Terascale Spectral Element Algorithms and Implementations

H.M. Tufo* and P.F. Fischer†

June 7, 1999

Abstract

We describe the development and implementation of an efficient spectral element code for multimillion gridpoint simulations of incompressible flows in general two- and three-dimensional domains. We review basic and recently developed algorithmic underpinnings that have resulted in good parallel and vector performance on a broad range of architectures, including the terascale computing systems now coming on line at the DOE labs. Sustained performance of 219 GFLOPS has been recently achieved on 2048 nodes of the Intel ASCI-Red machine at Sandia.

Introduction

One of the primary motivations driving computational science is to augment experiments as a means of scientific investigation. To this end, we are currently working with several collaborators on the development and use of a spectral element code for comparative numerical and experimental studies on challenging problems in fluid mechanics and heat transfer. As illustrated in Fig. 1, these problems include the generation of hairpin vortices resulting from the interaction of a flat-plate boundary layer with a hemispherical roughness element; forced convection heat transfer in grooved and grooved-flat channels; modeling the Geophysical Fluid Flow Cell (GFFC) space laboratory experiment of buoyant convection in a rotating hemispherical shell; and study of Rayleigh-Taylor instability.

For performance benchmarking, we consider numerical simulation of the interaction of a flat-plate boundary layer with an isolated hemispherical roughness element at Reynolds number $Re_\delta = 1600$. This problem has been studied experimentally by Acalar and Smith [1] and, more recently, by Klebanoff, Cleveland, and Tidstrom [10]. Of principal interest in these studies is the creation of hairpin vortices that form an interlacing pattern in the wake of the hemisphere, lift away from the wall, and are stretched by the shearing action of the boundary layer, since the tails remain in the low-speed (near-wall) region of the flow while their heads are entrained in the high-speed region. Hairpin vortices are of interest because they are frequently observed in whole or in part in transitional and turbulent boundary layers and are believed to play an important role turbulence production. [18, 19].

This paper presents a brief overview of the critical algorithmic and implementation features that have led to efficient simulation of incompressible flows on terascale architectures.

Spectral Element Discretization

The spectral element method is a high-order weighted residual technique developed by Patera and coworkers in the '80s that couples the tensor-product efficiency of global spectral methods with the geometric flexibility of finite elements [12, 15]. Locally, the mesh is structured, with the solution, data, and geometry expressed as sums of Nth-order tensor-product Lagrange polynomials, based on the Gauss or Gauss-Lobatto (GL) quadrature points. Globally, the mesh is an unstructured array of K deformed hexahedral elements and can include geometrically nonconforming elements. The discretization is illustrated in Fig. 2, which shows a three-element mesh in $\mathbb{R}^2$ with the GL grid for the case $N = 4$. Also shown is the reference (r, s) coordinate system used for all function evaluations.

*Center on Astrophysical Thermonuclear Flashes, University of Chicago, Chicago, IL 60637
†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439

