Gordon Bell Prize Finalist Presentation
SC'99

Achieving High Sustained Performance in an Unstructured Mesh CFD Application

http://www.mcs.anl.gov/petsc-fun3d

Kyle Anderson, NASA Langley Research Center

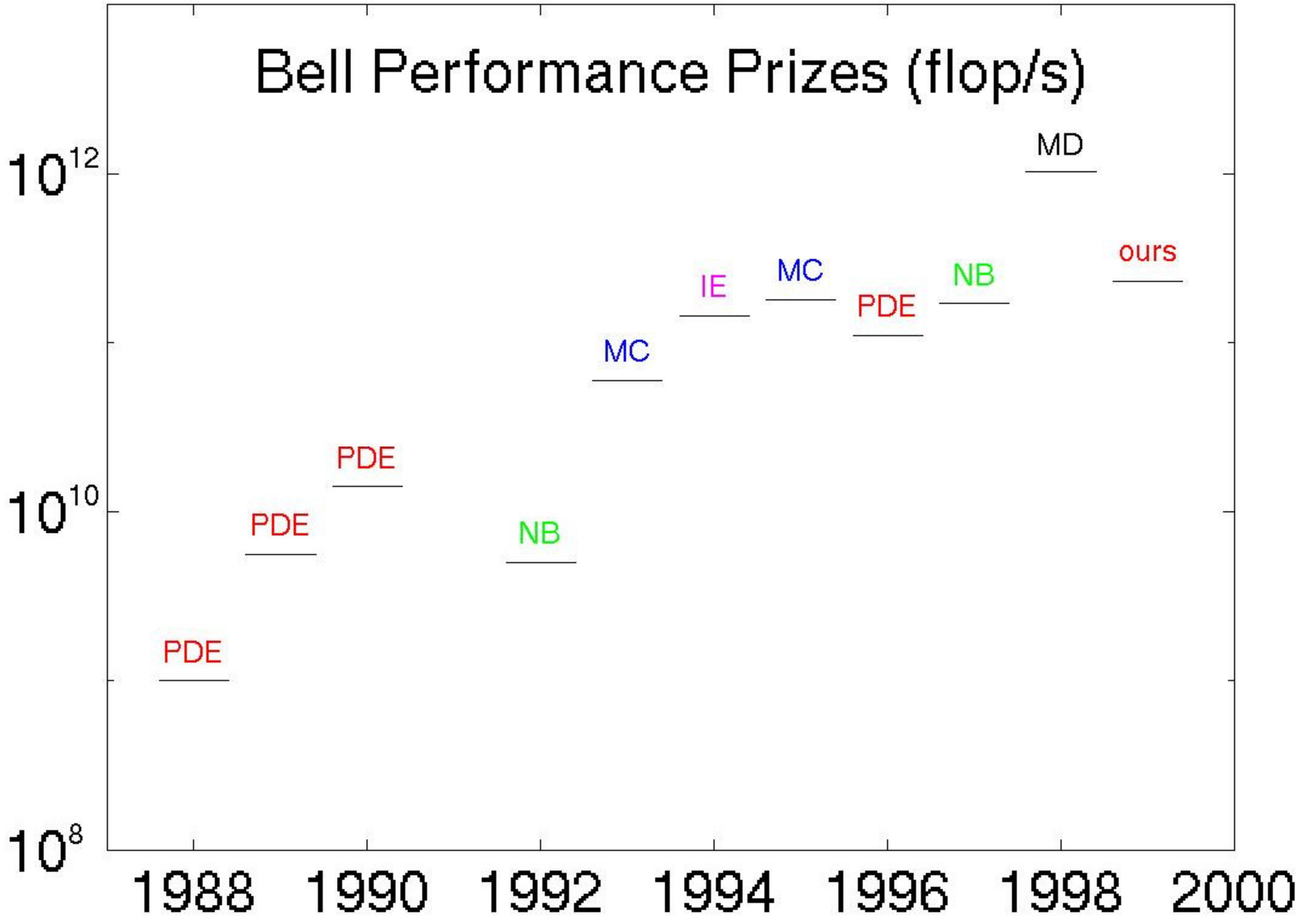
William Gropp, Argonne National Laboratory

