JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 1, June 2004

Multilevel Parallelization Models in CFD

Suchuan Dong* and George Em Karniadakis†
Brown University, Providence, Rhode Island 02912

We present two multilevel parallel models based on MPI/MPI (MPI denoting Message Passing Interface) and MPI/OpenMP (OpenMP denoting Open Multi-Processing) for high-order CFD methods and compare their performances. These models are implemented within the spectral/hp element framework to take advantage of the hierarchical structures arising from deterministic and stochastic CFD computations. For MPI/MPI, we employ MPI process groups to decompose the computations into different levels. For MPI/OpenMP, we take a Single-Program-Multiple-Data (SPMD) style approach to OpenMP shared memory parallelism that significantly reduces the OpenMP synchronizations. These models demonstrate a good scalability with respect to the problem size and a good speedup for fixed problem sizes. With identical configurations, the MPI/MPI parallel model is observed to be generally more efficient. The advantage of these multilevel approaches lies in that they reduce the number of processes participating in each communication and the latency overhead, and thus enable the applications to scale to a large number of processors more efficiently. The models have been applied to the direct simulation of turbulent flows past a circular cylinder at Reynolds number Re=10,000.

I. Multilevel Parallelism

MULTILEVEL parallelism has been motivated by the performance limitations demonstrated in single-level parallel computations that prevent effective scaling to a large number of processors on modern supercomputers.[1] It can potentially exploit the hierarchical structures inherent in modern high-performance computer architectures and a range of applications more effectively.

Most modern high-performance computers utilize distributed memory architecture for scalable performance. However, manufacturers often incorporate shared-memory parallelism at the node level to address issues of cost-effective packaging and power. As a result, most platforms, including the top ten supercomputers in the world (of the present time), are essentially clusters of shared-memory multiprocessors (SMP). A challenge presented to application developers is the hierarchical parallelism with increasingly complex non-uniform memory access exhibited by such machines. Flat message-passing paradigm has been the dominant model on these architectures. However, a multilevel parallelization approach.[2-4] through pure message passing or by combining message-passing and shared memory parallelism would be a more natural alternative for these architectures.[5]

Hybrid parallelism employing MPI/threads has been studied for kernel calculations,[6] simplified model problems[7] and more complex applications,[5,8,9] with both loop- and subroutine-level shared memory parallelism,[7,9,10] and coarse grain shared memory parallelism.[5,8,11] These studies indicate that the thread management overhead is an important factor affecting the performance of hybrid applications. The employment of a large number of threads within the SMP node also impacts negatively the performance due to the memory bandwidth contention amongst multiple threads.[5]

Multilevel parallelism using MPI was previously applied to optimization problems.[1,3] In the current work, we employ an MPI multilevel parallelization approach to high-order CFD methods within the spectral/hp element framework, and compare its performance with MPI/OpenMP hybrid parallelism.[5] Numerical simulations employing high-order methods can particularly benefit from multilevel parallelism as increasing the problem size by using

Received 3 February 2004; revision received 1 May 2004; accepted for publication 3 May 2004.
*Research Associate, Division of Applied Mathematics, 182 George Street.
†Professor, Division of Applied Mathematics, 182 George Street; gk@dam.brown.edu.

higher spectral orders (P-type refinement) will localize the work which can then be computed efficiently at a different level.

II. Hierarchical Structures in High-Order CFD Computations

High-order deterministic and stochastic CFD computations demonstrate inherent hierarchical structures when the problem is discretized with the spectral/hp element method.[12]

Specifically, hierarchical structures arise from the stochastic CFD computations using generalized polynomial chaos.[13] The key idea of polynomial chaos is to represent stochasticity spectrally with polynomial functionals, first introduced by Wiener for Gaussian random processes. The randomness is absorbed by a suitable orthogonal basis function from the Askey family of polynomials.[13] Subsequently, the Navier-Stokes equation is projected onto the space spanned by the same orthogonal polynomial functions, and a set of deterministic differential equations results. As a result, the Navier-Stokes equations are reduced to a set of equations for the expansion coefficients (called random modes), which are three-dimensional deterministic functions of both space and time. The random modes are de-coupled from one another (except in the non-linear terms) and are solved with the spectral/hp element method.[12] Computations of the random modes, sub-domains of each random mode, spectral elements within the sub-domain, and at the sub-element level form the hierarchy of operations in the stochastic CFD computations.

Deterministic CFD computations for Vortex-Induced Vibrations (VIV) demonstrate similar hierarchical structures, for example, the flow past a flexible cylinder subject to VIV (Refs. 14,15). For the VIV problem the flow velocity is represented by

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

This representation applies to three-dimensional unsteady flow problems on geometries with one homogeneous direction while the non-homogeneous two-dimensional domain is arbitrarily complex. A combined spectral element-Fourier discretization[12] can be employed to accommodate the requirements of high-order as well as the efficient handling of multiply-connected computational domain in the non-homogeneous planes. Spectral expansions in the homogeneous direction involve Fourier modes that are decoupled from one another (except in nonlinear terms) and can be solved with the spectral element approach. Computations of the Fourier modes, spectral element plane, spectral elements within the plane, and at the sub-element level form the hierarchy of operations in the solution process of VIV simulations.

The *inherent* hierarchical structures in high-order CFD computations suggest a multi-level strategy to parallelization. At the top-most level are groups of MPI processes. Each group computes one or more random mode. At the next level, the three-dimensional domain of each random mode is decomposed into sub-domains, each consisting of spectral elements. Each MPI process within the group computes one sub-domain. At the third level, multiple threads are employed to share the computations within the sub-domain (or MPI process). Compared with the flat message-passing model on the same number of processors, this multilevel parallelization strategy reduces the network latency overhead because a greatly reduced number of processes are involved in the communications at each level. This enables the applications to scale to a large number of processors more efficiently.

To exploit the hierarchical structures arising from spectral/hp element CFD computations, we developed a hybrid parallelism with MPI/OpenMP.[5] The main idea is to use MPI for domain decomposition in the homogeneous direction and use OpenMP threads for the computations in the non-homogeneous spectral element planes. The hybrid MPI/OpenMP approach has been shown to perform better than both MPI (single-level) and OpenMP on SGI Origin and Intel IA64 platforms.[5] In this paper we present an MPI/MPI two-level parallelization approach for the spectral/hp element method. We employ MPI for domain decomposition in the homogeneous direction (first level). In the non-homogeneous spectral element planes (second level), we parallelize the computations further employing MPI in conjunction with a graph-partitioning algorithm.[16] This MPI/MPI two-level parallelization model, together with the MPI/OpenMP hybrid approach, not only eliminates the performance restrictions in the single-level MPI computations,[17-19] but also improves the efficiency in exploiting a large number of processors. The objectives of this paper are to present these two multilevel parallelization paradigms, investigate their influence on p-type refinement in high-order methods, and compare their performance.