DISCLAIMER

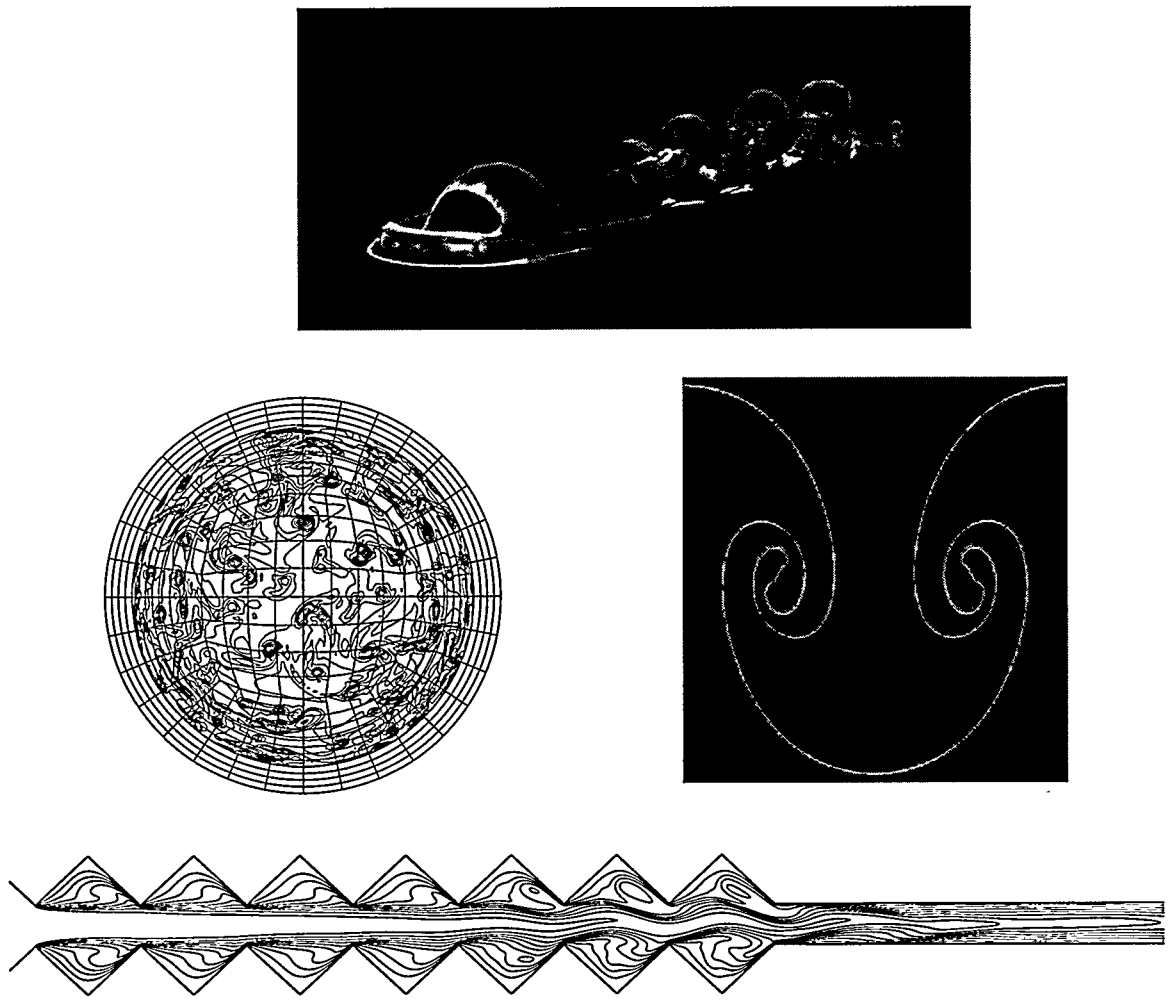

Figure 1: Recent spectral element simulations. Clockwise from top: hairpin vortex generation in wake of hemispherical roughness element ($Re_\delta = 700$); two-dimensional Rayleigh-Taylor instability; temporal-spatial evolution of convective instability in heat-transfer augmentation simulations; spherical convection simulation of the geophysical fluid flow cell (GFFC) at $Ra = 110\text{K}$, $Ta = 1,440\text{K}$.

For problems having smooth solutions, such as the incompressible Navier-Stokes equations, exponential convergence is obtained with increasing N, despite the fact only C^0 continuity is enforced across elemental interfaces. This is demonstrated in Table 1, which shows the computed growth rates when a small-amplitude Tollmien-Schlichting wave is superimposed on plane Poiseuille channel flow at $Re = 7500$, following [7, 14]. The amplitude of the perturbation is 10^{-5}, so that the Navier-Stokes results can be compared directly with linear theory. Three error measures are computed: $error_1$ and $error_2$ are the relative amplitude errors at the end of the first and second periods, respectively, and $error_g$ is the error in the growth rate at a convective time of 50. From Table 1, it is clear that doubling the number of points in each spatial direction yields several orders of magnitude reduction in error, implying that just a small increase in resolution is required for very good accuracy. The significance of this is underscored by the fact that, in three dimensions, the effect on the number of gridpoints scales as the cube of the relative savings in resolution.

Operator Evaluation

The computational efficiency of spectral element methods derives from the use of tensor-product forms. To illustrate, we express the stiffness matrix for an undeformed element k in $\mathbb{R}^2$ as a sum of