Dinesh Kaushik, Old Dominion University & Argonne

David Keyes, Old Dominion University, LLNL & ICASE

Barry Smith, Argonne National Laboratory

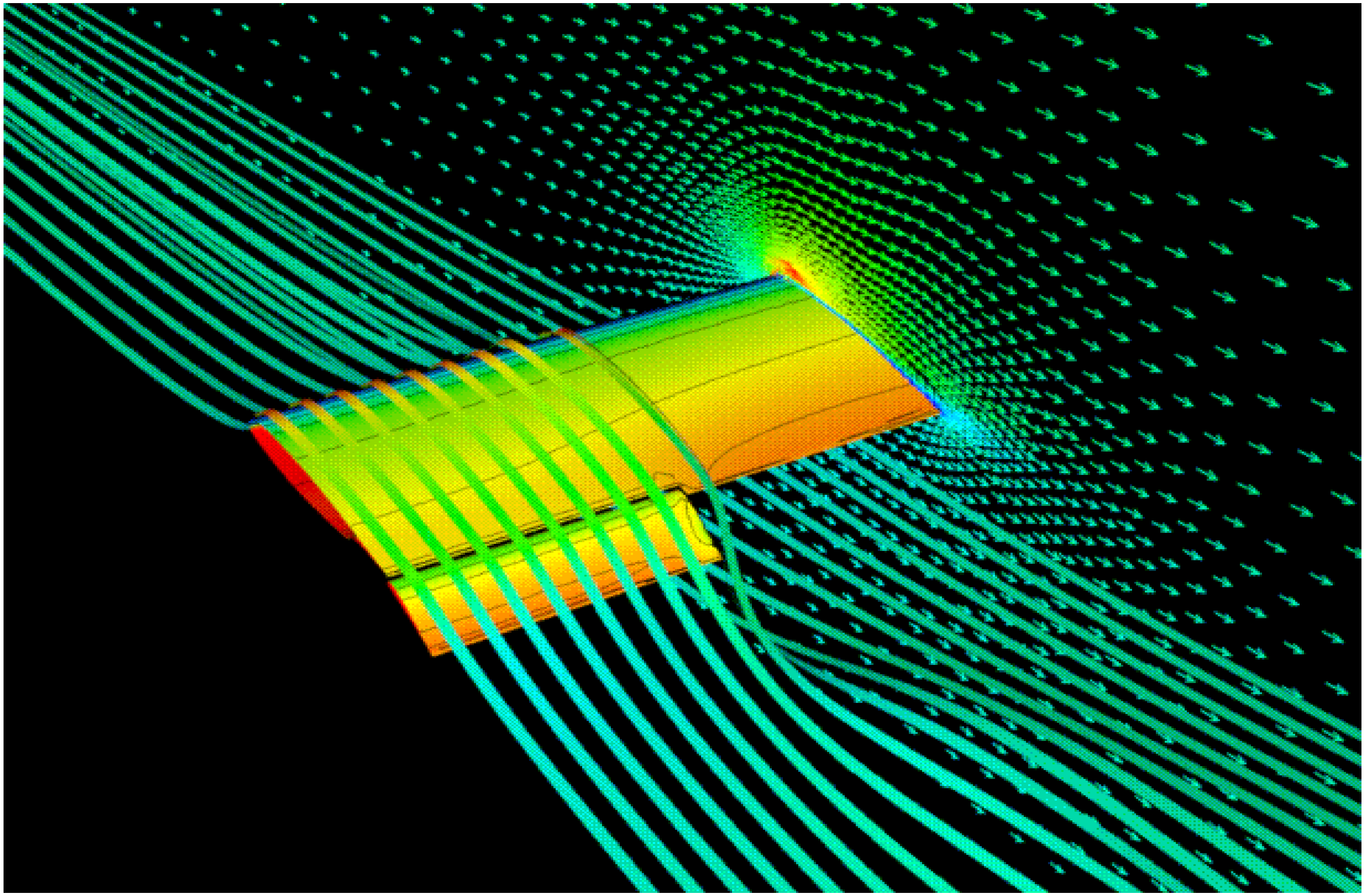
Application Performance History
3 orders of magnitude in 10 years



Features of this 1999 Submission

- Based on “legacy” (but contemporary) CFD application with significant F77 code reuse
- Portable, message-passing library-based parallelization, run on NT boxes through Tflop/s ASCI platforms
- Simple multithreaded extension (for ASCI Red)
- Sparse, unstructured data, implying memory indirection with only modest reuse - nothing in this category has ever advanced to Bell finalist round
- Wide applicability to other implicitly discretized multiple-scale PDE workloads - of interagency, interdisciplinary interest
- Extensive profiling has led to follow-on algorithmic research