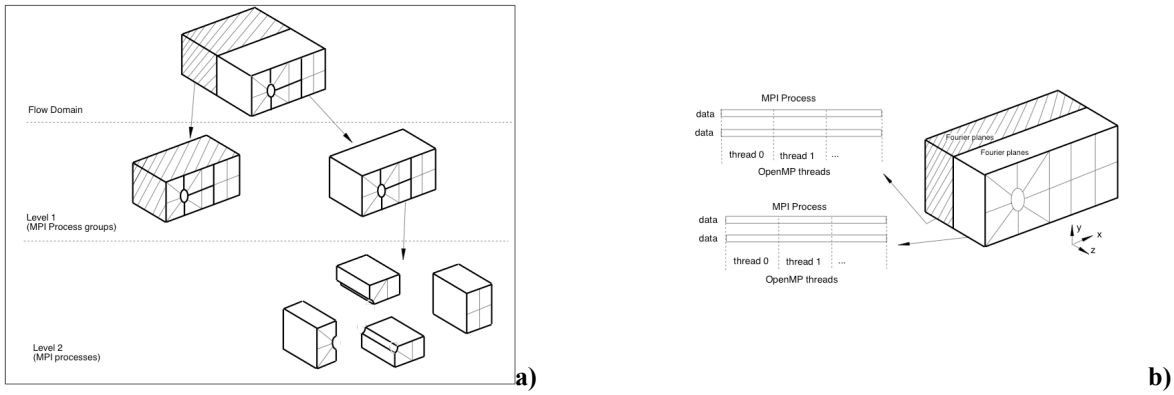
III. MPI/MPI Two-Level Parallelism

Figure 1a provides a schematic for the MPI/MPI two-level parallel paradigm. The flow domain is decomposed along the homogeneous direction first. Each sub-domain consists of two or more non-homogeneous spectral element

planes, corresponding to one or more Fourier modes. At the top level are groups of MPI processes. Each group computes one sub-domain in the homogeneous (Fourier) direction. The non-homogeneous spectral element planes are further decomposed into sub-domains at the second level. Each of these sub-domains comprises a number of structured or un-structured spectral elements. Correspondingly, each MPI process within the group computes one sub-domain at the second level.

We employ MPI communicators/groups to map the flow sub-domains onto MPI processes. Consider the configuration of N_z flow sub-domains in the homogeneous direction at the first level and N_{xy} sub-domains in spectral element planes at the second level. The initial communicator is split into two sets of disjoint process sub-groups/communicators. In the first set, the partition of MPI processes is along the homogeneous direction, and correspondingly N_z disjoint sub-groups/communicators are created. Each of these communicators contains N_{xy} MPI processes, and provides the context for communications in the spectral element planes (second level). In the second set, the partition of MPI processes is in the spectral element planes, and correspondingly N_{xy} disjoint sub-groups/communicators are created. Each of them contains N_z MPI processes, and provides the context for communications in the homogeneous direction (first level).

Distinct communication patterns appear at these two levels, and dominate different stages of the computation. The dominant pattern at the first level is all to all communication for transposing the distributed matrices[18] when an FFT is evaluated. These operations occur when the non-linear terms and the velocity divergence are computed in the Navier-Stokes equations. At the second level, the reduction operations dominate the communications for evaluating the inner products in the conjugate gradient iterative solver. These occur in the Poisson solve for the pressure and the Helmholtz solve for the velocity. The communications on these two levels take place at different stages during the computation, and alternate as the simulation marches in time. A high-order stiffly stable time integration scheme is employed[20] consisting of three stages: 1) non-linear term evaluation, 2) pressure solve, and 3) viscous solve. The communications at the first level occur at stage 1 in evaluating the non-linear terms and computing and at stage 2 in calculating the right-hand-side of the Poisson's equation for the pressure. The communications at the second level occur at stages 2 and 3 in the iterative solutions of the Poisson's equation for pressure and of the Helmholtz equation for velocity.



a) **b)**

Fig. 1 Schematic showing a) MPI/MPI two-level parallelism and b) MPI/Open MP hybrid parallelism.

IV. MPI/OpenMP Hybrid Parallelism

Figure 1b provides a schematic for the MPI/OpenMP hybrid paradigm. The flow domain is again decomposed along the homogeneous z-direction. At the outer level multiple MPI processes are utilized with each process computing one sub-domain. At the inner level, within each sub-domain multiple OpenMP threads conduct the computations in parallel. Data exchange across sub-domains is implemented with MPI. Within the sub-domain, access to shared objects by multiple threads is coordinated with OpenMP synchronizations. We take an SPMD-style approach to OpenMP shared memory parallelism that greatly reduces the OpenMP barriers.[5]

Specifically, a single parallel region is placed at the topmost level. This avoids the overhead associated with frequent thread creations and destructions inherent in fine grain computations. OpenMP threads work on disjoint groups of elements or disjoint sections of the vectors (of roughly equal size). The vector length, the element number, and the number of entries in the linked list are split based on the number of threads. This computation is done only once at the pre-processing stage, and the results are stored in a shared table. This configuration avoids the synchronization overhead associated with dynamic scheduling. Working on a large section of a vector with contiguous memory rather than a strided one (as with dynamic scheduling) improves the cache-hit rate. The MPI

calls are handled by only one thread within each process. Advantageous over single-level pure MPI programs on SMP nodes, this configuration assembles the nodal messages into a single one and thus reduces the network latency overhead. Barriers are the main OpenMP synchronizations. The majority of OpenMP barriers occur at the switching points between global and local operations. We have developed a consistent workload-splitting scheme across local and global operations that eliminate the majority of OpenMP barriers.[5]

V. Simulation Results

We apply the MPI/MPI parallel model to simulate the turbulent flow past a long stationary circular cylinder at the Reynolds number Re = 10,000 based on the inflow velocity and the cylinder diameter. A "z-slice" of the computational domain in the x - y plane consists of 6272 triangular spectral elements (Fig. 2). In the homogeneous z-direction a maximum of 64 Fourier modes (or 128 spectral element planes) are employed. The flow domain extends from –20D (where D is cylinder diameter) at the inlet and to 50D at the outlet, and from –20D to 20D in the cross-flow direction. The spanwise length of the domain is fixed at $L_z/D = \pi$. A uniform flow is prescribed at the inlet. Neumann boundary conditions are applied at the outlet. Periodic boundary conditions are used in the cross-flow direction as well as in the homogeneous direction.