Table 1: Spatial convergence, O-S problem: $K = 15$, $\Delta t = .003125$

N	$E(t_1)$	$error_1$	$E(t_2)$	$error_2$	$error_g$
7	1.11498657	0.003963	1.21465285	0.037396	0.313602
9	1.11519192	0.003758	1.24838788	0.003661	0.001820
11	1.11910382	0.000153	1.25303597	0.000986	0.004407
13	1.11896714	0.000016	1.25205855	0.000009	0.000097
15	1.11895646	0.000006	1.25206398	0.000014	0.000041

tensor products of one-dimensional operators,

$$A^k = \widehat{B}_y \otimes \widehat{A}_x + \widehat{A}_y \otimes \widehat{B}_x \,, \tag{1}$$

where $\widehat{A}$ and $\widehat{B}$ are the one-dimensional stiffness and mass matrices associated with the respective spatial dimensions. If $\underline{u}^k = u^k_{ij}$ is the matrix of nodal values on element k, then a typical matrix-vector product required of an iterative solver takes the form

$$\begin{aligned}(A^k \underline{u}^k)_{lm} &= \sum_{i=0}^{N}\sum_{j=0}^{N}(\widehat{B}_{y,mj}\widehat{A}_{x,li}\underline{u}^k_{ij} + \widehat{A}_{y,mj}\widehat{B}_{x,li}\underline{u}^k_{ij}) \\ &= \widehat{A}_x \underline{u}^k \widehat{B}_y^T + \widehat{B}_x \underline{u}^k \widehat{A}_y^T \,.\end{aligned}$$

The latter form illustrates how the tensor-product basis leads to matrix-vector products ($A\underline{u}$) being recast as *matrix-matrix* products, a feature central to the efficiency of spectral element methods. Similar forms result for other operators and for complex geometries. For example, evaluation of the discrete Laplacian for a deformed hexahedral element in $\mathbb{R}^3$ takes the form

$$A^k \underline{u}^k = \begin{pmatrix} D_r \\ D_s \\ D_t \end{pmatrix}^T \begin{pmatrix} G_{rr} & G_{rs} & G_{rt} \\ G_{rs} & G_{ss} & G_{st} \\ G_{rt} & G_{st} & G_{tt} \end{pmatrix} \begin{pmatrix} D_r \\ D_s \\ D_t \end{pmatrix} \underline{u}^k \,, \tag{2}$$

where $D_r = (I \otimes I \otimes \hat{D})$, and so forth, and the G_{ij}'s are diagonal matrices of order $(N+1)^3$ that combine the quadrature weights with the Jacobian and metrics associated with the transformation from the physical to computational domain. The total work per element is $12N^4 + 15N^3$, and the required number of memory references is $7N^3$. Note that A^k in (2) is full, implying that the work and storage would be $O(N^6)$ if it was explicitly computed and stored.

The matrix-vector product (2) requires six matrix-matrix products of form $(N \times N) \times (N \times N^2)$, and so forth, associated with the derivative operators, D. These typically account for roughly 90%

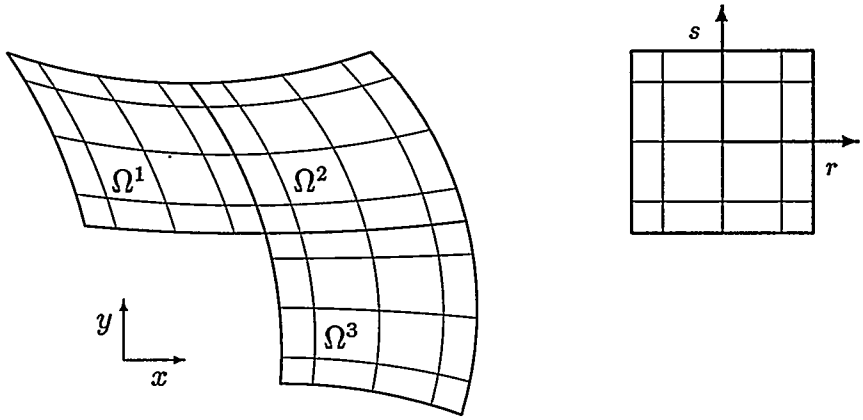

Figure 2: Example of spectral element discretization in $\mathbb{R}^2$ showing GLL nodal lines for $(K, N) = (3, 4)$.

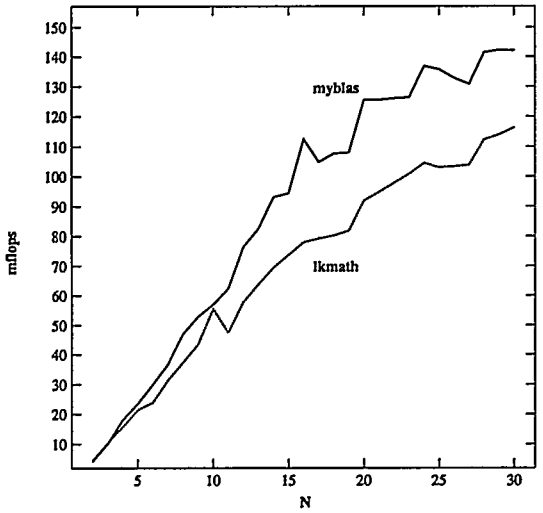

Figure 3: ASCI-Red-333 DGEMM() performance on uncached data.

