High Performance Computing Workshop

Session 5: Scalable Software Tools

David Keyes
KAUST and Columbia University

Philosophy of presentation

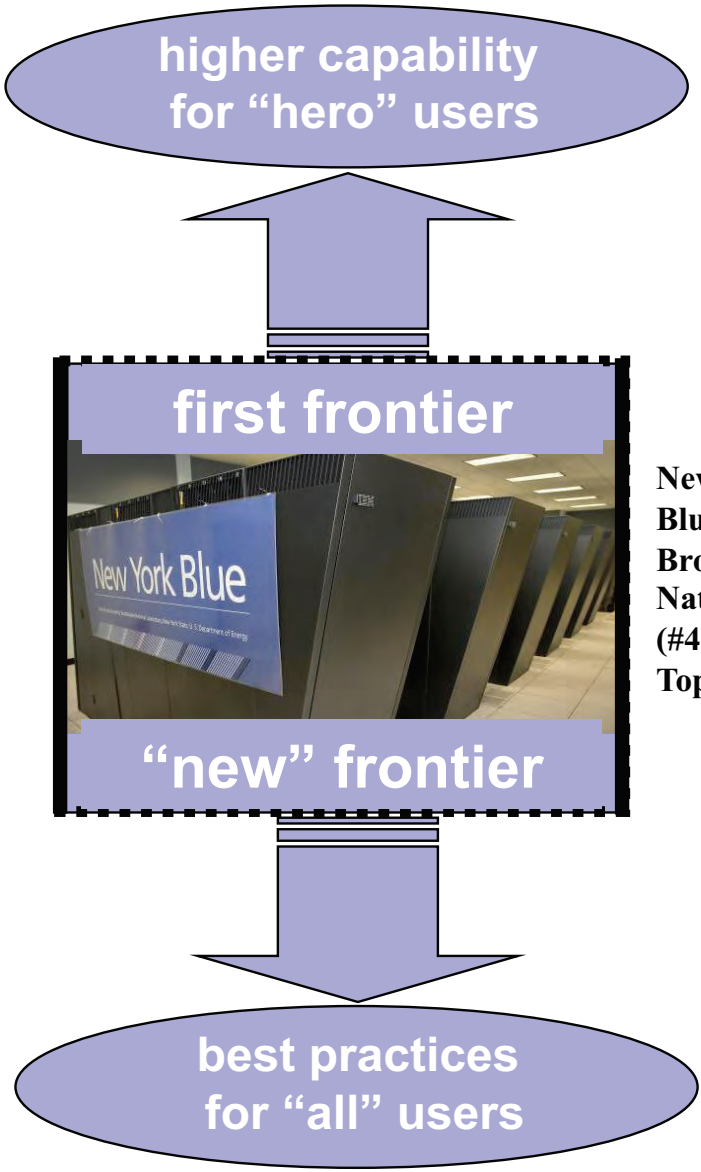
- Standards from commercial software engineering are beginning to become established in scientific software libraries
- Benefits of this canonization of "scientific cyberinfrastructure" include
  - confidence of users in quality and persistence of software
  - economies of scale
  - greater recognition for developers
  - better trained CS&E workforce
  - propagation of best practices into user applications
  - propagation of library quality standards into user applications
- We examine the trends, survey the contributions of the *Scientific Discovery through Advanced Computing* (SciDAC) program, and give examples from the *Towards Optimal Petascale Simulations* (TOPS) project

Caveat

- **The early part of the talk does a breadth-oriented survey of SciDAC enabling technologies (to support SciDAC applications such as John's lectures describe)**
- **In the latter part of the talk does a depth-oriented of the speaker's own TOPS project (Towards Optimal Petascale Simulations)**
- **The depth-oriented parts of the talk could be claimed for *many* of the SciDAC Centers for Enabling Technologies (CETs), of which TOPS is merely representative**
- **However, so as not to irresponsibly state the intentions or inaccurately characterize the progress of other CETs, the concrete focus will be on TOPS only**

High-performance computing: two frontiers

- Two frontiers
  - raise the peak capability for simulation experts
  - lower the HPC simulation entry threshold for people who are expert in something else
- Historically, rewards and attention go to the former
- U.S. DOE's Scientific Discovery through Advanced Computing (SciDAC) program attempts the latter, as well



New York Blue at Brookhaven National Lab (#45 on the Top 500)

Prominence of scientific software engineering defines our simulation era

(dates are symbolic)