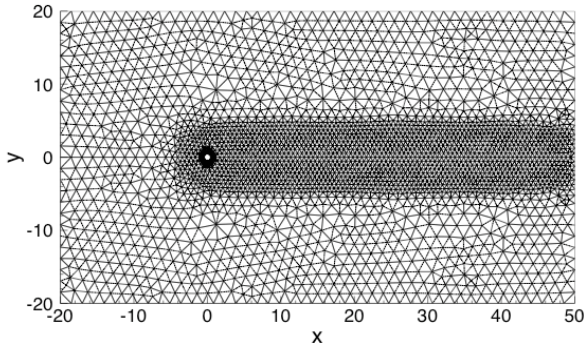


Fig. 2 Mesh in the x – y plane with 6272 triangular elements in the simulation.

In the MPI/MPI two-level parallel simulations we divide MPI processes into a number of groups based on the number of Fourier modes in homogeneous direction (first level) such that each group computes one Fourier mode. At the second level, we deploy 8 MPI processes in each group for the current problem size. Correspondingly, the mesh in the spectral element planes is partitioned into 8 sub-domains, and each MPI process in the group computes one partition of the mesh.

We vary the grid resolution by changing the number of Fourier modes in the homogeneous direction and the order of spectral elements in the non-homogeneous planes. Table 1 lists the wall timing for various problem sizes collected on the Compaq Alpha Cluster at PSC. As the problem size increases only a slight increase in wall time is observed as the number of processors is increased in proportion, indicating a good scaling of the MPI/MPI parallel model with respect to problem sizes. Table 2 lists the drag coefficient, Strouhal number and the base pressure coefficient from the simulations, together with their experimental values. The computed values are in good agreement with the measured values from the experiments. Figure 3 shows the signals of the drag and lift coefficients on the cylinder with 64 Fourier modes in the homogeneous direction and 5-th order spectral elements in the spectral element planes. In Fig. 4 we plot the contours of the streamwise mean velocity (top row) and rms velocity (bottom row) from the simulations (left column) and from the PIV experiment by Salim and Rockwell[21] (right column). The simulation has produced the same distributions for the mean and rms velocities as the experiment.

Table 1 Wall timing vs problem sizes for cylinder flow at Re = 10,000 on the Compaq Alpha Cluster at PSC. "DOF" denotes the total degrees of freedom of the system.

Fourier Modes	Elements	Spectral Order	DOF (million)	Processors	Wall time/step (s)
2	6272	5	2.1	16	0.83
8	6272	5	8.5	64	1.04
16	6272	5	17	128	1.15
32	6272	5	33.9	256	1.21
64	6272	5	67.7	512	1.54

Table 2 Flow quantities of turbulent flow past a cylinder at Re = 10,000: C_D, drag coefficient; St, Strouhal number; $-C_{P_b}$, base pressure coefficient; P, spectral element order; M: number of Fourier modes.

	C_D	St	$-C_{P_b}$
DNS (P=5, M=8)	1.155	0.195	1.129
DNS (P=5, M=32)	1.110	0.209	1.084
DNS (P=5, M=64)	1.128	0.205	1.171
Bishop and Hassan[22]	---	0.201	---
Gopalkrishnan[23]	1.186	0.193	---
Williamson[24]	---	---	1.112
Norberg[25]	---	0.202	---

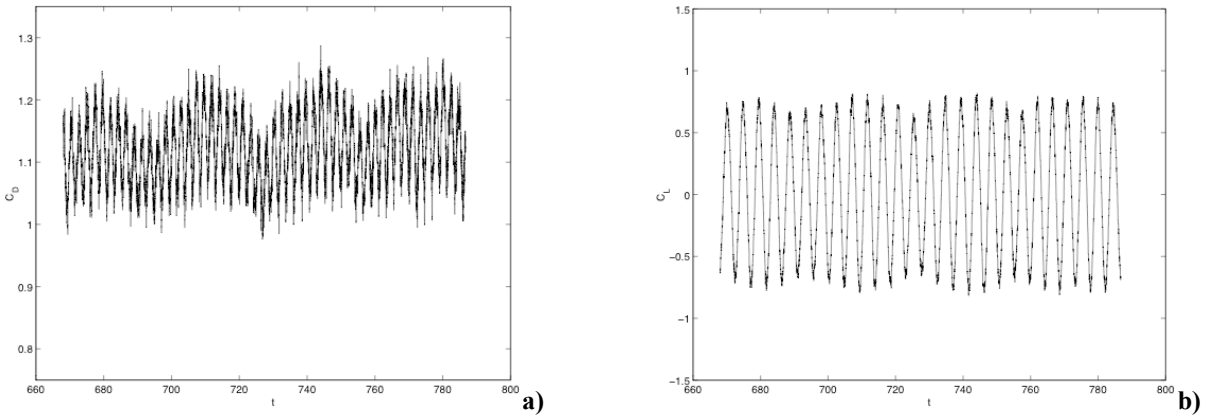


Fig. 3 Cylinder flow at Re = 10,000: time history of a) drag coefficient and b) lift coefficient.

VI. Performance Results

We next examine the performance of the MPI/MPI model more systematically with a three-dimensional turbulent flow past a cylinder at Reynolds number Re = 500 based on the inflow velocity and the cylinder diameter. We choose this Reynolds number and the other flow parameters in accordance with Ref. 5 for the purpose of comparison with MPI/OpenMP hybrid parallelization. We employ 16 Fourier modes (i.e. 32 spectral element planes) in z-direction, and a mesh of 412 triangular elements in each spectral element x - y plane.

Two groups of tests are considered on three platforms: Intel IA64 Cluster at NCSA (800MHz Itanium), IBM SP3 at SDSC (375 MHz Power3) and Compaq Alpha Cluster at PSC (1GHz EV68). The first group is to examine the scaling with respect to the total number of processors for a fixed problem size. A fixed spectral polynomial order of N_{order} is used. The number of MPI processes or process groups in the homogeneous direction is varied from 1 through 16. The number of processors in the spectral element plane (for MPI/MPI) is varied between 1 and 4 on SP3 and Compaq machine and between 1 and 2 on IA64 cluster. The number of OpenMP threads per process (for MPI/OpenMP) is varied in a similar fashion. Table 3 summarizes the configurations of the tested cases on all three platforms. The second test group is to check the scalability with respect to the problem size. The same mesh as in the

first group is used while the polynomial order and the number of processors in the spectral element plane (for MPI/MPI) or the number of threads per process (for MPI/OpenMP) are varied.

Table 3 Configurations of tested cases. Table shows the platforms of a configuration (number of processors at first and second levels) is tested on. SDSC SP3, PSC Alpha, and NCSA IA64 are denoted by the letters S, A, and I respectively.