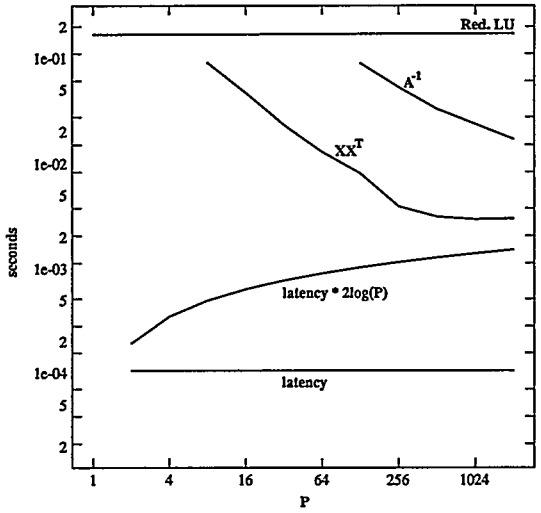

Figure 4: ASCI-Red-200 solve times for a 16129 d.o.f. coarse grid problem.

of the work and are usually implemented via calls to DGEMM, unless hand-unrolled F77 loops prove faster on a given platform. For ASCI-Red, we considered two versions of DGEMM, the standard one obtained with the `-lkmath` link option, and the "myblas.a" version developed by Greg Henry at Intel. Timings for noncached, $N \times N$ matrix-matrix products are plotted in Fig. 3 and show an expected upper bound on performance of 20 to 110 MFLOPS per processor for the range of $N =$ 5–16 typically encountered in practice. The myblas.a library is significantly faster than the standard DGEMM and is used in all the single-processor-per-node tests presented below. However, it was not stable for the dual-processor timings, so we used the standard library in those tests. Note that, while the work in (2) scales as $O(N^4)$, we can expect the time to scale more slowly because of DGEMM performance gains with increasing N.

The message-passing-based parallel implementation follows the standard SPMD model in which contiguous groups of elements are distributed to processors. Since iterative solvers are used, the principal communication kernel is the gather-scatter operation required for the residual vector assembly procedure. Because data is always stored on an element-by-element basis, the gather-scatter procedure required for residual evaluation is combined into a single communication phase wherein shared nodal values are exchanged and summed. This is a single local-to-local transformation, rather than separate gather and scatter phases common to many finite element implementations. Communication overhead is further reduced through use of a recursive spectral bisection based element partitioning scheme to minimize the number of vertices shared amongst processors [16].

The gather-scatter operation is implemented by using a stand-alone MPI/NX-based message-passing utility that supports a vector mode for problems having multiple degrees-of-freedom per vertex as well as a general set of commutative/associative operations [19]. The easy-to-use interface requires only two calls:

handle=gs-init(global-node-numbers,n) and *ierr=gs-op(u,op,handle)*,

where *global-node-numbers()* associates the n local values contained in the vector $u()$ with their global counterparts, and *op* denotes the reduction operation performed on shared elements of $u()$.

On ASCI-Red, additional parallelism has been obtained by exploiting the second processor for a few compute-intensive matrix-vector product evaluations (particularly the pressure operator).

Time Advancement

The Navier-Stokes time-stepping is based on the second-order operator splitting methods developed in [2, 13]. The convective term is expressed as a material derivative, and the resultant form is

discretized via a stable second-order BDF scheme:

$$\frac{\tilde{\underline{\mathbf{u}}}^{n-2} - 4\tilde{\underline{\mathbf{u}}}^{n-1} + 3\underline{\mathbf{u}}^n}{2\Delta t} = S(\underline{\mathbf{u}}^n),$$

where $S(\underline{\mathbf{u}}^n)$ is the linear symmetric Stokes problem to be solved implicitly, and $\tilde{\underline{\mathbf{u}}}^{n-q}$ is the velocity field at time step $n-q$ computed as the explicit solution to a pure convection problem. The sub-integration of the convection term permits values of Δt corresponding to convective CFL numbers of 2–5, thus significantly reducing the number of (expensive) Stokes solves.