1686

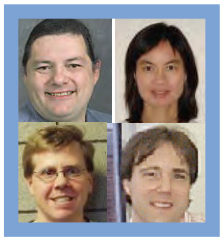
scientific models

1947

numerical algorithms

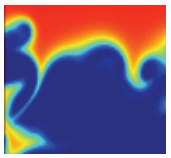
1976

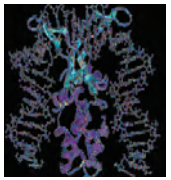
computer architecture

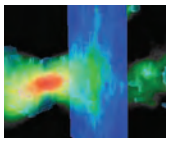


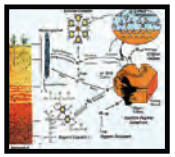
1992

scientific software engineering







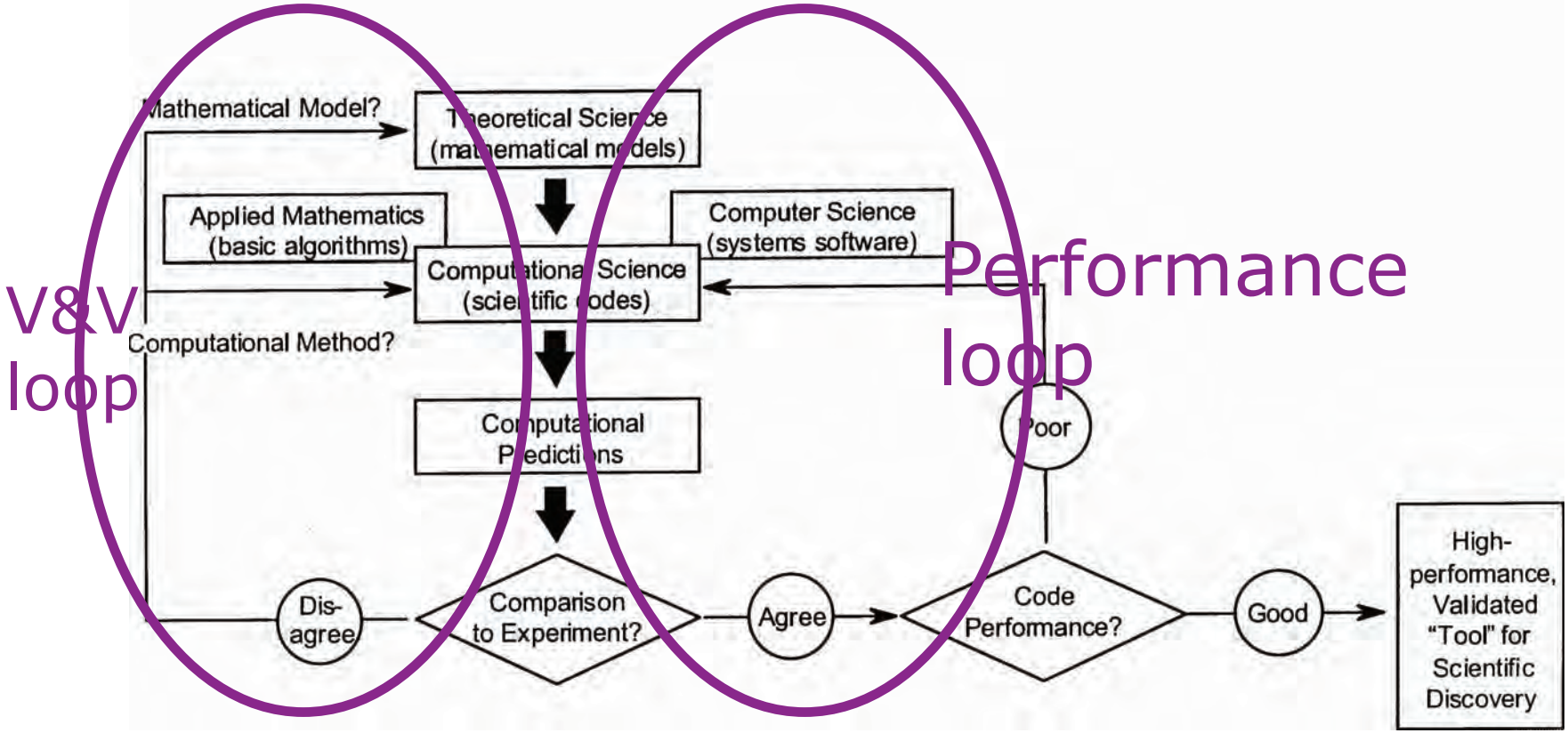


"Computational science is undergoing a phase transition." – D. Hitchcock, DOE

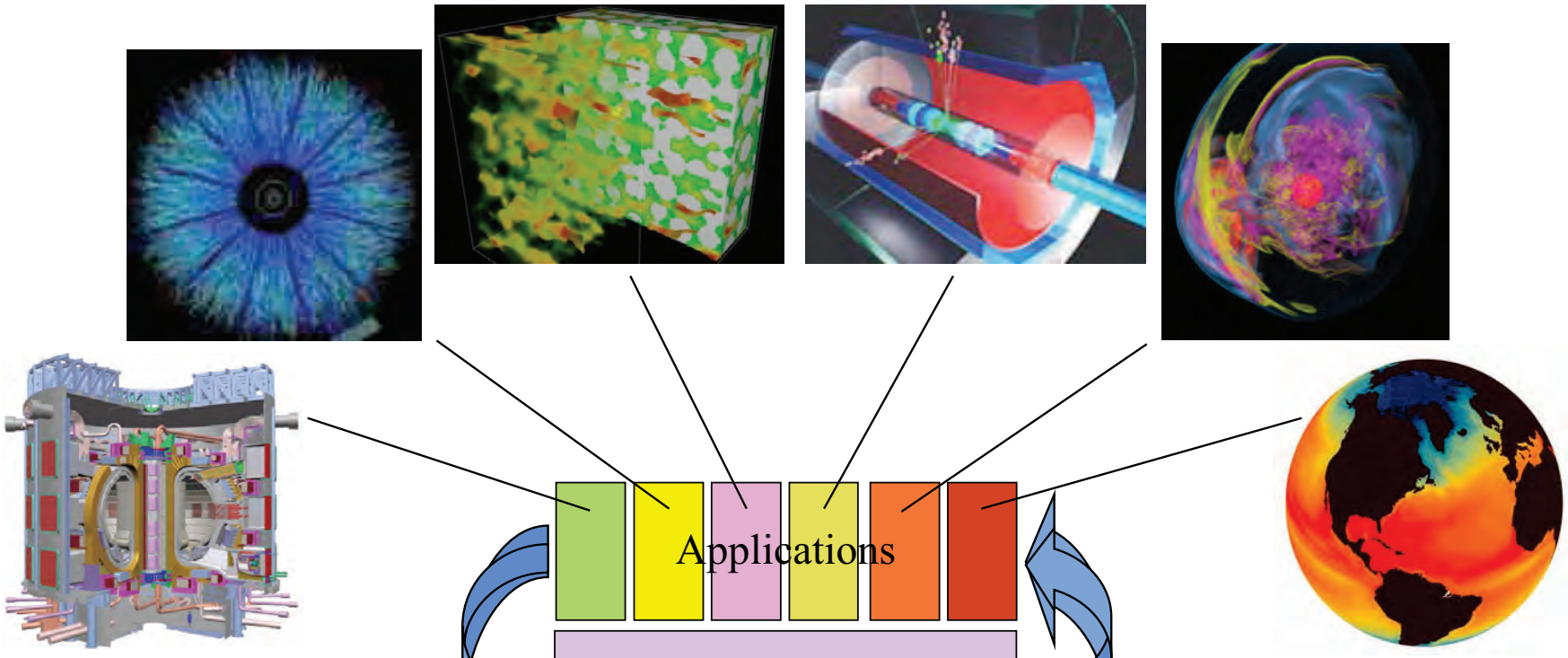
Defining characteristics of this phase change

- Encapsulation – an abstraction of data
- Polymorphism – an abstraction of operations
- Composability – an abstraction of interfaces, “plug’n’play”
- Extensibility – ease of adding functionality underneath the abstractions
- Portability – ease of moving code across architectures while maintaining correctness and *performance*
- Documentation and support
- Development disciplines
  - design to interface
  - versioning
  - unit testing
  - nightly regressions

Designing a simulation code – the diagram that launched the SciDAC program



SciDAC philosophy: common cyberinfrastructure



Many applications drive

Enabling technologies respond to all

Design principle: multiple layers

- **Top layer (all users)**
  - **Abstract interface featuring language of application domain, hiding details, with conservative parameter defaults**
  - ***Robustness, correctness, ease of use***
- **Middle layers (experienced users)**
  - **Rich collection of state-of-the-art methods and data structures, exposed upon demand, highly configurable**
  - ***Capability, algorithmic efficiency, extensibility, composability, comprehensibility of performance and resource use***
- **Bottom layer (developers)**
  - **Support for variety of execution environments**
  - ***Portability, implementation efficiency***

HPC enabling technologies

- **Model-related**
  - Geometric modelers
  - Meshers
  - Discretizers
  - Partitioners
  - Solvers / integrators
  - Adaptivity systems
  - Visualization systems
  - Workflow controllers
  - Data miners
- **Development-related**
  - Configuration systems
  - Source-to-source translators
  - Compilers
  - Messaging systems
  - Debuggers
  - Profilers
  - Version controllers
  - Regression testers

- **"Enabling technologies" groups develop reusable software and partner with application groups, who are *not* funded to "roll their own"**
- **At 2006 renewal ("SciDAC-2"), 49 projects share over $60M/year, divided between:**
  - **applications projects**
  - **lab-based Centers for Enabling Technology (CETs)**
  - **academic-hosted "institutes"**
- **Plus, petaflop/s-scale IBM BlueGene/P machine at Argonne, and Cray XT-5 machine available at Oak Ridge, and XT-4 at Berkeley for SciDAC researchers**

SciDAC's computer science "centers"

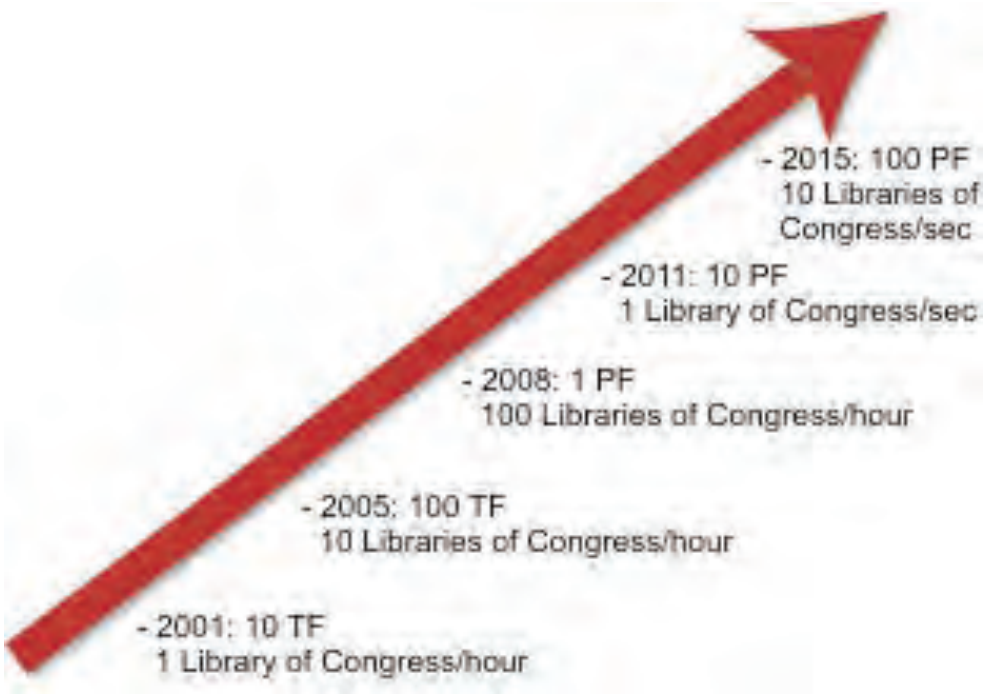
- **Petascale Data Storage Institute (PDSI)**
  PI: ***G. Gibson, CMU***
  For storage and retrieval of petascale data sets
- **Performance Engineering Research Institute (PERI)**
  PI: ***R. Lucas, USC***
  For understanding and improving SciDAC application performance
- **Technology for Advanced Scientific Component Software (TASCS)**
  PI: ***D. Bernholdt, ORNL***
  For development of a common component architecture
- **Center for Enabling Distributed Petascale Science (CEDPS)**
  PI: ***I. Foster, ANL***
  For providing user-friendly, secure access to distributed scientific data
- **Earth System Grid (ESG)**
  PI: ***D. Williams, LLNL***
  For easier exchange of climate change research data

See: www.scidac.gov/compsci/compsci.html

PDSI

Petascale Data Storage Institute

Focus on the data storage problems found in petascale scientific computing environments, with special attention to community issues such as interoperability and shared tools to meet petascale demands on information storage capacity, performance, concurrency, reliability, availability, and manageability. The last decade has shown that parallel file systems can barely keep pace with high performance computing along these dimensions.