		First Level (MPI processes or process groups)				
		1	2	4	8	16
Second Level (MPI processes or OpenMP threads)	1	S/A/I	S/A/I	S/A/I	S/A/I	S/A/I
	2	S/A/I	---	---	---	S/A/I
	4	S/A	---	---	---	S/A

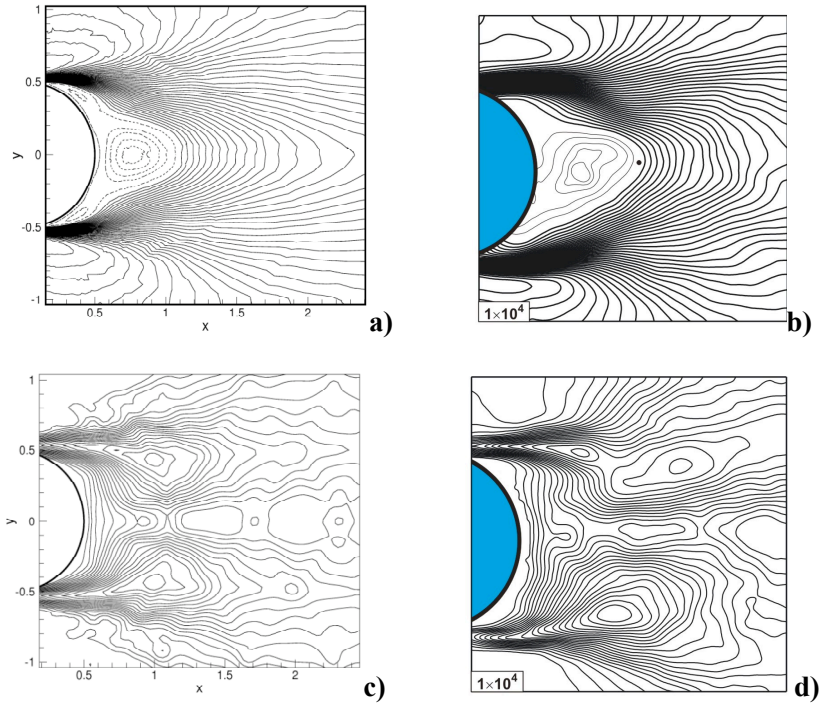


Fig. 4 Comparison between simulation and PIV (Ref. 21) of cylinder flow at Re = 10,000: a) mean streamwise velocity from simulation and b) from PIV; c) streamwise rms velocity from simulation and d) from PIV. PIV plots courtesy of D. Rockwell (Ref. 21).

A. Test Group One: Fixed Problem Size

In Fig. 5 we compare the performance of MPI/MPI parallelism and MPI/OpenMP hybrid parallelism on the IA64 cluster at NCSA. Three configurations are tested for each model. For MPI/MPI, in the first configuration (denoted "Z-Decomposition") we decompose the flow domain only at the first level (in z-direction), with no decomposition in the spectral element x - y plane. In the second configuration (denoted "XY-Decomposition"), we decompose the spectral element x - y planes into sub-domains at the second level, with no decomposition in the homogeneous direction. In the third configuration (denoted "Mixed Configuration"), the flow domain is decomposed at both the first and the second levels. The first configuration corresponds to the original single-level MPI parallelization in which only one processor computes in the spectral element planes. With the second and third configurations multiple processors share the computations in the spectral element planes. For MPI/OpenMP we test three similar configurations. In the first configuration ("pure MPI") the flow domain is decomposed in homogeneous z direction, while only one OpenMP thread per MPI process computes in each sub-domain. In the second configuration ("pure OpenMP"), no domain decomposition is performed (only one MPI process), while multiple OpenMP threads share the computations. In the third configuration ("hybrid"), multiple MPI processes are employed at the first level and multiple OpenMP threads per process are deployed to split the workload in the spectral element planes.

Figure 5a shows the wall clock time per step for the three configurations as a function of the total number of processors for MPI/MPI, and the speedup factor, S_p, is shown in Fig. 5c. As the number of partitions increases at the first level we observe a near-linear speedup, indicating that this configuration is an efficient decomposition algorithm. The disadvantage of the first configuration is that the number of Fourier modes imposes an upper bound on the number of partitions (and hence the number of processors) in the homogeneous direction. Domain decomposition only in the non-homogeneous spectral element planes (second configuration) produces better wall clock timings and a super linear speedup for up to 8 processors, pointing to the effect of increased aggregate cache size and improved cache reuse. The scaling deteriorates as the number of partitions in the spectral element planes increases further to 16. This is attributed to the relatively small problem size and the load imbalance in different sub-domains. The load imbalance problem does not exist for the domain decomposition in the homogeneous direction. In the spectral element planes the mesh partition can produce load imbalance across sub-domains, although this problem is minimized with METIS.[16] The impact of load imbalance becomes greater as the number of sub-domains increases and the size of each sub-domain decreases. In the third configuration, we decompose the flow domain to the maximum extent at the first level (i.e., 16 for current problem) and vary the number of sub-domains at the second level. As the number of sub-domains in the spectral element planes increases, a super-linear speedup is observed, which is due to the cache effect.

Figure 5b shows the wall timings for MPI/OpenMP hybrid parallelism. The corresponding parallel speedup factors are plotted in Fig. 5d. Pure MPI configuration demonstrates a near-linear speedup. The pure OpenMP run with two threads shows a super-linear speedup, with a wall-clock time lower than the pure MPI run with two processes. The hybrid run (with 16 MPI processes and 2 threads per process) demonstrates a scaling comparable to pure MPI. Comparison of the wall clock timings between MPI/MPI and MPI/OpenMP indicates that the former is slightly better than the latter.