Application Domain: Computational Aerodynamics

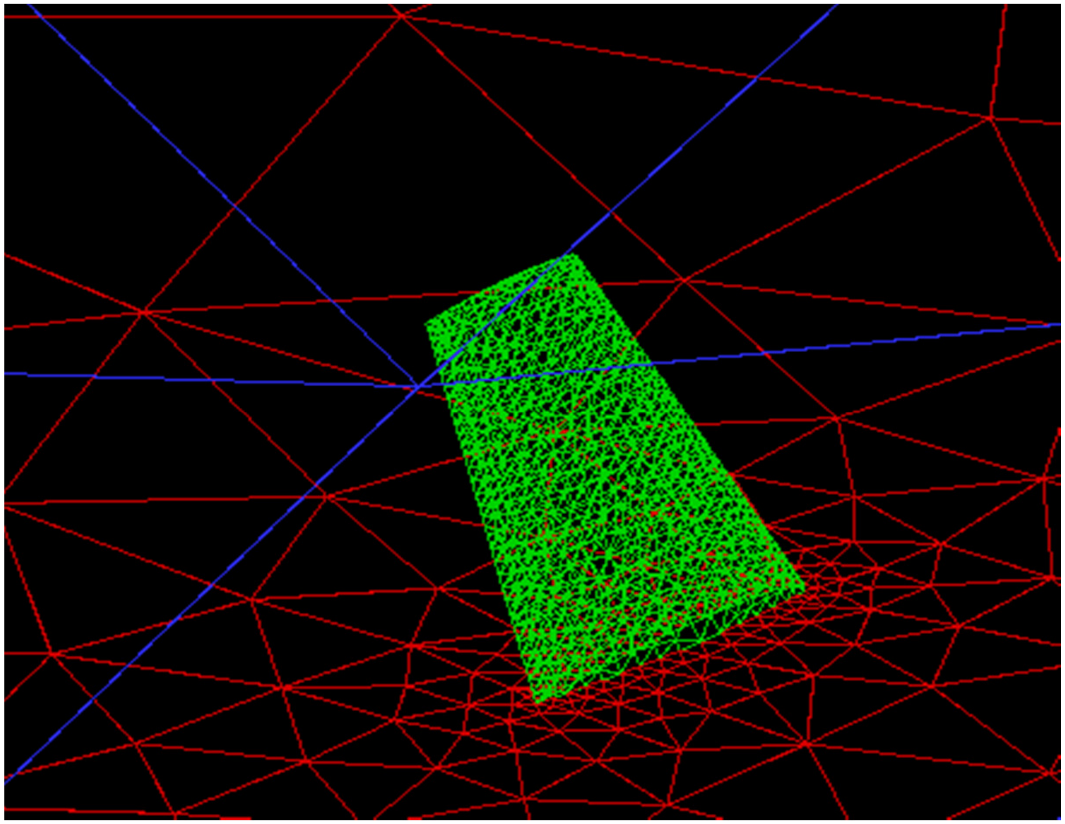


Background of FUN3D Application

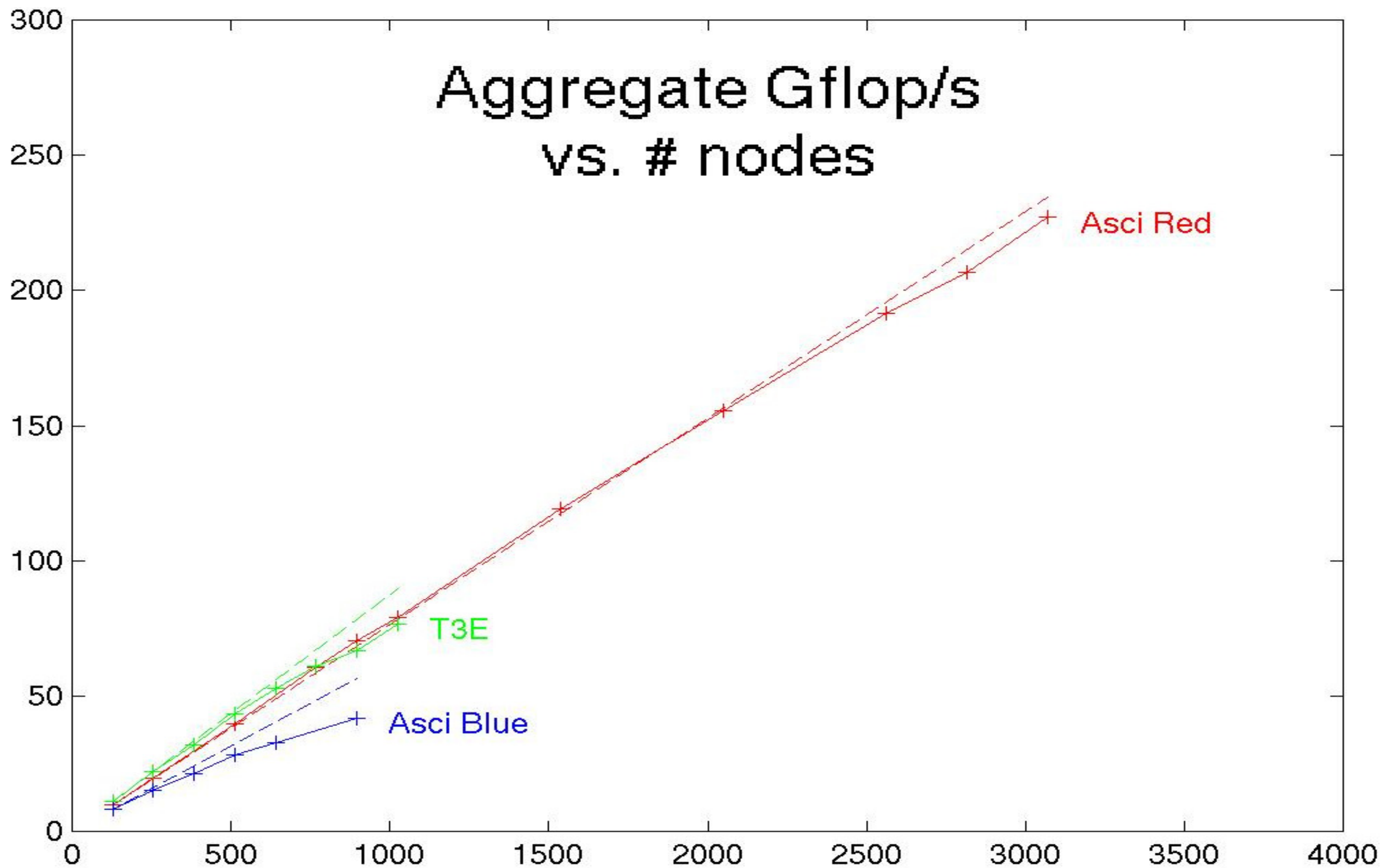
- Tetrahedral vertex-centered unstructured grid code developed by W. K. Anderson (LaRC) for steady compressible and incompressible Euler and Navier-Stokes equations (with one-equation turbulence modeling)
- Used in airplane, automobile, and submarine applications for analysis and design
- Standard discretization is 2nd-order Roe for convection and Galerkin for diffusion
- Newton-Krylov solver with global point-block-ILU preconditioning, with false timestepping for nonlinear continuation towards steady state; competitive with FAS multigrid in practice
- Legacy implementation/ordering is vector-oriented