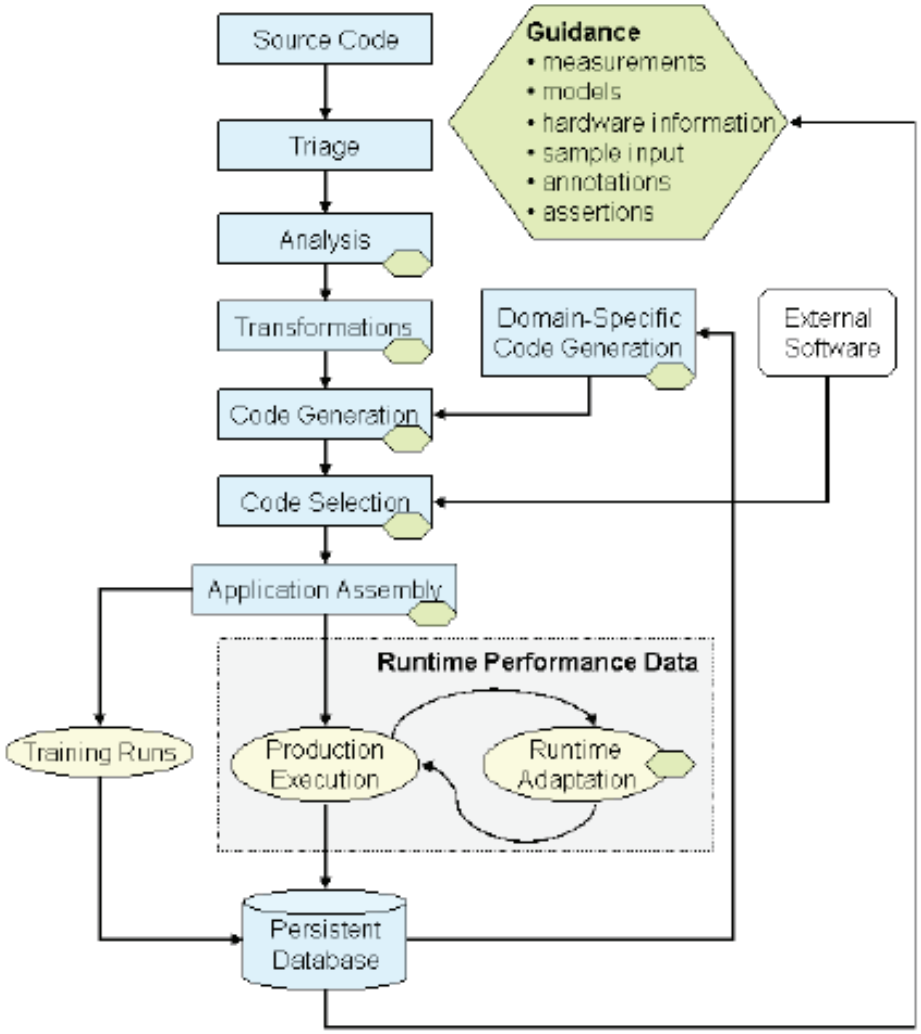


PERI

Performance Engineering Research Institute

Focus on delivering petascale performance to complex scientific applications on leadership class computing systems. Develop tools that analyze a scientific application, both as source code and during execution, generate a space of tuning options, and search for a near-optimal performance solution.

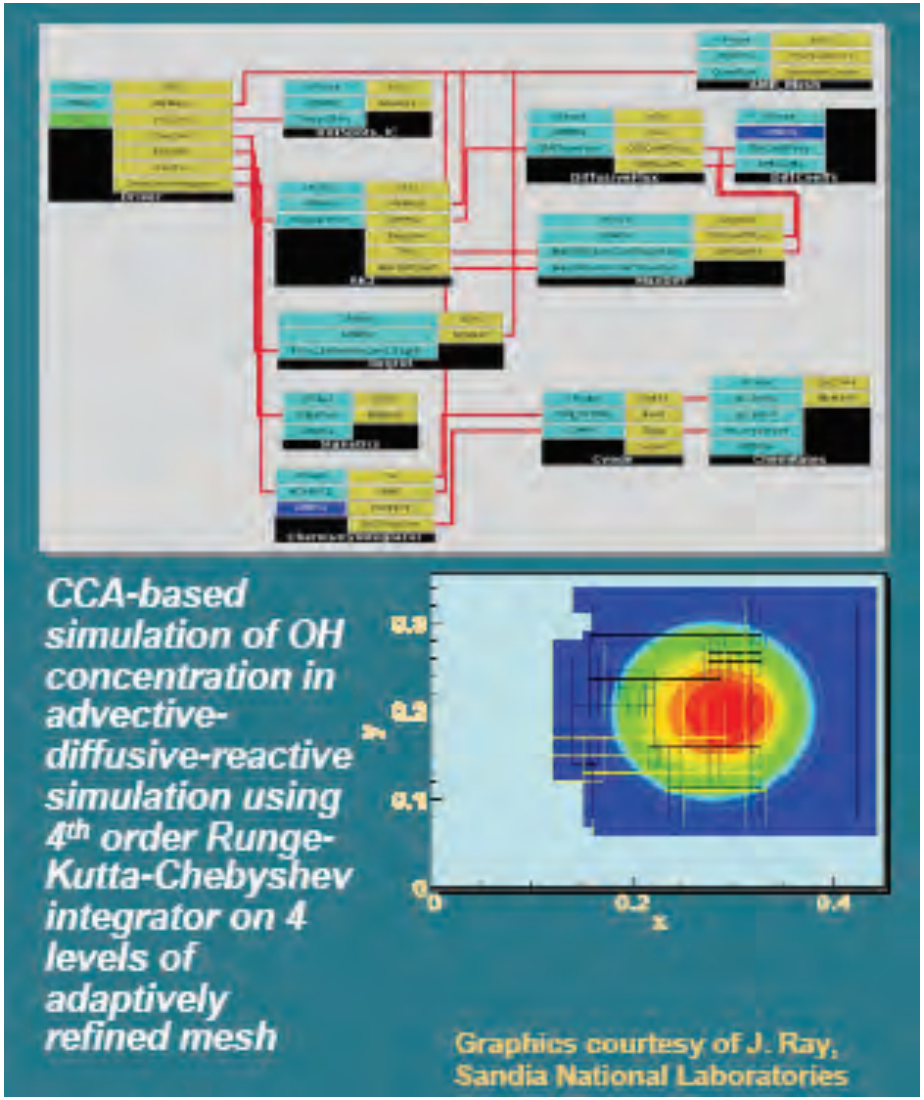
To realize this vision requires enhancement of automatic code manipulation tools, automatic run-time parameter selection, automatic communication optimization, and intelligent heuristics to control the combinatorial explosion of tuning possibilities.