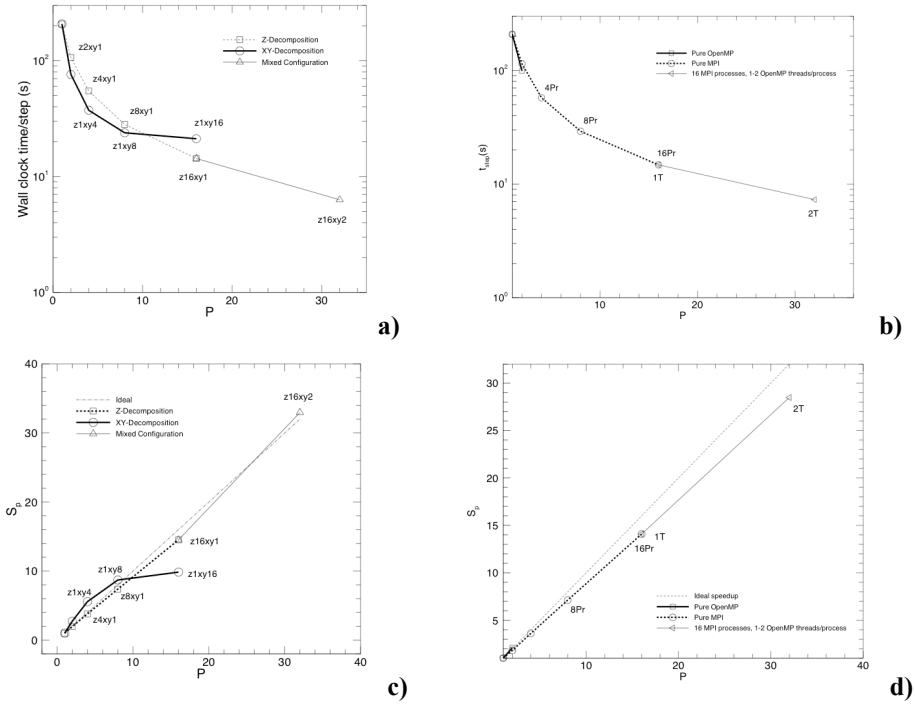


Fig. 5 IA64 Cluster (NCSA): Wall time/step with a) MPI/MPI and b) MPI/Open MP models vs processors; and parallel speedup with a) MPI/MPI and d) MPI/Open MPI models vs processors. Symbols: "z16xy2": 16 sub-domains in *z*-direction and 2 sub-domains in *x*-*y* planes; "2T": 2 Open MP threads per MPI process; "8Pr": 8 MPI processes.

Figure 6 shows the wall time (top row) and the speed-up factor (bottom row) with respect to the number of processors for MPI/MPI (left column) and MPI/OpenMP (right column) on the IBM SP3 at SDSC. Domain decomposition at the first level results in a linear scaling almost comparable to the ideal speedup. Partition of the flow domain at the second level produces a scaling slightly better than that at the first level. In the third configuration with 16 sub-domains in the homogeneous direction, increasing the number of partitions in the spectral

element planes to 2 results in a linear speedup. Increasing the number of partitions further to 4 results in a speedup factor that slightly deteriorates. Figures 6b and 6d show the wall timings and speedup factors for MPI/OpenMP. Pure MPI runs produce a scaling the same as the ideal one. The pure OpenMP run shows a performance inferior to the corresponding MPI run on the same number of processors. The hybrid runs demonstrate a good speedup as the number of threads per process increases. Comparison of the corresponding MPI/MPI and MPI/OpenMP runs again indicates that the MPI/MPI model results in a lower wall clock time and a higher parallel speedup.

In Fig. 7 we plot the wall clock time (top row) and parallel speedup (bottom row) with respect to the number of processors for MPI/MPI (left column) and MPI/OpenMP (right column) on the Compaq Alpha cluster at PSC. We observe a trend similar to that on the other platforms. For both models a linear speedup is observed as the number of processors at the first level. For MPI/MPI, in the second configuration as the number of processors in the spectral element planes increases a super-linear speedup is observed for up to 4 processors, which is due to increased aggregate cache size. The third configuration also results a good scaling. For MPI/OpenMP, the pure OpenMP run does not scale quite well while the pure MPI run and the hybrid run demonstrate a good scalability. Comparing the wall clock timings for these two models for identical configurations indicates that MPI/MPI runs demonstrate a superior performance to the MPI/OpenMP runs for this problem.

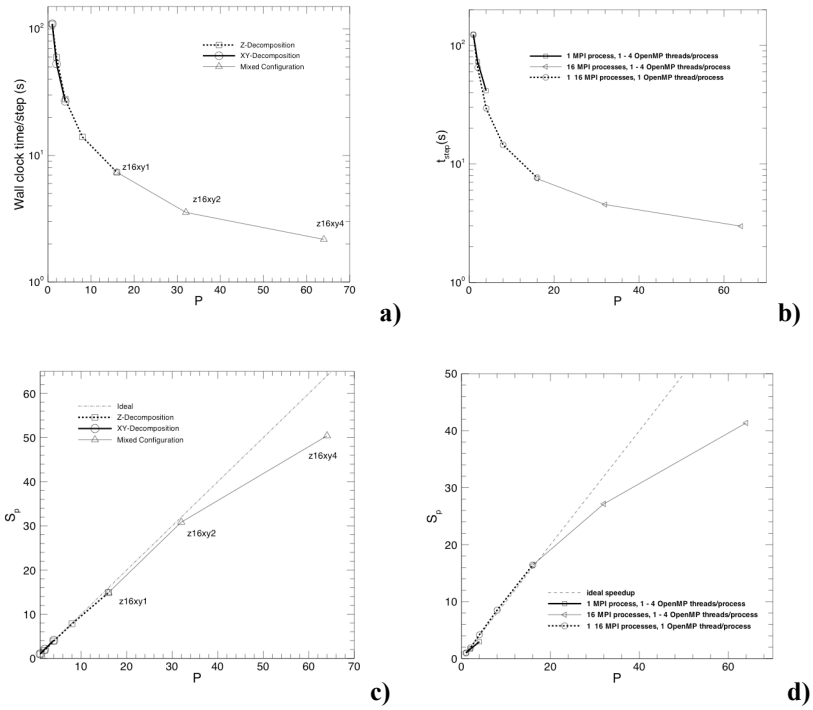


Fig. 6 IBM SP3 (SDSC): Wall time/step with a) MPI/MPI and b) MPI/Open MP models vs processors; and parallel speedup with a) MPI/MPI and d) MPI/Open MPI models vs processors. Symbols: "z16xy2": 16 sub-domains in *z*-direction and 2 sub-domains in *x-y* planes; "2T": 2 Open MP threads per MPI process; "8Pr": 8 MPI processes.

B. Test Group Two: Variable Problem Size

Next we examine the scaling of these two models with respect to the problem size. We concentrate on the scaling with respect to grid refinement in the non-homogeneous x - y plane through p-type refinement.

We use the same mesh as in test group one, while the order of elements is varied. The number of Fourier modes in the homogeneous z-direction is fixed at 16, and correspondingly 16 MPI processes or process groups are employed in the homogeneous direction for all the following cases. Three different problem sizes are tested corresponding to the polynomial orders of 7, 10, and 13. For MPI/MPI, we first fix the number of processors in the spectral element planes (second level) to one, and collect the wall timing results for all the problem sizes. Then, as element order increases we increase the number of processors at the second level approximately in proportion to the cost increase for the one-processor cases. For MPI/OpenMP, we first deploy one OpenMP thread per MPI process and collect the timing data for all the orders. Then, as the problem size increases we increase the number of threads per process approximately in proportion to the cost increase for the single-thread cases. Corresponding to the

polynomial orders 7, 10 and 13 we employ 1, 2 and 4 processors in the spectral element planes for MPI/MPI and 1, 2 and 4 threads per process for MPI/OpenMP. The results for these cases are shown in Figs. 8, 9, and 10 for the IA64 cluster, IBM SP3 and Compaq Alpha cluster, respectively. As the element order increases, the execution time increases significantly for the runs with one processor at the second level (MPI/MPI) or with one thread per process (MPI/OpenMP). When the number processors at the second level (for MPI/MPI) or the number of threads per process (for MPI/OpenMP) is increased in proportion, the execution time increases only slightly as element order increases, indicating that both parallelization models demonstrate a good scaling with respect to the problem size on the three platforms.