Surface Visualization of Test Domain for Computing Flow over an ONERA M6 Wing

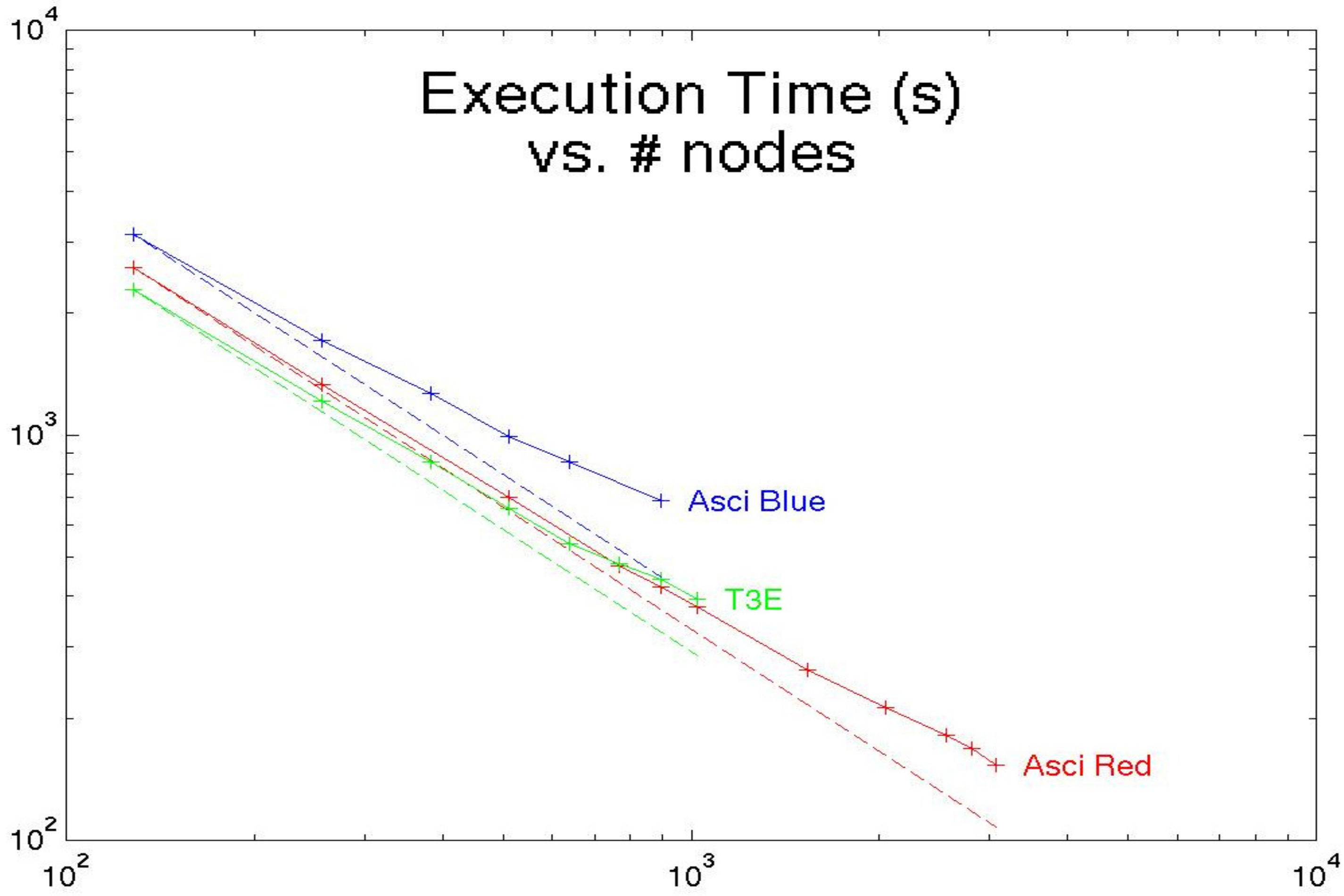
- Wing surface outlined in green triangles
- Nearly 2.8 M vertices in this computational domain



Fixed-size Parallel Scaling Results (Flop/s)