TASCS

Technology for Advanced Scientific Component Software

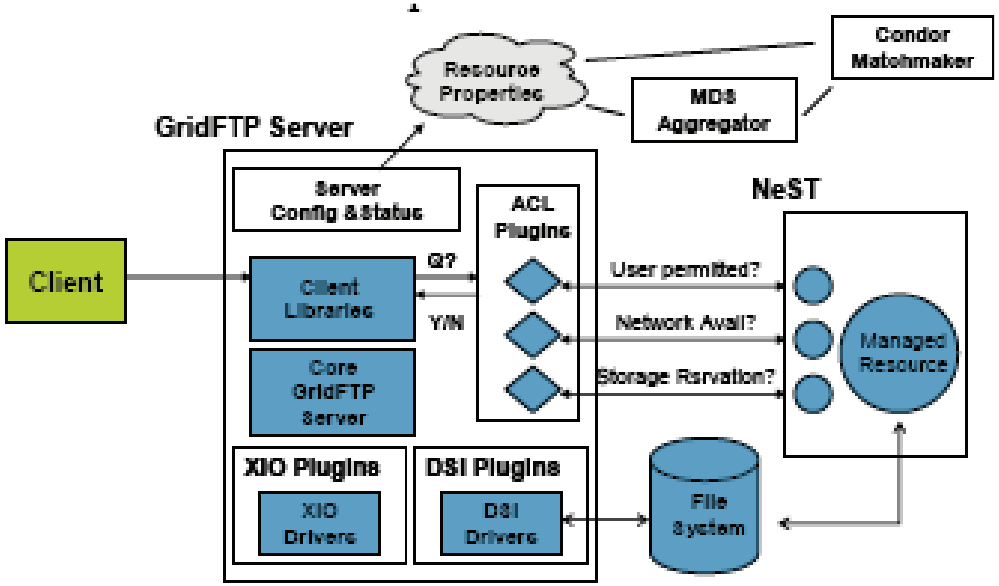
Enhance the core Common Component Architecture (CCA) software environment, with emphasis on improving usability, and build a component ecosystem to provide more off-the-shelf components. CCA software development emphasizes *components* as units of software with well-defined functionality that interact with other components through clearly-defined *interfaces*. Components are natural units of decomposition and interaction for both software and software developers. They enable scientists to work together as a cohesive scientific enterprise across disciplines, geographical boundaries, and technical preferences.



CEDPS

Center for Enabling Distributed Petascale Science

Design and develop and then—in collaboration with DOE application projects—deploy and evaluate powerful services and tools for data placement and science service construction. Produce technical innovations designed to allow for (a) rapid and dependable data placement within a distributed high-performance environment and (b) the convenient construction of scalable science services that provide for the reliable and high-performance processing of computation and data analysis requests from many remote clients. CEDPS will also address the important problem of troubleshooting these and other related ultra-high-performance distributed activities from the perspective of both performance and functionality.

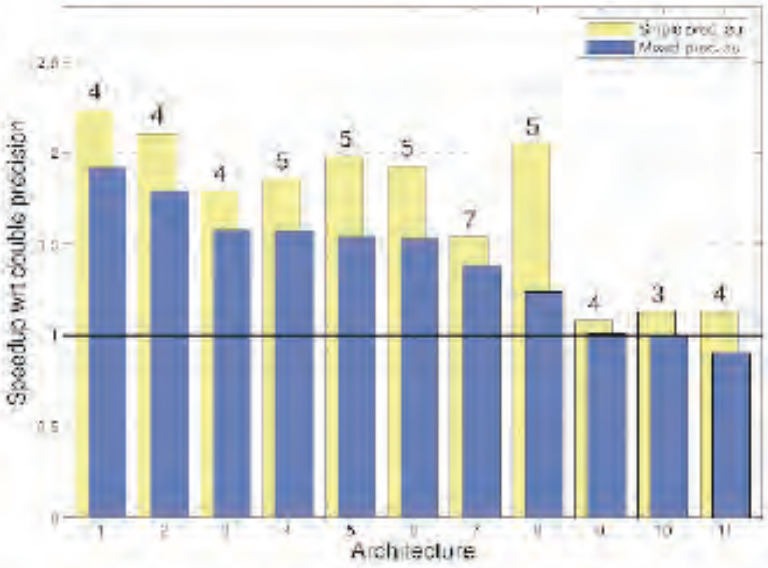


CScADS

Center for Scalable Application Development Software

Effectively utilize emerging multi-core architectures. Emphasize compilation methodology for both existing languages such as Fortran 90 and emerging partitioned global address space languages including co-array Fortran; auto-tuning techniques; rapid prototyping systems; and scalable performance tools. Develop and maintain open-source shared software infrastructures to enable the research and development community to incrementally construct programming support technologies that are portable across a broad range of high-end computer architectures.

Iterative Refinement for Dense $Ax = b$

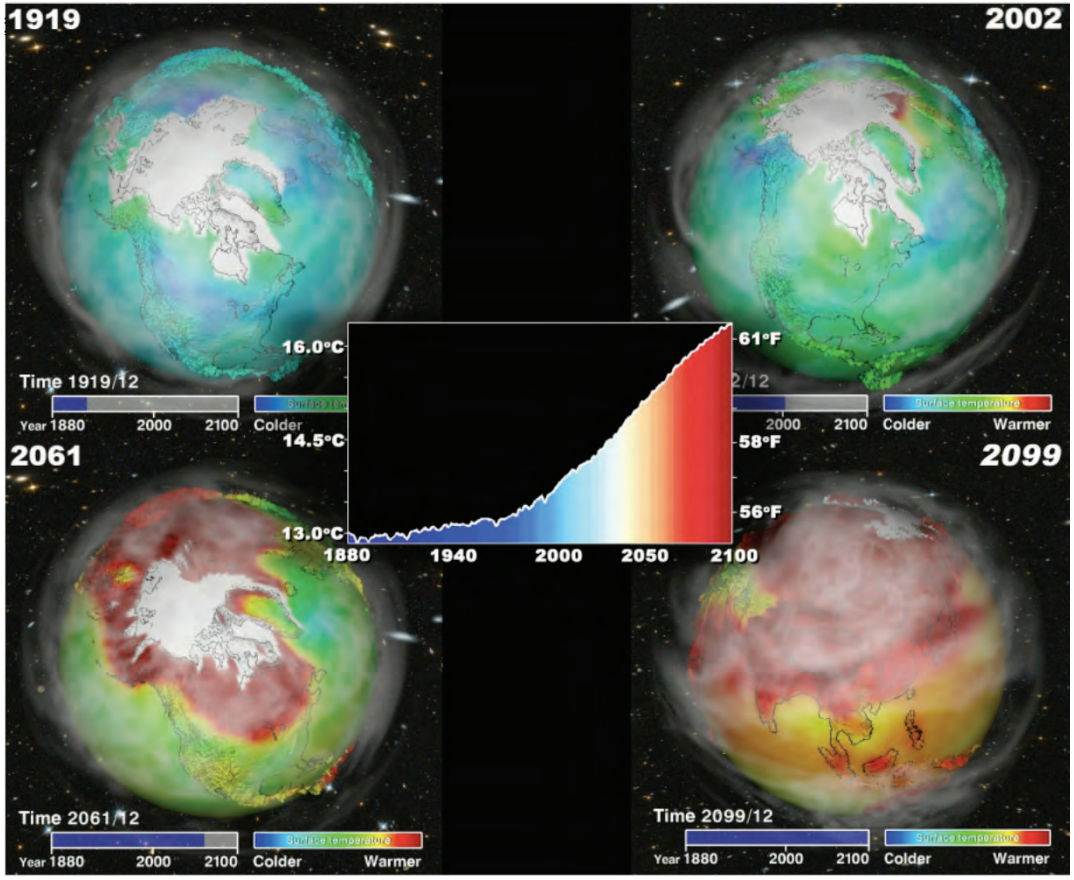


	Architecture (BLAS)
1	Intel Pentium III Coppermine (Goto)
2	Intel Pentium III Katmai (Goto)
3	Sun UltraSPARC IIe (Sunperf)
4	Intel Pentium IV Prescott (Goto)
5	Intel Pentium IV-M Northwood (Goto)
6	AMD Opteron (Goto)
7	Cray X1 (libsci)
8	IBM Power PC G5 (2.7 GHz) (VecLib)
9	Compaq Alpha EV6 (CXML)
10	IBM SP Power3 (ESSL)
11	SGI Octane (ATLAS)

ESG

Earth System Grid

Provide climate researchers worldwide with access to: data, information, models, analysis tools, and computational resources required to make sense of enormous climate simulation datasets. Integrate distributed data and computers, high-bandwidth wide-area networks, and remote computing using climate data analysis tools in a highly collaborative problem-solving environment.