The Stokes problem is of the form

$$\begin{bmatrix} \mathbf{H} & -\mathbf{D}^T \\ -\mathbf{D} & 0 \end{bmatrix} \begin{pmatrix} \underline{\mathbf{u}}^n \\ \underline{p}^n \end{pmatrix} = \begin{pmatrix} \mathbf{B}\underline{\mathbf{f}} \\ \underline{0} \end{pmatrix}$$

and is also treated by second-order splitting, resulting in subproblems of the form:

$$H\underline{u}_i^n = \underline{f}_i,$$

for the each velocity component ($i = 1, \ldots, 3$), and

$$E\underline{p}^n = \underline{g}^n.$$

Here, H is a diagonally dominant Helmholtz operator representing the parabolic component of the momentum equations and is readily treated via Jacobi-preconditioned conjugate gradients; $E := \mathbf{D}\mathbf{B}^{-1}\mathbf{D}^T$ is the Stokes Schur complement governing the pressure, and $\mathbf{B}$ is the (diagonal) mass matrix in the velocity space. E is a consistent Poisson operator and is effectively preconditioned by using the overlapping additive Schwarz procedure of Dryja and Widlund [3, 7, 8].

Solvers

The pressure solve is the most computationally intensive step in the time advancement scheme. In addition to preconditioning, the solution time is further reduced by first projecting the solution at time level n onto the space of L previous solutions ($L \sim 25$, typ.) and solving only for the perturbation from this solution. The magnitude of the perturbation is $O(\Delta t^L) + O(\epsilon)$, where ϵ is the iteration tolerance. Under normal production tolerances the projection step yields a two- to four-fold reduction in work [5].

The overlapping Schwarz preconditioner requires a local solve for each (overlapping) subdomain, plus a global coarse grid solve based upon the spectral element vertex mesh. The local subdomain solves are based upon the fast diagonalization method [11], in which the inverse of A^k (1) is expressed as

$$(A^k)^{-1} = (S_y \otimes S_x)[I \otimes \Lambda_x + \Lambda_y \otimes I]^{-1}(S_y^T \otimes S_x^T),$$

where S is the matrix of eigenvectors and Λ the diagonal matrix of eigenvalues solving the generalized eigenvalue problem $\widehat{A}\underline{z} = \lambda\widehat{B}\underline{z}$. The tensor-product forms involving S can be applied via fast matrix-matrix products as described above, and the storage is minimal. The method is not exact when the subdomain is deformed, but suffices as a preconditioner [8].

The coarse grid problem, $\underline{x}_c = A_c^{-1}\underline{b}_c$ is a well-known source of difficulty on large distributed-memory architectures [4, 9]. The problem arises because the solution and data are distributed vectors, and A_c^{-1} is completely full, implying the need for an all-to-all communication. Moreover, because there is very little work on the coarse grid (typ. O(1) d.o.f. per processor), the problem is communication intensive. We have recently developed a fast coarse grid solution algorithm that readily extends to thousands of processors [6, 17]. It is based upon construction of sparse factors $XY^T = A_c^{-1}$. The solution is thus cast a pair of fully concurrent matrix-vector products. Moreover, it can be shown that, for n-point grid problems having compact stencils in $\mathbb{R}^3$, the required communication volume on a P-processor machine is bounded by $3n^{2/3}\log_2 P$, a clear gain over the $O(n)$ or $n\log_2 P$ cost incurred by most competing approaches. The performance of the method on ASCI-Red is illustrated in Fig. 4 for a (127×127) point Poisson problem ($n = 16129$) discretized by a standard 5-point stencil. Shown also are the times for the commonly used approaches of redundant LU solves and row-distributed A_c^{-1}.