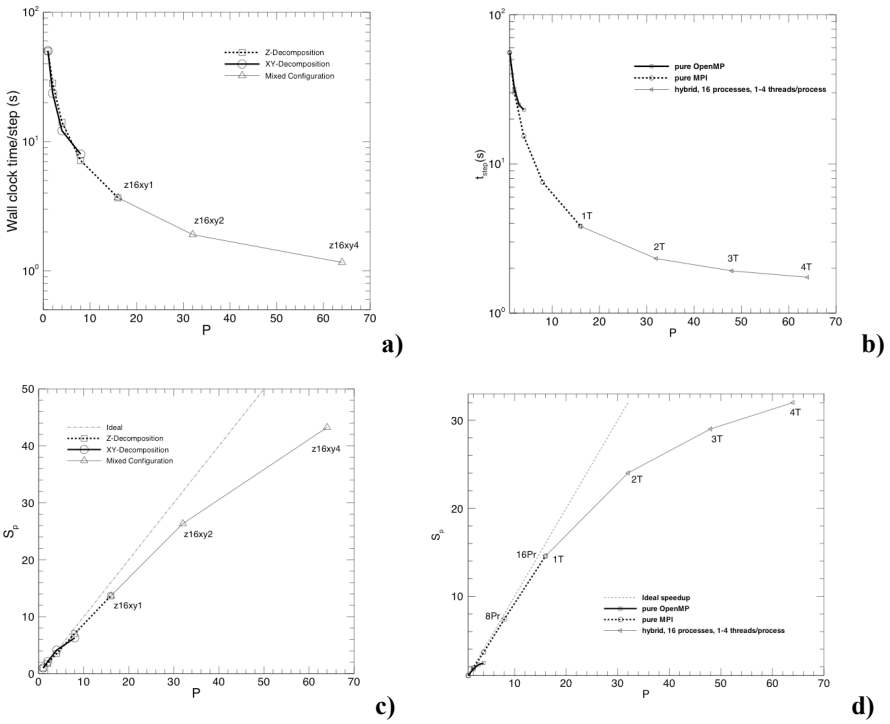


Fig. 7 Compaq Alpha (PSC): Wall time/step with a) MPI/MPI and b) MPI/Open MP models vs processors; and parallel speedup with a) MPI/MPI and d) MPI/Open MPI models vs processors. Symbols: "z16xy2": 16 sub-domains in *z*-direction and 2 sub-domains in *x*-*y* planes; "2T": 2 Open MP threads per MPI process; "8Pr": 8 MPI processes.

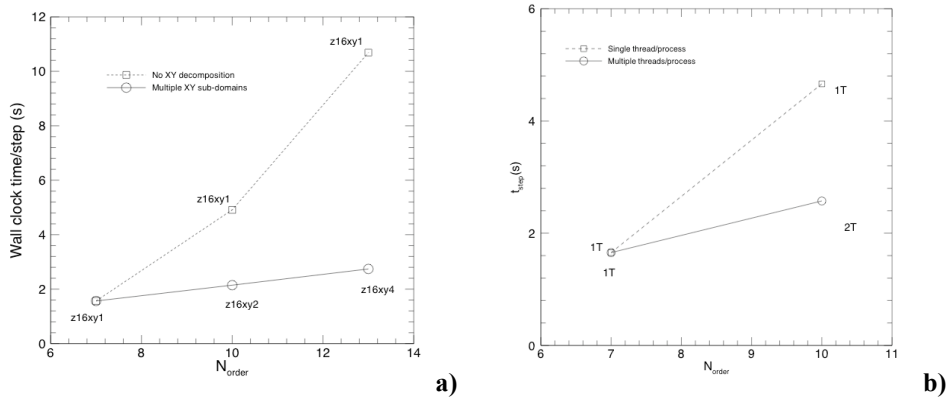
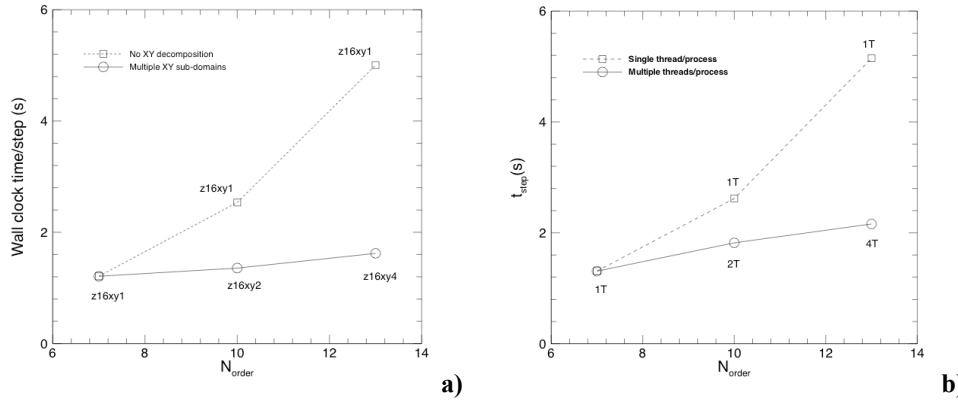
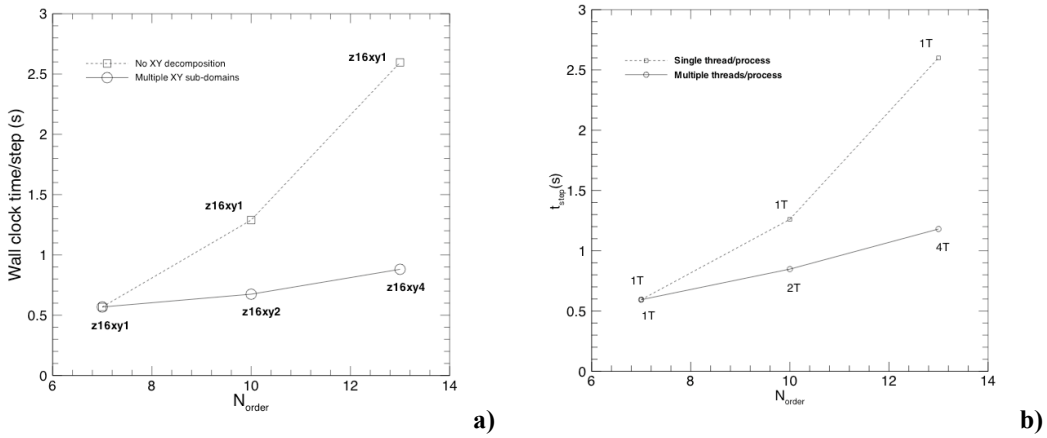


Fig. 8 IA64 Cluster (NCSA): Wall time/step vs spectral element order for a) MPI/MPI and b) MPI/Open MP. Dashed line: 1 processor at the second level; Solid line: multiple processors at the second level. Symbols: "z16xy2": 16 sub-domains in *z*-direction and 2 sub-domains in *x*-*y* planes; "2T": 2 Open MP threads per MPI process; "8Pr": 8 MPI processes.