SciDAC's visualization science "centers"

- **Interoperable Tools for Advanced Petascale Simulations (ITAPS)**

  PI: ***K.-L. Ma, UC-Davis***

  For multimodal scientific visualization

- **Visualization and Analytics Center for Enabling Technology (VACET)**

  PI: ***W. Bethel, LBNL***

  For multimodal scientific visualization and analyses, including overlay of uncertainty quantification

- **Scientific Data Management (SDM) Center**

  PI: ***A. Shoshani, LBNL***

  For end-to-end data curation

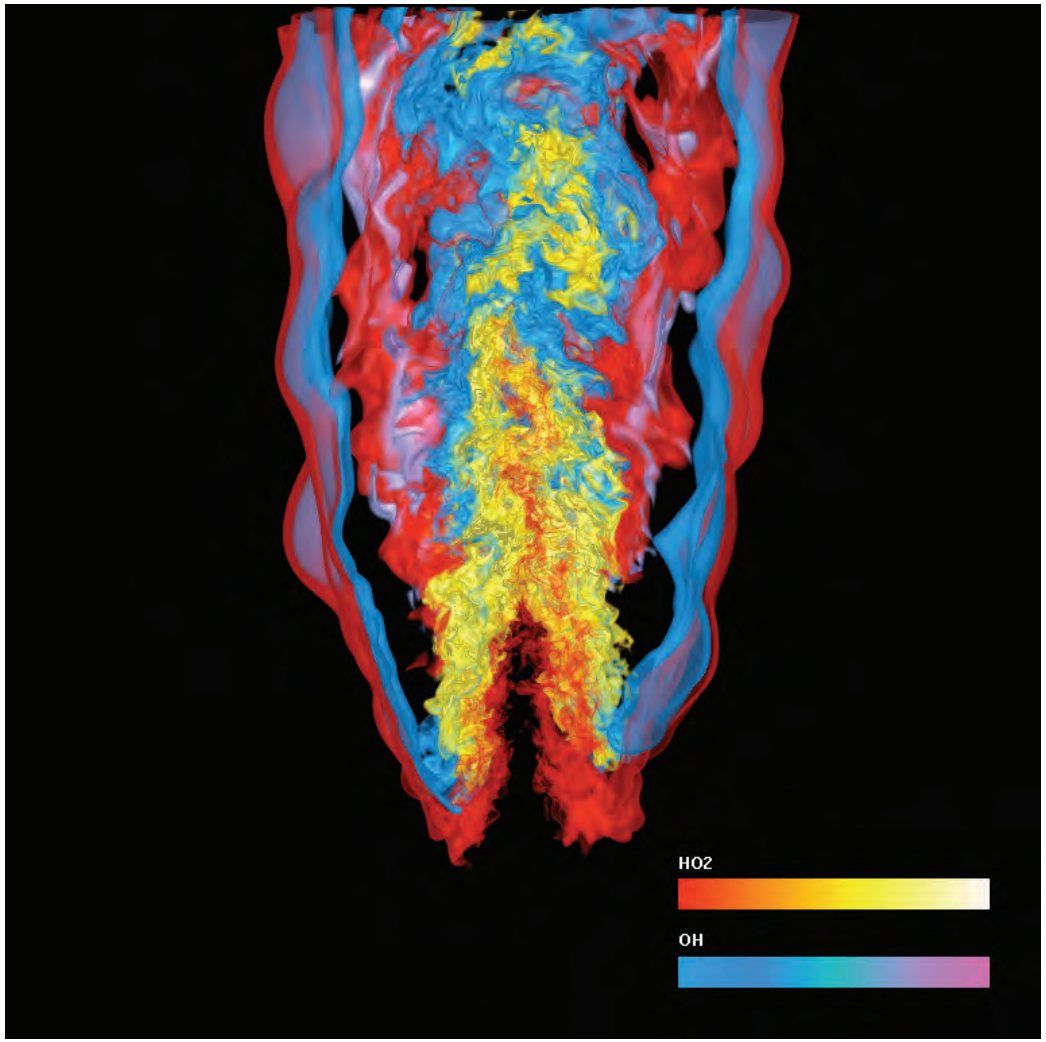
See: www.scidac.gov/viz/viz.html

IUV

Institute for Ultra-scale Visualization

Develop and evaluate parallel visualization strategies on diverse platforms including general-purpose clusters, dedicated visualization clusters, and especially general-purpose massively parallel supercomputers. Develop software for the following five critical technology focus areas:

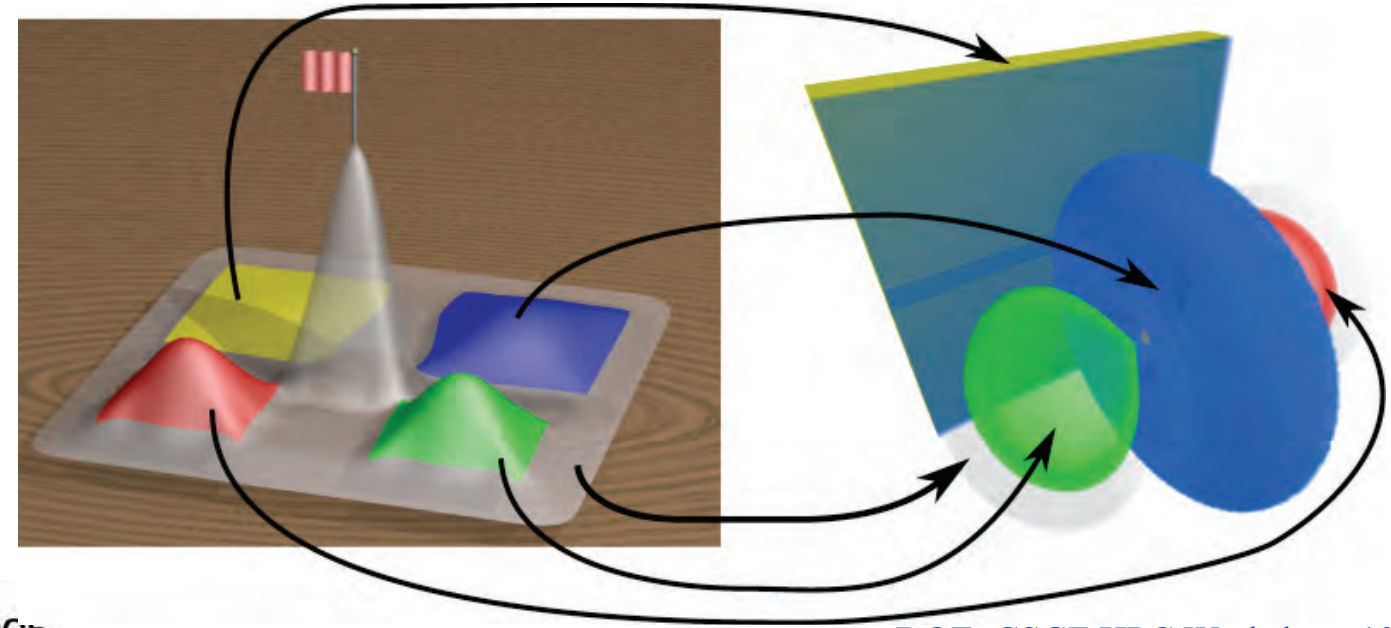
- **parallel rendering framework and API**
- **parallel I/O**
- **parallel visualization algorithms**
- **GPU abstraction and interfaces**
- **parallel visualization tools**



VACET

Visualization and Analytics Center for Enabling Technology

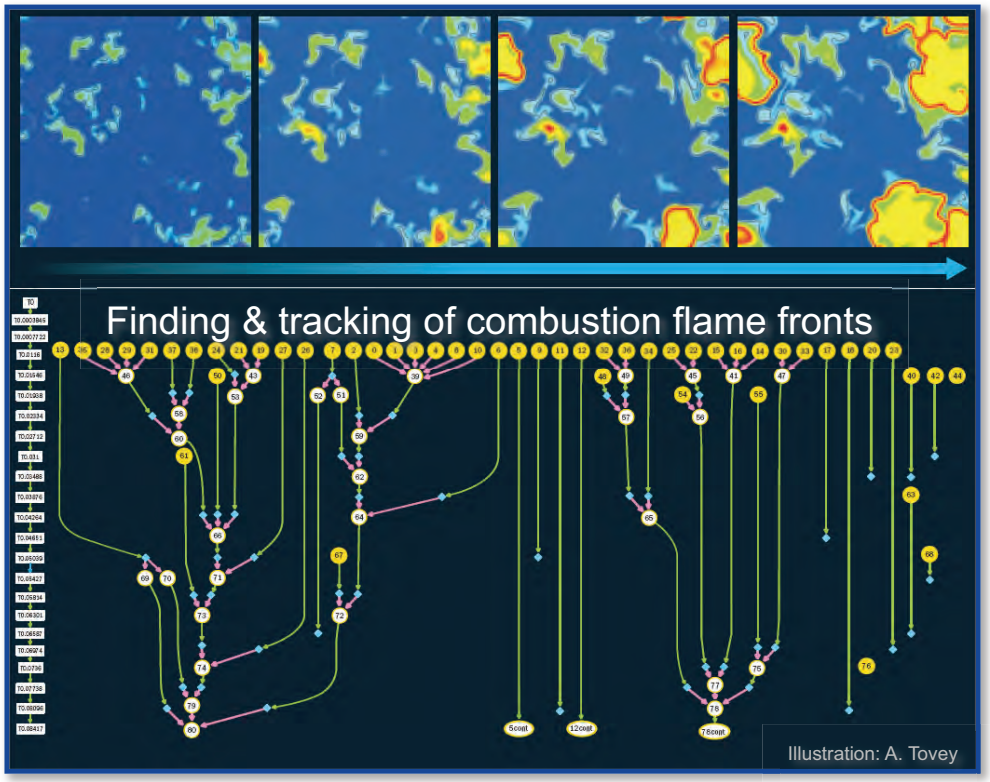
Integrate visualization staples for visualizing scalar fields (isocontouring, hyperslicing, direct volume rendering), vector fields (direct representation techniques like glyph-based visualization and indirect representation techniques like streamlines and particle advection), support tools like transfer function editors, dimension reduction and projection techniques like orthogonal and arbitrary slicing, and techniques for displaying computational grids. Visualize uncertainty estimates along with the data itself.