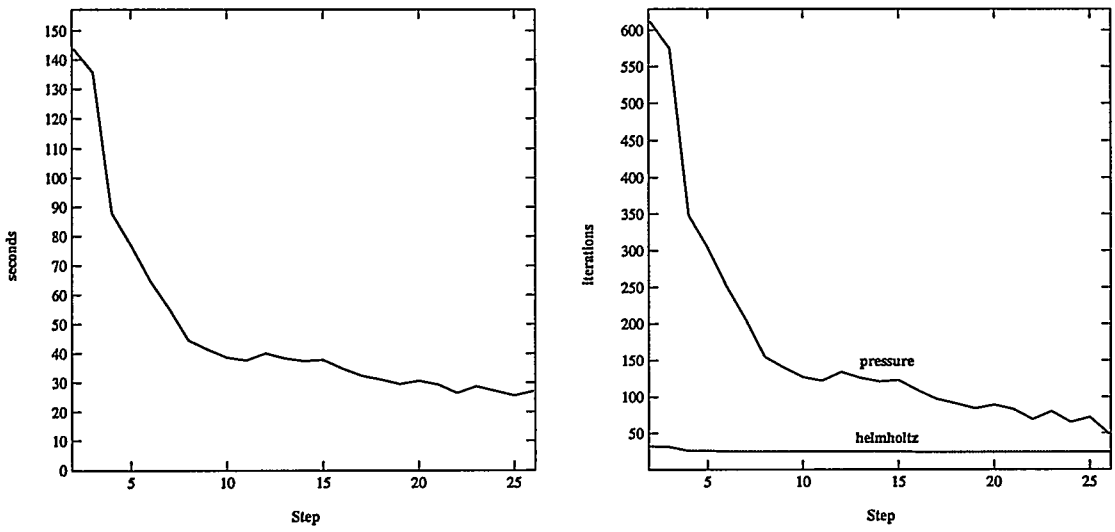

Figure 5: $P = 2048$ ASCI-Red-333 performance results for the first 25 time steps for $(K, N) = (8168, 15)$ corresponding to 27 M velocity points and 24 M pressure points: solution time-per-step (left) and number of pressure and Helmholtz iterations per-step (right).

Performance Results

We have run our spectral element code on a number of distributed-memory platforms, including the Paragon at Caltech, the T3E-600 at NASA Goddard, the Origin 2000 and SP at Argonne, ASCI-Blue at Los Alamos, and ASCI-Red at Sandia. For this paper we concentrate on recent timing results for the hairpin vortex problem of Fig. 1 that were obtained on ASCI-Red using up to 2048 nodes in single- and dual-processor mode.

Each node on ASCI-Red consist of two Zeon 333 MHz Pentium II processors which can be run in a message-passing (internode) / SMP (intranode) mode. Currently, only the matrix-vector product routine for the pressure operator, E, has been cast into dual-processor mode. The fact that we employ nonoverlapping storage for elements and have loops over the blocks exhibiting little or no data dependence implies that we should be able to exploit the second processor for additional work. Preliminary experiments indicate that a factor of 1.5 over the single-processor performance should be achievable on 2048 nodes.

The timing results presented are for the time stepping portion of the runs only. During production runs, usually 14 to 24 hours in length, our setup and I/O costs are typically in the range of 2–5%. To determine operation count we access the hardware operation counters via calls to the `perfmon` library. In addition, we have instrumented the code to provide various performance metrics, including per processor flop count. The two methods yield results within 2% of each other. Finally, all floating-point calculations were done in 64-bit precision.

Figure 5 shows time per step for the first 25 time steps (left) and the pressure and (x-component) Helmholtz iteration counts (right). The significant reduction in pressure iteration count is due to the difficulty of computing the initial transients and the benefits gained from the pressure projection procedure.

Table 3 presents results for a $K = 8168$ element case, of order $N = 15$ (28 M grid points for velocity, 22 M for pressure). The mesh was obtained via an oct-refine of the production mesh used for the transitional boundary layer/hemisphere calculation of Fig. 1. We note that the coarse grid for this problem has 10,142 distributed degrees-of-freedom and accounts for only 3.5% of the total solution time in the worst-case scenario of 2048 nodes in dual-processor mode.

Table 3: ASCI-Red-333: average time-per-step and GFLOPS, $N = 15$

	$K = 8168$, $N = 15$			
	single		dual	
P	time(s)	GFLOPS	time(s)	GFLOPS
512	247.1	45.9	191.7	59.2
1024	124.6	91.0	96.3	118.0
2048	64.1	176.9	51.8	219.0

Conclusion

We have developed a highly accurate spectral element code based on scalable solver technology that exhibits excellent parallel efficiency and sustains high MFLOPS. It attains exponential convergence, allows a convective CFL of 2–5, and has efficient multilevel elliptic solvers including a coarse grid solver that is communication minimal. The code currently runs on thousands of processors and is clearly ready to run on machines with tens of thousands of processors.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, and by the Department of Energy under Grant No. B341495 to the Center on Astrophysical Thermonuclear Flashes at University of Chicago. Computational time on the T3E-600 was provided with support from NASA microgravity grant NAG 32139 to the University of Maryland.