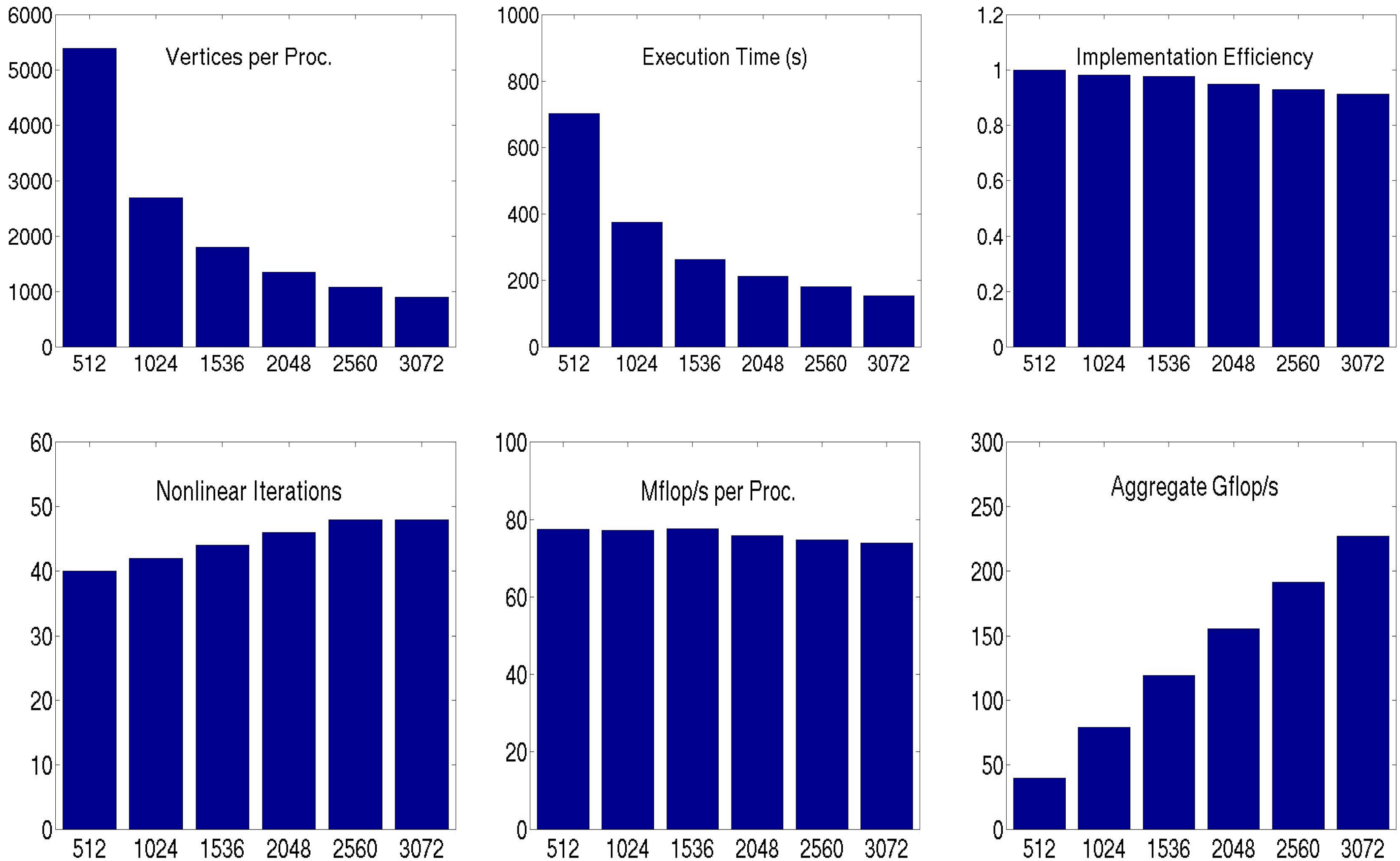


Fixed-size Parallel Scaling Results (Time in seconds)

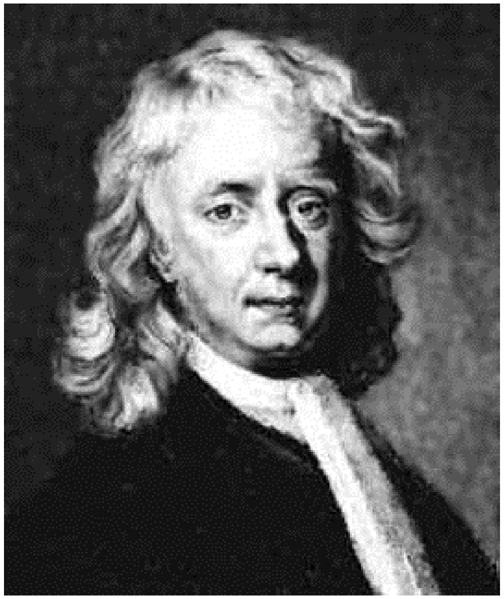


Inside the Parallel Scaling Results on ASCI Red

ONERA M6 Wing Test Case, Tetrahedral grid of 2.8 million vertices (about 11 million unknowns) on up to 3072 ASCI Red Nodes (each with dual Pentium Pro 333 MHz processors)



Algorithm: Newton-Krylov-Schwarz



Newton
nonlinear solver
asymptotically quadratic

Krylov
accelerator
spectrally adaptive

Schwarz
preconditioner
parallelizable

Merits of NKS Algorithm/Implementation

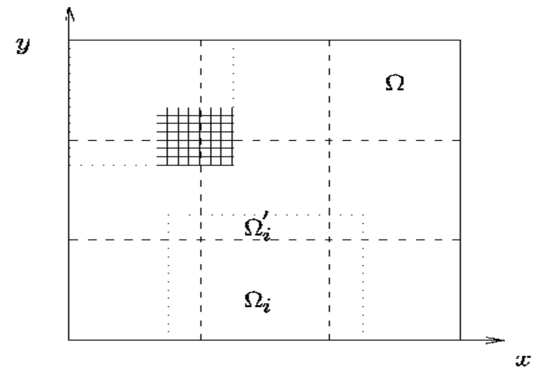
- Relative characteristics: the “exponents” are *naturally* good
  - Convergence scalability
    - weak (or no) degradation in problem size and parallel granularity (with use of small global problems in Schwarz preconditioner)
  - Implementation scalability
    - no degradation in ratio of surface communication to volume work (in problem-scaled limit)
    - only modest degradation from global operations (for sufficiently richly connected networks)
- Absolute characteristics: the “constants” can be *made* good
  - Operation count complexity
    - residual reductions of 10^{-9} in 10^{3} “work units”
  - Per-processor performance
    - up to 25% of theoretical peak
- Overall, machine-epsilon solutions require as little as 15 microseconds per degree of freedom!

