influence of processor configuration on cross-site run performance. Table 1 lists the wall-clock time per step on a total of 256 processors in cross-site runs between the NCSA and SDSC machines with a fixed problem size for turbulent cylinder flow at Re = 10,000. We tested several different configurations with a different number of processors from each site, but the wall-clock timing for different configurations is essentially the same; we didn't observe any significant influence of processor configuration on performance.

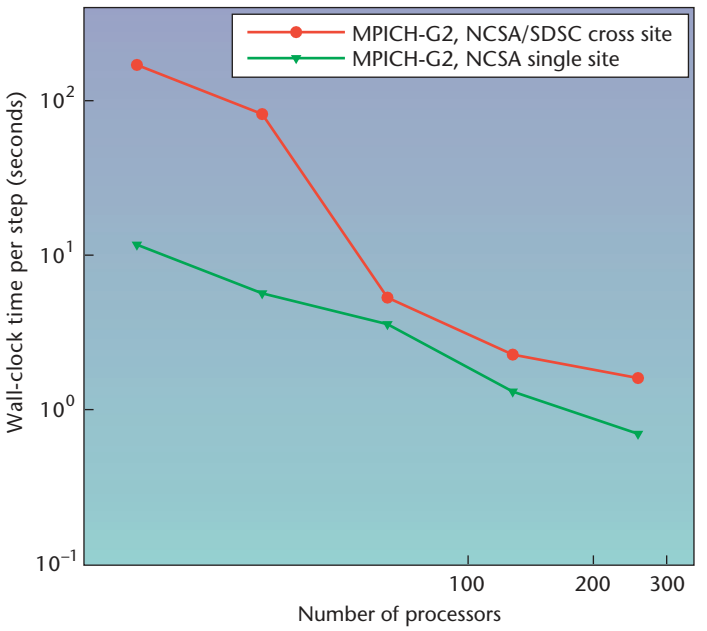

Figure 6. Benchmarking for a fixed problem size. In comparing the US National Center for Supercomputing Applications (NCSA)/San Diego Supercomputer Center (SDSC) cross-site runs with the NCSA single-site runs, we can show the wall-clock time per step as a function of the total number of processors for a simulation of turbulent wake.

Compared to single-site runs, the slowdown ratio of cross-site runs decreases dramatically as the number of processors increases. We performed the cross-site runs with a Fourier-modal based algorithm, which is characterized by a stressful all-to-all type of cross-site communication and in a sense represents the worst-case scenario. For applications

Table 1. The performance of cross-site runs for flow past a cylinder at Reynolds number *Re* = 10,000.

Cross-site run	CPUs from the US National Center for Supercomputing Applications	CPUs from the San Diego Supercomputer Center	Wall-clock time per step (seconds)
Total of 256 CPUs	128	128	16.307
	144	112	16.467
	160	96	15.935

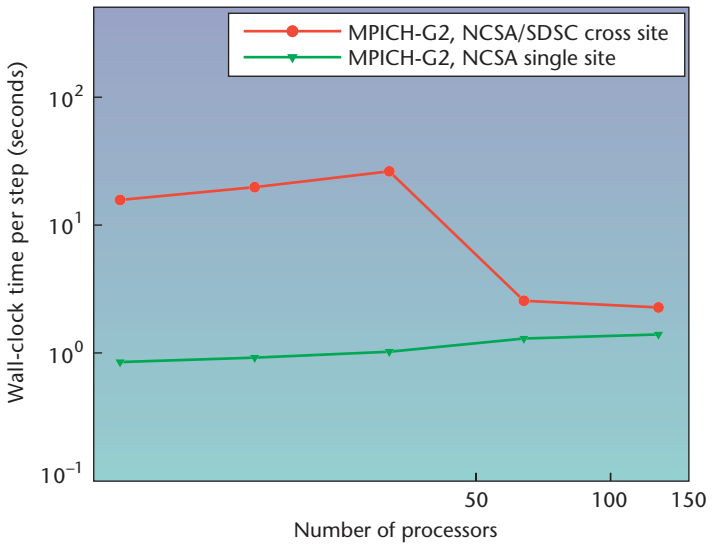

Figure 7. Benchmarking for a fixed workload per processor. In comparing the US National Center for Supercomputing Applications (NCSA)/San Diego Supercomputer Center (SDSC) cross-site runs with the NCSA single-site runs, the problem size increases proportionally as the number of processors increases such that the workload per processor remains unchanged.

characterized by less stressful communication patterns, we can achieve even better performance for cross-site runs. The human arterial tree simulation shows a much less demanding communication characteristic; we're currently developing and testing this application in cross-site computations on the TeraGrid as well as on machines between the US and UK. Several techniques can further boost cross-site performance to possibly even match single-site performance—for example, multithreading to truly overlap cross-site communications with in-site computations, and User Datagram Protocol (UDP)-based messaging to improve cross-site communication's bandwidth utilization. Developers are currently incorporating these approaches into the next-generation implementation of MPICH-G2. Grid computing enabled by Globus/MPICH-G2 and other grid services and middleware (see NaReGi, www.naregi.org; DEISA, www.deisa.org; and PACX-MPI16) hold the key to the solution of grand-challenge problems in biological and physical sciences.