SDM

SciDAC Data Management Center

Effectively generate, manage, and analyze information through a comprehensive, end-to-end approach encompassing all of the stages from the initial data acquisition to the final analysis of the data. Priorities are: (1) more efficient access to storage systems, (2) technologies to facilitate better understanding of data, in particular the ability to effectively perform complex data analysis and searches over large data sets, and (3) generating, collecting and storing, data post-processing, and analysis of results. Workflow tools for automation of this process in a robust, tractable, and recoverable fashion are required to enhance scientific exploration.



SciDAC's computational math "centers"

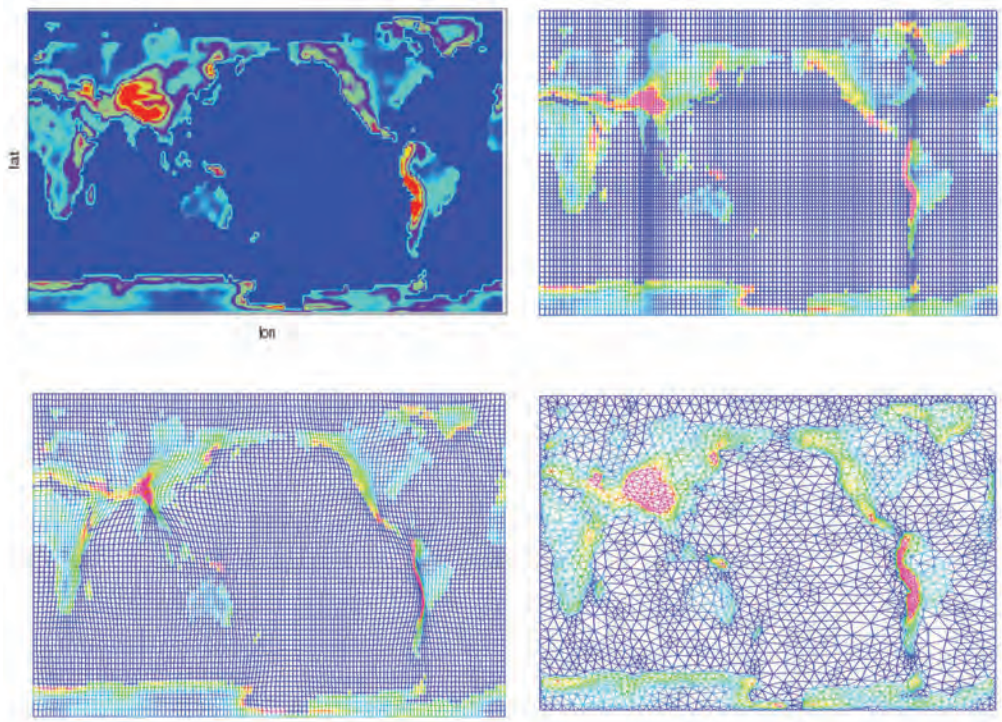
- **Interoperable Tools for Advanced Petascale Simulations (ITAPS)**
  PI: ***L. Freitag-Diachin, LLNL***
  For complex domain geometry
- **Algorithmic and Software Framework for Partial Differential Equations (APDEC)**
  PI: ***P. Colella, LBNL***
  For solution adaptivity
- **Combinatorial Scientific Computing and Petascale Simulation (CSCAPES)**
  PI: ***A. Pothen, Purdue U***
  For partitioning and ordering
- **Towards Optimal Petascale Simulations (TOPS)**
  PI: ***D. Keyes, Columbia U***
  For scalable solution

See: www.scidac.gov/math/math.html

ITAPS

Interoperable Tools for Advanced Petascale Simulations

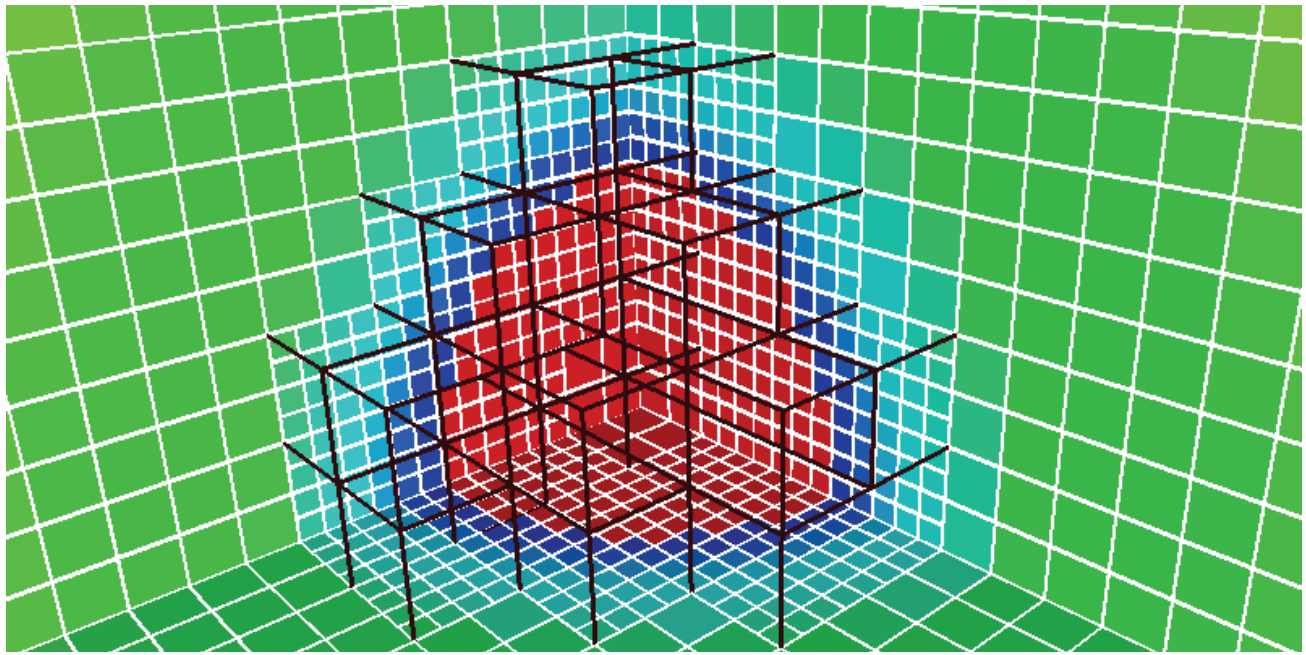
Develop framework for use of multiple mesh and discretization strategies within a single PDE simulation. Focus on high-quality hybrid mesh generation for representing complex and evolving domains, high-order discretization techniques, and adaptive strategies for automatically optimizing a mesh to follow moving fronts or to capture important solution features.

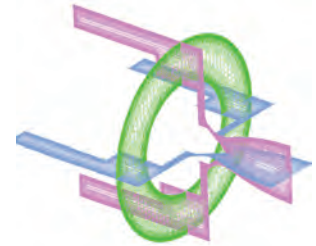


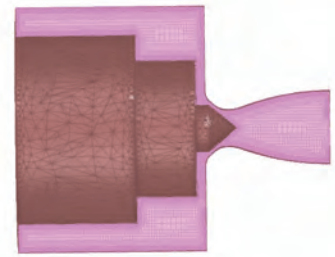
APDEC

Algorithmic and Software Framework for PDEs

Develop framework for PDE simulation based on locally structured grid methods, including adaptive meshes for problems with multiple length scales; embedded boundary and overset grid methods for complex geometries; efficient and accurate methods for particle and hybrid particle/mesh simulations.