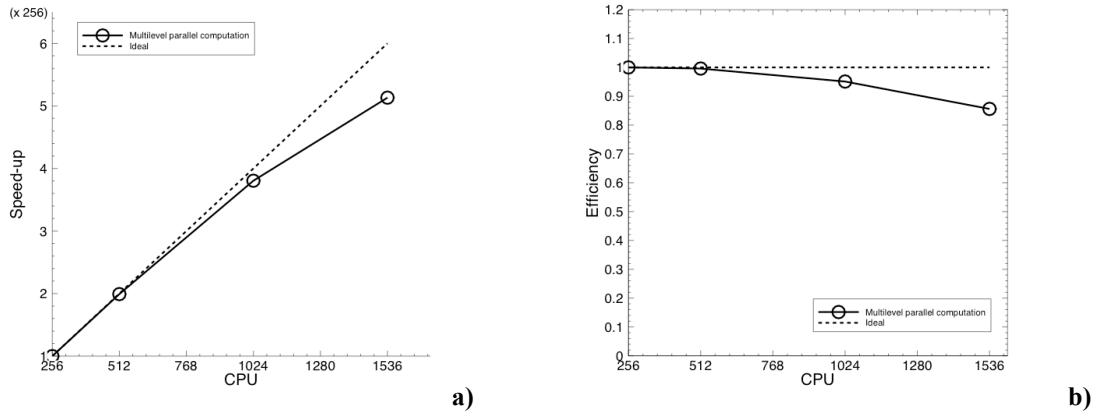
a) b)

Fig. 9 IBM SP3 (SDSC): Wall time/step vs spectral element order for a) MPI/MPI and b) MPI/Open MP. Dashed line: 1 processor at the second level; Solid line: multiple processors at the second level. Symbols: "*z*16xy2": 16 sub-domains in *z*-direction and 2 sub-domains in *x*-*y* planes; "2T": 2 Open MP threads per MPI process; "8Pr": 8 MPI processes.



a) b)

Fig. 10 Compaq Alpha (PSC): Wall time/step vs spectral element order for a) MPI/MPI and b) MPI/Open MP. Dashed line: 1 processor at the second level; Solid line: multiple processors at the second level. Symbols: "*z*16xy2": 16 sub-domains in *z*-direction and 2 sub-domains in *x*-*y* planes; "2T": 2 Open MP threads per MPI process; "8Pr": 8 MPI processes.



a) b)

Fig. 11 a) Parallel speedup and b) parallel efficiency of the MPI/MPI model vs the number of processors on PSC Compaq Alpha cluster for the flow past a cylinder at Reynolds number Re = 10,000 with a fixed problem size of 300,000,000 degrees of freedom. Speedup is calculated based on the wall time on 256 processors.

VII. Summary and Conclusions

Multilevel parallelism can particularly benefit high-order numerical methods as increasing the problem size by using higher spectral orders (P-refinement) will localize the work which can then be computed efficiently at a different level. In this paper we have presented a new MPI/MPI two-level parallelization model for high-order numerical methods within the spectral/hp element framework. Both MPI/MPI and MPI/OpenMP models can effectively take advantage of the hierarchical structures inherent in high-order CFD computations. The simulation results, obtained with the multilevel parallel algorithms for the flow past a cylinder at Re = 10,000, agree very well with the experiments. We highlight the most important points from this study as follows:

1) The advantage of these multilevel parallel paradigms lies in that they facilitate the reduction of the number of processes (or threads) participating in communications. In a single-level parallelism, for global reduction operations, which is often encountered in iterative linear-equation solvers, and all-to-all operations, which is often encountered in FFT-based applications, all processes are involved in the communications. In contrast, in the multilevel approach processes (or threads) participate in communications at different levels; they communicate with the other processes (or threads) at the same level. As a result, a greatly reduced number of processes are involved in each communication. This reduces the communication latency overhead and enables the applications to scale to a large number of processors more efficiently. We take the FFT computation of the three-dimensional flow data in the MPI/MPI model and in a single-level MPI computation, in which the domain is decomposed with respect to the Fourier modes, on the same number of processors to illustrate the above point. In MPI/MPI we employ N_z groups (first level) with N_{xy} MPI processes (second level) in each group. The total number of processors in this computation is $N_z \cdot N_{xy}$. Accordingly, in the single-level MPI computation we employ the same number of processors $N_z \cdot N_{xy}$. Therefore, when the FFT is evaluated, in MPI/MPI N_z processors are involved in each all-to-all communication while N_{xy} independent all-to-all communications (each involving different sets of N_z processors) proceed simultaneously (overlap in time). In contrast, in the single-level MPI computation all processors are involved in the all-to-all communication. The MPI/MPI model effectively replaces a single all-to-all communication involving $N_z \cdot N_{xy}$ processors in a single-level MPI computation with N_{xy} simultaneous and independent all-to-all communications with each involving N_z processors. Note that these N_{xy} different all-to-all communications involve different sets of processors and proceed simultaneously in time. To demonstrate the extremely high scalability these multilevel models have achieved, in Fig. 11 we plot the parallel speedup (a) and the parallel efficiency (b) of the MPI/MPI model as a function of the number of processors on the Compaq Alpha cluster at PSC for a fixed problem size with 300,000,000 degrees of freedom (turbulent flow past cylinder at Reynolds number Re = 10,000). This multilevel model has achieved over 95% parallel efficiency on 1024 processors and over 85% parallel efficiency on 1536 processors.

2) For identical configurations, MPI/MPI model demonstrates a slightly superior performance compared with MPI/OpenMP in current implementations. This performance difference is attributed to several factors. First, the performances of the MPI and OpenMP libraries play an important role. MPI implementations have become very mature and demonstrated very high performances on almost all platforms; OpenMP, on the other hand, is relatively new, and there is room for performance improvement on many platforms. Other unfavorable factors for OpenMP are the thread management overhead, which tends to increase significantly as the number of threads increases, and the memory bandwidth contention among threads since they run on the same node (see Ref. 5 for detailed discussions on the influence of these factors on OpenMP).

Table 4 Wall time/step of MPI/MPI and MPI/OpenMP models on 32 processors for the cylinder flow problem in Sec. VI.A. First level: 16 processors; Second level: 2 processors.