Primary PDE Solution Kernels

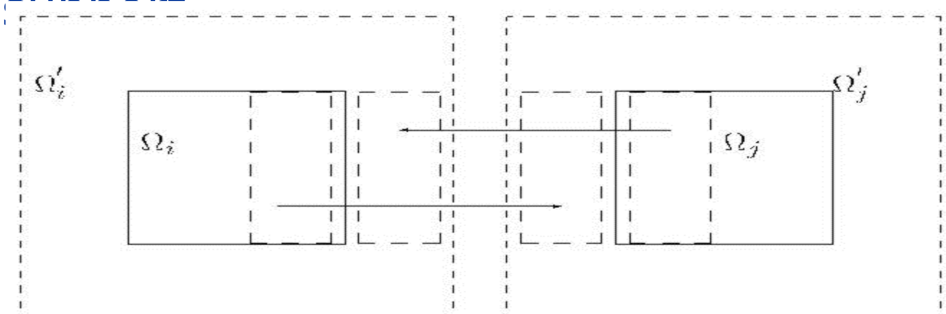
- Vertex-based loops
  - state vector and auxiliary vector updates
- Edge-based "stencil op" loops
  - residual evaluation
  - approximate Jacobian evaluation
  - Jacobian-vector product (often replaced with matrix-free form, involving residual evaluation)
- Sparse, narrow-band recurrences
  - approximate factorization and back substitution
- Vector inner products and norms
  - orthogonalization/conjugation
  - convergence progress and stability checks

Additive Schwarz Preconditioning for $Au=f$ in Ω

- Form preconditioner B out of (approximate) local solves on (overlapping) subdomains



- Let R_i and R_i^T be Boolean gather and scatter operations, mapping between a global vector and its i^{th} subdomain support



$$A_i = R_i A R_i^T$$

$$B_i = R_i^T \tilde{A}_i^{-1} R_i$$

$$B = \sum_i B_i$$

Iteration Count Estimates from the Schwarz Theory

[ref: Smith, Bjorstad & Gropp, 1996, Camb. Univ. Pr.]

- Krylov-Schwarz iterative methods typically converge in a number of iterations that scales as the square-root of the condition number of the Schwarz-preconditioned system
- In terms of N and P, where for d-dimensional isotropic problems, $N=h^{-d}$ and $P=H^{-d}$, for mesh parameter h and subdomain diameter H, iteration counts may be estimated as follows:

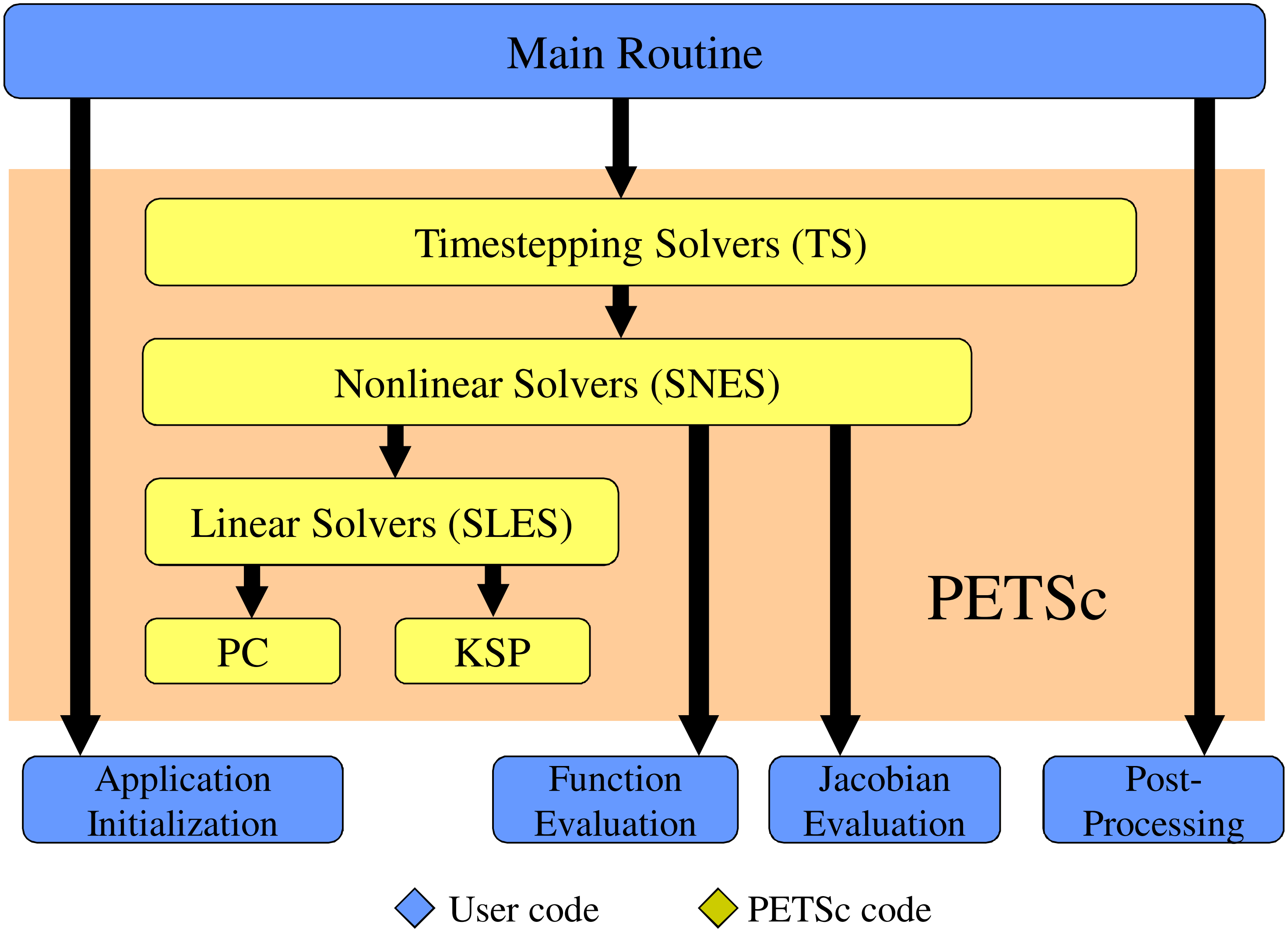
Preconditioning Type	in 2D	in 3D
Point Jacobi	$O(N^{1/2})$	$O(N^{1/3})$
Domain Jacobi	$O((NP)^{1/4})$	$O((NP)^{1/6})$
1-level Additive Schwarz	$O(P^{1/3})$	$O(P^{1/3})$
2-level Additive Schwarz	$O(1)$	$O(1)$