Acknowledgments

The US National Science Foundation, the Office of Naval Research, and the Defense Advanced Research Projects Agency (DARPA) supported this work. Computer time for the TeraGrid was provided through the US National Center for Supercomputing Applications (NCSA), the San Diego Supercomputer Center (SDSC), and the Pittsburgh Supercomputing Center (PSC). We thank John Towns (NCSA), David O'Neal (PSC), Donald Frederick (SDSC), Frank Wells (NCSA), Dongju Choi (SDSC), and the TeraGrid support team for their assistance in setting up the TeraGrid for the benchmark tests.

References

1. G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM J. Scientific Computing*, vol. 20, no. 1, 1998, pp. 359–392.
2. M. Yokokawa et al., "16.4-Tflops Direct Numerical Simulation of Turbulence by a Fourier Spectral Method on the Earth Simulator," *Proc. Supercomputing 2002*, IEEE CS Press, 2002; www.sc-conference.org/sc2002/.
3. S.J. Kovacs, D.M. McQueen, and C.S. Peskin, "Modelling Cardiac Fluid Dynamics and Diastolic Function," *Philosophical Trans. Royal Soc. London, Series A: Mathematical Physics and Eng. Science*, vol. 359, 15 June 2001, pp. 1299–1314.
4. S. Shingu et al., "A 26.58 Tflops Global Atmospheric Simulation with the Spectral Transform Method on the Earth Simulator," *Proc. Supercomputing 2002*, IEEE CS Press, 2002; www.sc-conference.org/sc2002/.
5. G. Allen et al., "Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus," *Proc. Supercomputing 2001*, IEEE CS Press, 2001; www.sc2001.org/.
6. M. Russel et al., "The Astrophysics Simulation Collaboratory: A Science Portal Enabling Community Software Development," *Cluster Computing*, vol. 5, no. 3, 2002, pp. 297–304.
7. M.S. Allen and R. Wolski, "The Livny and Plank-Beck Problems: Studies in Data Movement on the Computational Grid," *Proc. Supercomputing 2003*, IEEE CS Press, 2003; www.sc-conference.org/sc2003/.
8. N.T. Karonis, B. Toonen, and I. Foster, "MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface," *J. Parallel and Distributed Computing*, vol. 63, no. 5, 2003, pp. 551–563.
9. S.J. Sherwin et al., "Computational Modeling of 1D Blood Flow with Variable Mechanical Properties in the Human Arterial System," *Int'l J. Numerical Methods in Fluids*, vol. 43, nos. 6–7, 2003, pp. 673–700.
10. G.E. Karniadakis and S.J. Sherwin, *Spectral/hp Element Methods*

for CFD, 2nd ed., Oxford Univ. Press, 2005.

11. S. Dong and G.E. Karniadakis, "Multilevel Parallelization Models in CFD," *J. Aerospace Computing, Information, Comm.*, vol. 1, no. 6, 2004, pp. 256–268.
12. A. Ekmekci, *Flow Structure from a Circular Cylinder with Defined Surface Perturbations*, PhD dissertation, Dept. Mechanical Eng. and Mechanics, Lehigh Univ., 2005.
13. C.G. Caro, J.M. Fitz-Gerald, and R.C. Schroter, "Atheroma and Arterial Wall Shear: Observation, Correlation and Proposal of a Shear Dependent Mass Transfer Mechanism for Atherogenesis," *Proc. Royal Soc. London Series B,* vol. 177, 1971, pp. 109–159.
14. M.H. Friedman, G.M. Hutchins, and C.B. Bargeron, "Correlation between Intimal Thickness and Fluid Shear in Human Arteries," *Atherosclerosis*, vol. 39, 1987, p. 425.
15. C. Zarins et al., "Carotid Bifurcation Atherosclerosis: Quantitative Correlation of Plaque Localization with Flow Velocity Profiles and Wall Shear Stress," *Circulation Research: J. Am. Heart Assoc.*, vol. 53, 1983, pp. 502–514.
16. E. Gabriel et al., "Distributed Computing in a Heterogeneous Computing Environment," *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, V. Alexandrov and J. Dongarra, eds., LNCS 1497, Springer-Verlag, 1998, pp. 180–187.
17. A. Prasad and C.H.K. Williamson, "The Instability of the Shear Layer Separating from a Bluff Body," *J. Fluid Mechanics*, vol. 333, Feb. 1997, pp. 375–402.

Suchuan Dong is a research assistant professor in the Division of Applied Mathematics at Brown University. His research interests include high-performance computing, flow structure interaction, biofluids and turbulence simulations, and bubbly flows. Dong has a PhD in mechanical engineering from the State University of New York at Buffalo. He is a member of the American Society of Mechanical Engineers, the American Institute of Aeronautics and Astronautics, and the American Physical Society. Contact him at sdong@dam.brown.edu.

George Em Karniadakis is a professor of applied mathematics at Brown University. His research interests include spectral methods in unstructured grids, parallel simulations of turbulence in complex geometries, and microfluid simulations. Karniadakis has a PhD in mechanical engineering from MIT. He is a fellow of the American Physical Society and the American Society of Mechanical Engineers, and a member of the American Institute of Aeronautics and Astronautics and SIAM. Contact him at gk@dam.brown.edu.

Nicholas T. Karonis is an associate professor at Northern Illinois University and a resident associate guest of Argonne National Laboratory's Mathematics and Computer Science Division. His current research interest is message-passing systems in computational grids. Karonis has a BS in finance and a BS in computer science from Northern Illinois University, an MS in computer science from Northern Illinois University, and a PhD in computer science from Syracuse University. Contact him at karonis@niu.edu.