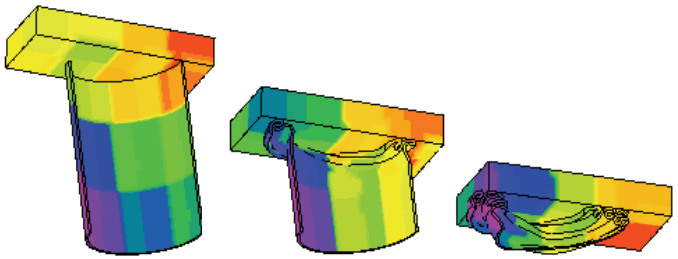




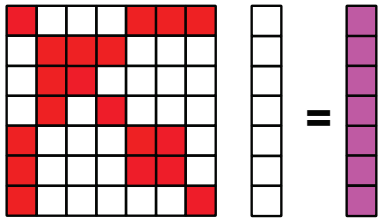
CSCAPES

Combinatorial Scientific Computing and Petascale Simulation

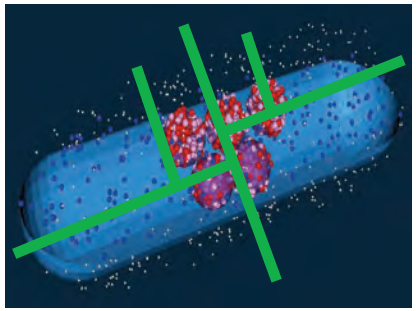
Develop toolkit of partitioners, dynamic load balancers, advanced sparse matrix reordering routines, and automatic differentiation procedures, generalizing currently available graph-based algorithms to hypergraphs



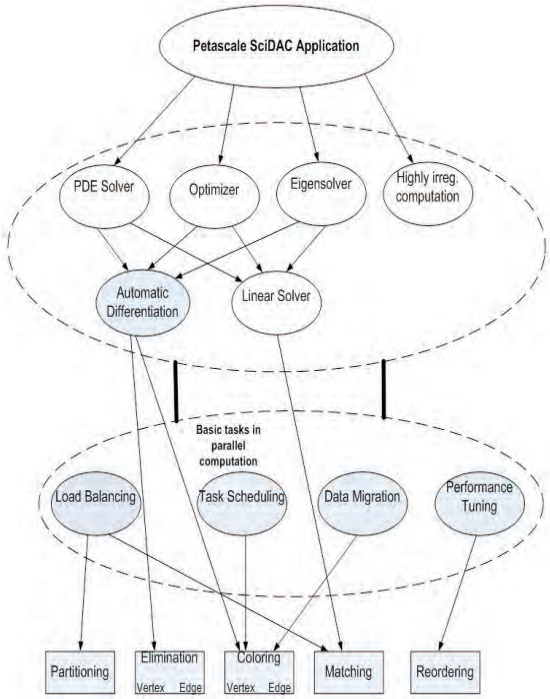
Contact detection



Linear solvers & preconditioners



Particle Simulations



The TOPS Center for Enabling Technology spans 4 labs & 5 universities

Our mission: Enable scientists and engineers to take full advantage of petascale hardware by overcoming the scalability bottlenecks traditional solvers impose, and assist them to move beyond "one-off" simulations to validation and optimization (~$32M/10 years)

Lawrence Livermore National Laboratory

Sandia National Laboratories

Columbia University

University of Colorado

University of Texas

Southern Methodist University

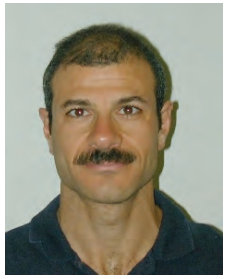

Adams


Baker


Cai


Demmel


Falgout


Ghattas


Heroux


Hu


Kaushik


Keyes


Knepley


Li


Manteuffel


McCormick


McInnes


Moré


Munson


Ng


Reynolds


Rouson


Salinger


Smith


Woodward


C. Yang


U. Yang


Zhang

DOE CSGF

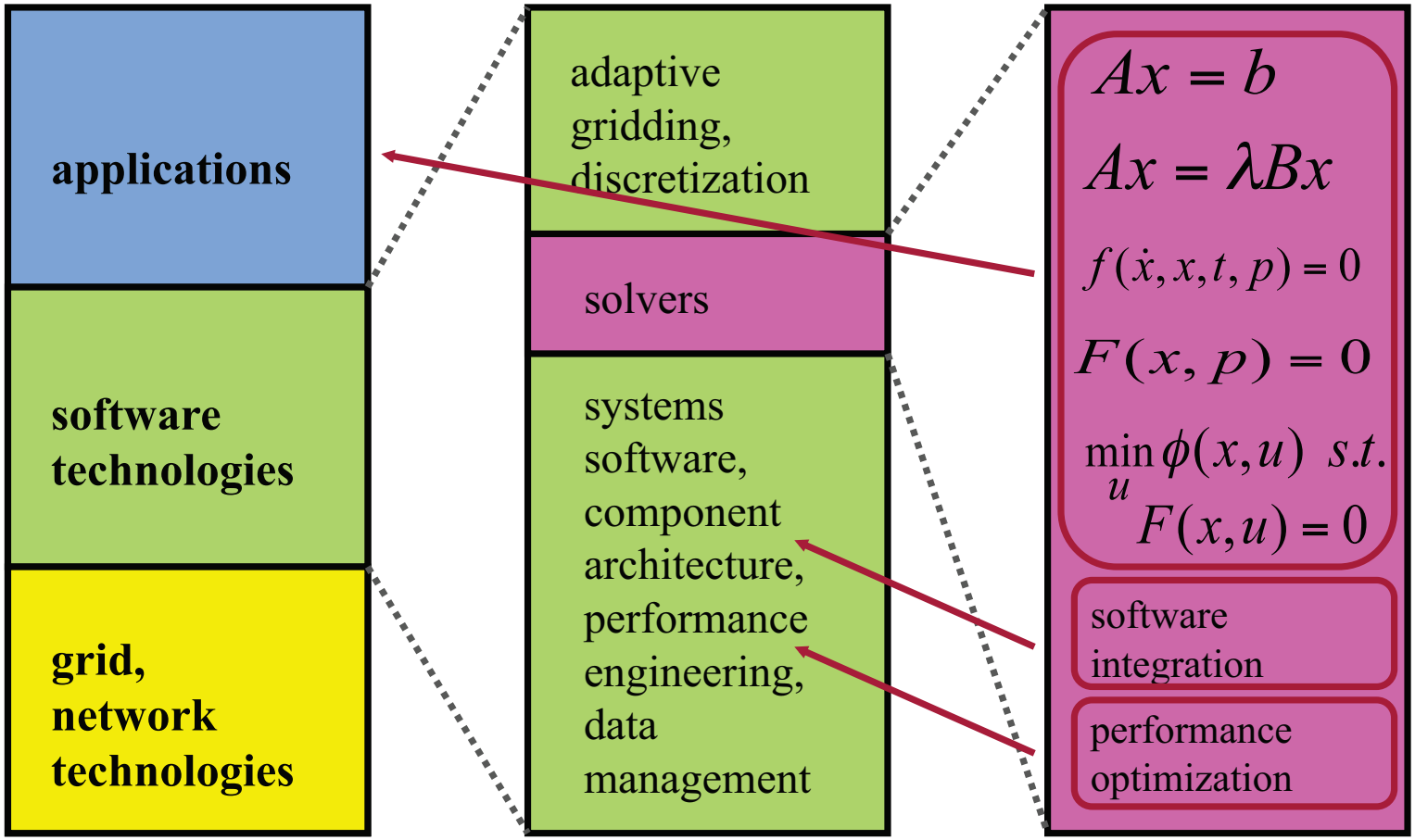
Faces of TOPS

TOPS
Terascale Optimal PDE Simulations

TOPS includes several elements

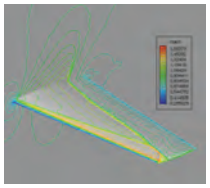
- **Propagation of "best practices" for solvers**
  - **fairly low-hanging fruit, throughout the Office of Science community and beyond (*lots* of this)**
- **First-time enablement of a simulation capability from better solvers**
  - **in hard-won collaborations, with specific priority Office of Science apps (*some* of this)**
- **Leading-edge exploration of extreme scaling**
  - **where the solver may lead or follow the physics, with ready apps (a *little* of this)**
- **Algorithmic research and implementation development to support these objectives**
  - **driven by application requirements and hardware trends (*as needed*)**