References

[1] M.S. ACALAR AND C.R. SMITH, "A study of hairpin vortices in a laminar boundary layer. Part 1. Hairpin vortices generated by a hemisphere protuberance", *J. Fluid Mech.*, **175**, pp. 1–41 (1987).

[2] J. BLAIR PEROT, "An analysis of the fractional step method", *J. Comput. Phys.*, **108**, pp. 51–58 (1993).

[3] M. DRYJA AND O.B. WIDLUND, "An additive variant of the Schwarz alternating method for the case of many subregions", *Tech. Rep. 339*, Dept. Comp. Sci., Courant Inst., NYU (1987).

[4] C. FARHAT AND P.S. CHEN, "Tailoring Domain Decomposition Methods for Efficient Parallel Coarse Grid Solution and for Systems with Many Right Hand Sides", *Contemporary Math.*, **180**, pp. 401–406 (1994).

[5] P.F. FISCHER, "Projection techniques for iterative solution of $A\underline{x} = \underline{b}$ with successive right-hand sides", *Comp. Meth. in Appl. Mech.*, **163** (1998).

[6] P.F. FISCHER, "Parallel multi-level solvers for spectral element methods", in *Proc. Intl. Conf. on Spectral and High-Order Methods '95, Houston, TX*, A.V. Ilin and L.R. Scott, eds., Houston J. Math., pp. 595-604 (1996).

[7] P.F. FISCHER, "An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations", *J. of Comp. Phys.*, **133**, pp. 84–101 (1997).

[8] P.F. FISCHER, N.I. MILLER, AND H.M. TUFO, "An overlapping Schwarz method for spectral element simulation of three-dimensional incompressible flows," *ANL/MCS Preprint* P730-1098.

[9] W.D. GROPP, "Parallel Computing and Domain Decomposition", in *Fifth Conf. on Domain Decomposition Methods for Partial Differential Equations*, T.F. Chan, D.E. Keyes, G.A. Meurant, J.S. Scroggs, and R.G. Voigt, eds., SIAM, Philadelphia, pp. 349–361 (1992).

[10] P.S. Klebanoff, W.G. Cleveland, and K.D. Tidstrom, "On the evolution of a turbulent boundary layer induced by a three-dimensional roughness element", *J. Fluid Mech.*, **92**, pp. 101-187 (1992).

[11] R.E. LYNCH, J.R. RICE, AND D.H. THOMAS, "Direct solution of partial difference equations by tensor product methods", *Numerische Mathematik*, **6**, pp. 185–199 (1964).

[12] Y. MADAY AND A.T. PATERA, "Spectral element methods for the Navier-Stokes equations", in *State of the Art Surveys in Computational Mechanics*, A.K. Noor, ed., ASME, New York, pp. 71–143 (1989).

[13] Y. MADAY, A.T. PATERA, AND E.M. RØNQUIST, "An operator-integration-factor splitting method for time-dependent problems: application to incompressible fluid flow", *J. Sci. Comput.*, 5(4), pp. 263–292 (1990).

[14] M.R. MALIK, T.A. ZANG, AND M.Y. HUSSAINI, "A spectral collocation method for the Navier-Stokes equations," *J. Comput. Phys.*, **61** (1985) pp. 64-88.

[15] A.T. PATERA, "A spectral element method for fluid dynamics: laminar flow in a channel expansion", *J. Comput. Phys.*, **54**, pp. 468–488 (1984).

[16] A. POTHEN, H.D. SIMON, AND K.P. LIOU, "Partitioning sparse matrices with eigenvectors of graphs", *SIAM J. Matrix Anal. Appl.*, **11** 3 (1990) pp. 430-452.

[17] H.M. TUFO AND P.F. FISCHER, "Fast parallel direct solvers for coarse grid problems", *J. Dist. Par. Comp.* (accepted).

[18] H.M. TUFO, P.F. FISCHER, M.E. PAPKA, AND M. SZYMANSKI, "Hairpin vortex formation, a case study for unsteady visualization", 41st CUG Conference, 1999.

[19] H.M. TUFO, "Algorithms for Large-Scale Parallel Simulation of Unsteady Incompressible Flows in Three-Dimensional Complex Geometries", Ph.D. Thesis, Brown University (1998).