	MPI/MPI Wall time/step (s)	MPI/OpenMP Wall time/step (s)
IA64 (NCSA)	6.31	7.29
IBM SP3 (SDSC)	3.55	4.54
Compaq Alpha (PSC)	1.91	2.32

3) From the applications perspective, the Compaq Alpha cluster at PSC yields the best performance among the three platforms we have benchmarked. This is evident from the wall clock timing data in Table 4 of both models on a total of 32 processors on all three platforms. Our CFD simulations using up to 1536 processors on PSC Compaq cluster, up to 512 processors on SDSC SP3 and up to 256 processors on the NCSA IA64 cluster, demonstrate a relative performance ratio of these platforms similar to that reflected in Table 4 on 32 processors. The PSC Compaq

cluster is about 2-3 times faster than the SDSC SP3, and the SP3 is roughly twice as fast as the IA64 cluster (or a little less).

Acknowledgments

This work was supported by ONR. Computer time for production runs was provided by DOD HPCMP (NAVO, ERDC, ARSC). Benchmarking time was provided by NCSA, NPACI and PSC. The authors would like to thank D. Rockwell for providing the PIV electronic data of the cylinder flow.

References

[1]Eldred, M. S., and Hart, W. E., "Design and Implementation of Multilevel Parallel Optimization on the Intel Teraflops," AIAA Paper 98-4707, *Proceedings of the 7th AIAA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998, pp. 44-54.

[2]Bova, S. W., Breshears, C., Cuicchi, C., Demirbilek, Z., and Gabb, H.A., "Dual-Level Parallel Analysis of Harbor Wave Response Using MPI and OpenMP," *International Journal of Supercomputing Applications*, Vol. 14, 2000, pp. 49-64.

[3]Eldred, M. S., Hart, W. E., Schimel, B. D., and van Bloemen Waanders, B. G., "Multilevel Parallelism for Optimization on MP Computers: Theory and Experiment," AIAA Paper 2000-4818, 2000.

[4]Taft, J. R., "Achieving 60 GFLOPS on the Production CFD Code OVERFLOW-MLP," *Parallel Computing*, Vol. 27, No. 4, 2001, pp. 521-536.

[5]Dong, S., and Karniadakis, G. E., "Dual-Level Parallelism for High-Order CFD Methods," *Parallel Computing*, Vol. 30, No. 1, 2004, pp. 1-20.

[6]Cappello, F., and Etiemble, D., "MPI versus MPI+OpenMP on the IBM SP for the NAS Benchmarks," *Supercomputing 2000: High Performance Networking and Computing* (SC2000), Nov. 2000.

[7]Henty, D. S., "Performance of Hybrid Message-Passing and Shared-Memory Parallelism for Discrete Element Modeling," *Supercomputing 2000: High Performance Networking and Computing* (SC2000), Nov. 2000.

[8]Dong, S., and Karniadakis, G. E., "P-refinement and P-threads," *Computer Methods in Applied Mechanics and Engineering*, Vol. 192, 2003, pp. 2191-2201.

[9]Gropp, W. D., Kaushik, D. K., Keyes, D. E., and Smith, B. F., "High-Performance Parallel Implicit CFD," *Parallel Computing*, Vol. 27, 2001, pp. 337-362.

[10]Luong, P., Breshears, C. P., and Ly, L. N., "Coastal Ocean Modeling of the U.S. West Coast with Multiblock Grid and Dual-Level Parallelism," *Supercomputing 2001: High Performance Networking and Computing* (SC2001), Nov. 2001.

[11]Loft, R. D., Thomas, S. J., and Dennis, J. M., "Terascale Spectral Element Dynamical Core for Atmospheric General Circulation Models," *Supercomputing 2001: High Performance Networking and Computing* (SC2001), Nov. 2001.

[12]Karniadakis, G. E., and Sherwin, S. J., *Spectral/hp element methods for CFD*, Oxford Univ. Press, 1999.

[13]Xiu, D., Lucor, D., Su, C.-H., and Karniadakis, G. E., "Stochastic Modeling of Flow-Structure Interactions Using Generalized Polynomial Chaos," *Journal of Fluids Engineering*, Vol. 124, 2002, pp. 51-59.

[14]Evangelinos C., and Karniadakis, G. E., "Dynamics and Flow Structures in the Turbulent Wake of Rigid and Flexible Cylinders Subject to Vortex-Induced Vibrations," *Journal of Fluid Mechanics*, Vol. 400, 1999, pp. 91-124.

[15]Newman, D., and Karniadakis, G. E., "A Direct Numerical Simulation Study of Flow Past a Freely Vibrating Cable," *Journal of Computational Physics*, Vol. 344, 1997, pp. 95-136.

[16]Karypis, G., and Kumar, V., "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," *SIAM Journal of Scientific Computing*, Vol. 20, 1998, pp. 359-392.

[17]Crawford, C. H., Evangelinos, C., Newman, D., and Karniadakis, G. E., "Parallel Benchmarks of Turbulence in Complex Geometries," *Computers & Fluids*, Vol. 25, 1996, pp. 677-698.

[18]Evangelinos, C., and Karniadakis, G. E., "Communication Patterns and Models in Prism: A Spectral Element-Fourier Parallel Navier-Stokes Solver," *Supercomputing 1996: High Performance Networking and Computing* (SC96), Nov. 1996.

[19]Karamanos, G. S., Evangelinos, C., Boes, R. C., Kirby, M., and Karniadakis, G. E., "Direct Numerical Simulation of Turbulence with a PC/Linux Cluster: Fact or Fiction?" *Supercomputing 1999: High Performance Networking and Computing* (SC99), Nov. 1999.

[20]Karniadakis, G. E., Israeli, M., and Orszag, S. A., "High-Order Splitting Methods for Incompressible Navier-Stokes Equations," *Journal of Computational Physics*, Vol. 97, 1991, p. 414.

[21]Saelim, N., and Rockwell, D., "Near Wake of a Cylinder in the Range of Shear Layer Transition," *Physics of Fluids*, submitted 2004.

[22]Biship, R. E. D., and Hassay, A. Y., "The Lift and Drag Forces on a Circular Cylinder in a Flowing Fluid," *Proceedings of the Royal Society of London*, Series A, Vol. 277, 1964, pp. 32-50.

[23]Gopalkrishnan, R., "Vortex-Induced Forces on Oscillating Bluff Bodies," Ph.D. dissertation, Dept. of Ocean Engineering, Massachusetts Institute of Technology, 1993.

[24]Williamson, C. H. K., "Vortex Dynamics in the Cylinder Wake," *Annual Review of Fluid Mechanics*, Vol. 28, 1996, pp. 477-539.

[25]Norberg, C., "Fluctuating Lift on a Circular Cylinder: Review and New Measurements," *Journal of Fluids and Structures*, Vol. 17, 2003, pp. 57-96.