Time-Implicit Newton-Krylov-Schwarz Method

For nonlinear robustness, NKS iteration is wrapped in time-stepping:

```
for (l = 0; l < n_time; l++) {                       # n_time ~ 50
   select time step
   for (k = 0; k < n_Newton; k++) {                  # n_Newton ~ 1
     compute nonlinear residual and Jacobian
      for (j = 0; j < n_Krylov; j++) {                        # n_Krylov ~ 50
         forall (i = 0; i < n_Precon ; i++) {
            solve subdomain problems concurrently
         } // End of loop over subdomains
         perform Jacobian-vector product
         enforce Krylov basis conditions
         update optimal coefficients
         check linear convergence
      } // End of linear solver
     perform DAXPY update
     check nonlinear convergence
  } // End of nonlinear loop
} // End of time-step loop
```

Separation of Concerns: User Code/PETSc Library



Key Features of Implementation Strategy

- Follow the "owner computes" rule under the dual constraints of minimizing the number of messages and overlapping communication with computation
- Each processor "ghosts" its stencil dependences in its neighbors
- Ghost nodes ordered after contiguous owned nodes
- Domain mapped from (user) global ordering into local orderings
- Scatter/gather operations created between **local sequential** vectors and **global distributed** vectors, based on runtime connectivity patterns
- Newton-Krylov-Schwarz operations translated into local tasks and communication tasks
- Profiling used to help eliminate performance bugs in communication and memory hierarchy

Background of PETSc

- Developed by Gropp, Smith, McInnes & Balay (ANL) to support research, prototyping, and production parallel solutions of operator equations in message-passing environments
- Distributed data structures as fundamental objects - index sets, vectors/gridfunctions, and matrices/arrays
- Iterative linear and nonlinear solvers, combinable modularly and recursively, and extensibly
- Portable, and callable from C, C++, Fortran
- Uniform high-level API, with multi-layered entry
- Aggressively optimized: copies minimized, communication aggregated and overlapped, caches and registers reused, memory chunks preallocated, inspector-executor model for repetitivetasks (e.g., gather/scatter)

Single-processor Performance of PETSc-FUN3D

Processor	Clock MHz	Peak Mflop/s	Opt. % of Peak	Opt. Mflop/s	Reord. Only Mflop/s	Interl. only Mflop/s	Orig. Mflop/s	Orig. % of Peak
R10000	250	500	**25.4**	127	74	59	26	**5.2**
P3	200	800	**20.3**	163	87	68	32	**4.0**
P2SC (2 card)	120	480	**21.4**	101	51	35	13	**2.7**
P2SC (4 card)	120	480	**24.3**	117	59	40	15	**3.1**
604e	332	664	**9.9**	66	43	31	15	**2.3**
Alpha 21164	450	900	**8.3**	75	39	32	14	**1.6**
Alpha 21164	600	1200	**7.6**	91	47	37	16	**1.3**
Ultra II	300	600	**12.5**	75	42	35	18	**3.0**
Ultra II	360	720	**13.0**	94	54	47	25	**3.5**
Ultra II/HPC	400	800	**8.9**	71	47	36	20	**2.5**
Pent. II/LIN	400	400	**20.8**	83	52	47	33	**8.3**
Pent. II/NT	400	400	**19.5**	78	49	49	31	**7.8**
Pent. Pro	200	200	**21.0**	42	27	26	16	**8.0**
Pent. Pro	333	333	**18.8**	60	40	36	21	**6.3**

Single-processor Performance of PETSc-FUN3D

Processor	Clock MHz	Peak Mflop/s	Opt. % of Peak	Opt. Mflop/s	Reord. Only Mflop/s	Interl. only Mflop/s	Orig. Mflop/s	Orig. % of Peak
R10000	250	500	**25.4**	127	74	59	26	**5.2**
P3	200	800	**20.3**	163	87	68	32	**4.0**
P2SC (2 card)	120	480	**21.4**	101	51	35	13	**2.7**
P2SC (4 card)	120	480	**24.3**	117	59	40	15	**3.1**
604e	332	664	**9.9**	66	43	31	15	**2.3**
Alpha 21164	450	900	**8.3**	75	39	32	14	**1.6**
Alpha 21164	600	1200	**7.6**	91	47	37	16	**1.3**
Ultra II	300	600	**12.5**	75	42	35	18	**3.0**
Ultra II	360	720	**13.0**	94	54	47	25	**3.5**
Ultra II/HPC	400	800	**8.9**	71	47	36	20	**2.5**
Pent. II/LIN	400	400	**20.8**	83	52	47	33	**8.3**
Pent. II/NT	400	400	**19.5**	78	49	49	31	**7.8**
Pent. Pro	200	200	**21.0**	42	27	26	16	**8.0**
Pent. Pro	333	333	**18.8**	60	40	36	21	**6.3**

Single-processor Performance of PETSc-FUN3D

Processor	Clock MHz	Peak Mflop/s	Opt. % of Peak	Opt. Mflop/s	Reord. Only Mflop/s	Interl. only Mflop/s	Orig. Mflop/s	Orig. % of Peak
R10000	250	500	**25.4**	127	74	59	26	**5.2**
P3	200	800	**20.3**	163	87	68	32	**4.0**
P2SC (2 card)	120	480	**21.4**	101	51	35	13	**2.7**
P2SC (4 card)	120	480	**24.3**	117	59	40	15	**3.1**
604e	332	664	**9.9**	66	43	31	15	**2.3**
Alpha 21164	450	900	**8.3**	75	39	32	14	**1.6**
Alpha 21164	600	1200	**7.6**	91	47	37	16	**1.3**
Ultra II	300	600	**12.5**	75	42	35	18	**3.0**
Ultra II	360	720	**13.0**	94	54	47	25	**3.5**
Ultra II/HPC	400	800	**8.9**	71	47	36	20	**2.5**
Pent. II/LIN	400	400	**20.8**	83	52	47	33	**8.3**
Pent. II/NT	400	400	**19.5**	78	49	49	31	**7.8**
Pent. Pro	200	200	**21.0**	42	27	26	16	**8.0**
Pent. Pro	333	333	**18.8**	60	40	36	21	**6.3**

Lessons for High-end Simulation of PDEs

- Unstructured (static) grid codes can run well on distributed hierarchical memory machines, with attention to partitioning, vertex ordering, component ordering, blocking, and tuning
- Parallel solver libraries can give new life to the most valuable, discipline-specific modules of legacy PDE codes
- Parallel scalability is easy, but attaining high per-processor performance for sparse problems gets more challenging with each machine generation
- The NKS family of algorithms can be and must be tuned to an application-architecture combination; profiling is critical
- *Some* gains from hybrid parallel programming models (message passing and multithreading together) require little work; squeezing the last drop is likely much more difficult

Remaining Challenges

- Parallelization of the solver leaves mesh generation, I/O, and post processing as Amdahl bottlenecks in overall time-to-solution
  - moving finest mesh cross-country with ftp may take hours -- ideal software environment would generate and verify correctness of mesh in parallel, from relatively small geometry file
- Solution adaptivity of the mesh and parallel redistribution important in ultimate production environment
- Better multilevel preconditioners needed in some applications
- In progress:
  - wrapping a parallel optimization capability (Lagrange-Newton-Krylov-Schwarz) around our parallel solver, with substantial code reuse (automatic differentiation tools will help)
  - integrating computational snooping and steering into the PETSc environment

Bibliography

- *Toward Realistic Performance Bounds for Implicit CFD Codes*, Gropp, Kaushik, Keyes & Smith, 1999, in "Proceedings of Parallel CFD'99," Elsevier (to appear)
- *Implementation of a Parallel Framework for Aerodynamic Design Optimization on Unstructured Meshes*, Nielsen, Anderson & Kaushik, 1999, in "Proceedings of Parallel CFD'99," Elsevier (to appear)
- *Prospects for CFD on Petaflops Systems*, Keyes, Kaushik & Smith, 1999, in "Parallel Solution of Partial Differential Equations," Springer, pp. 247-278
- *Newton-Krylov-Schwarz Methods for Aerodynamics Problems: Compressible and Incompressible Flows on Unstructured Grids*, Kaushik, Keyes & Smith, 1998, in "Proceedings of the 11th Intl. Conf. on Domain Decomposition Methods," Domain Decomposition Press, pp. 513-520
- *How Scalable is Domain Decomposition in Practice,* Keyes, 1998, in "Proceedings of the 11th Intl. Conf. on Domain Decomposition Methods," Domain Decomposition Press, pp. 286-297
- *On the Interaction of Architecture and Algorithm in the Domain-Based Parallelization of an Unstructured Grid Incompressible Flow Code*, Kaushik, Keyes & Smith, 1998, in "Proceedings of the 10th Intl. Conf. on Domain Decomposition Methods," AMS, pp. 311-319

Acknowledgments

- Accelerated Strategic Computing Initiative, DOE
  - *access to ASCI Red and Blue machines*
- National Energy Research Scientific Computing Center (NERSC), DOE
  - *access to large T3E*
- SGI-Cray
  - *access to large T3E*
- National Science Foundation
  - *research sponsorship under Multidisciplinary Computing Challenges Program*
- U. S. Department of Education
  - *graduate fellowship support for D. Kaushik*

Related URLs

- Follow-up on this talk

  http://www.mcs.anl.gov/petsc-fun3d
- PETSc

  http://www.mcs.anl.gov/petsc
- FUN3D

  http://fmad-www.larc.nasa.gov/~wanderso/Fun
- ASCI platforms

  http://www.llnl.gov/asci/platforms
- International Conferences on Domain Decomposition Methods

  http://www.ddm.org
- International Conferences on Parallel CFD

  http://www.parcfd.org