SciDAC
Scientific Discovery through Advanced Computing

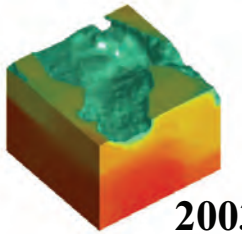


TOPS software has taken applications to the architectural leading edge

- **TOPS software at the heart of three Gordon Bell "Special" Prizes**



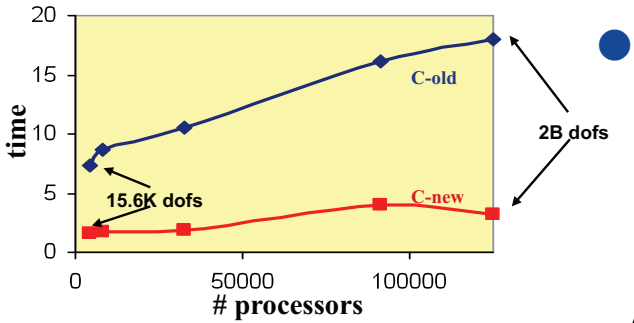
1999 fluids



2003 seismic

2004 mechanics

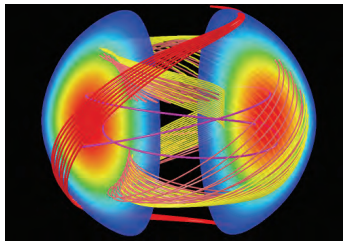
After new coarsening algorithm (red), nearly flat scaled speedup for Algebraic Multigrid



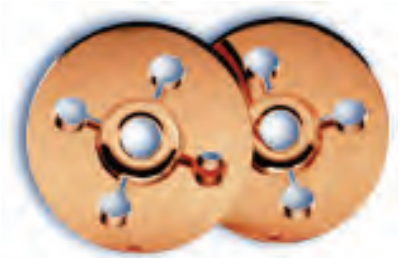
- **Scales to the edge of BlueGene/L (131,072 processors, 2B unknowns)**

- **Enabled numerous physics attainments in SciDAC-1**

~5X speedup of plasma fusion codes through linear solver replacement – like providing "next generation" computer

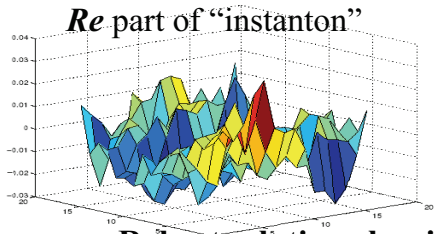


magneto-hydro-dynamics

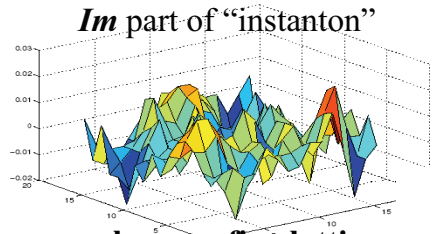


accelerator design

Prototype shape optimization capability



QCD



Robust solution algorithm for zero quark mass, fine lattices

DOE CSGF

Supported in part by DOE SciDAC TOPS, NSF PetaApps/Earth Sciences, and DOE NNSA PSAAP

2008 Gordon Bell finalist: hypre-enabled dynamic AMR in Earth mantle convection

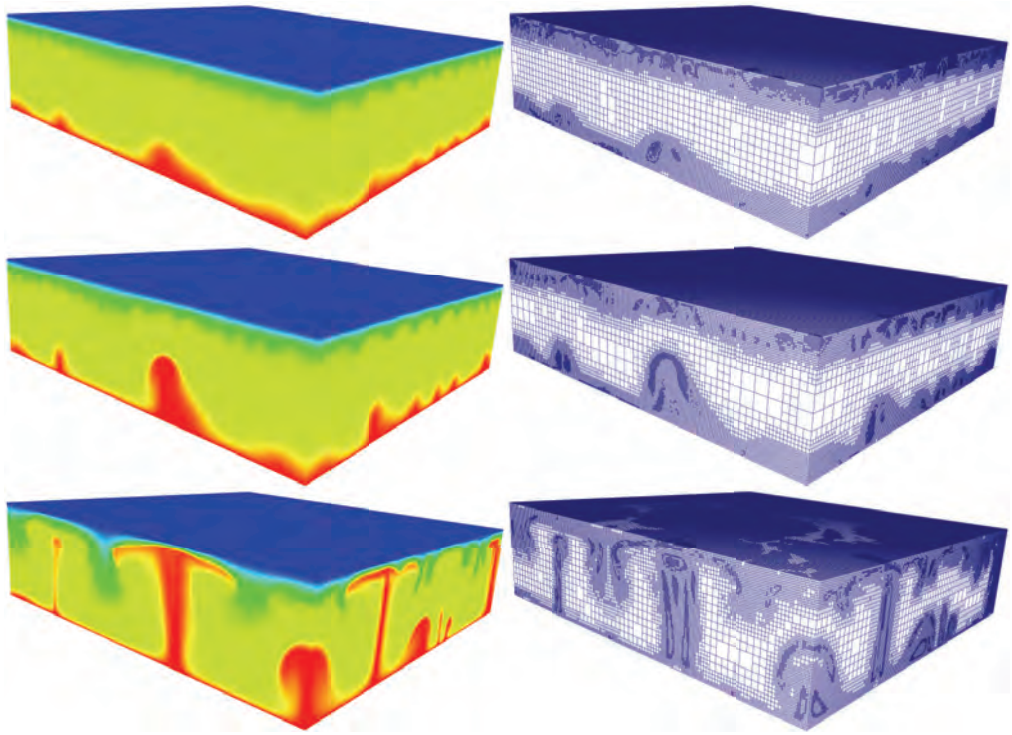
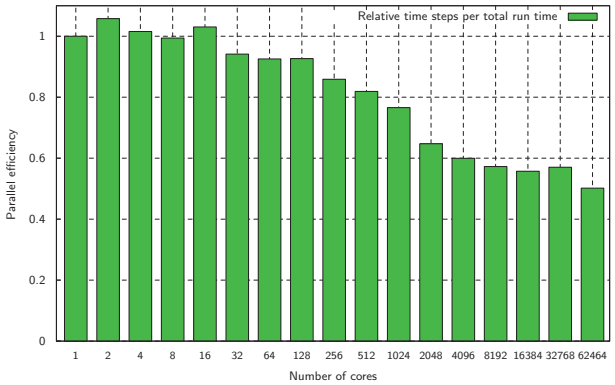
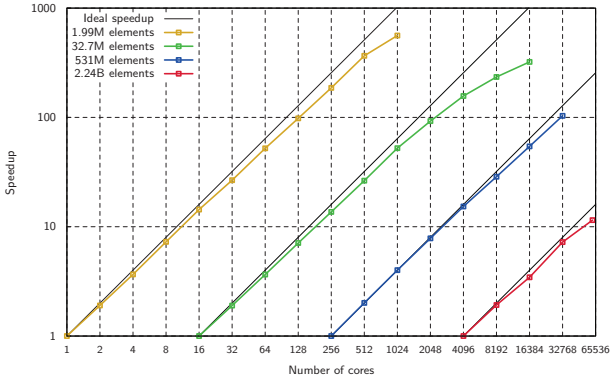


Illustration of parallel dynamic AMR for problem of modeling convection in Earth's mantle. The figure shows snapshots of the thermal field at three time instants (left column) and corresponding adapted meshes (right column). The mesh resolves the rising plumes as well as the instabilities at the top layer. The elements span levels 4 to 9 in octree depth.



Weak scaling of parallel AMR on advection-diffusion problem with ~131,000 elements per core. Results demonstrate 50% parallel efficiency over a range of 1 to 62,464 cores on TACC's Sun/AMD *Ranger* system. Largest problem has ~7.9B finite elements. AMR imposes <10% overhead over a static mesh solver.



Strong scaling of parallel AMR on advection-diffusion problem for small (yellow), medium (green), large (blue), and very large (red) problems sizes. Blue curve demonstrates nearly ideal strong scaling for 0.5B element problem over a range of 256 to 32,768 cores on *Ranger*.

c/o O. Ghattas (UTexas) *et al.*

TOPS themes

- **Resolution requirements of multiscale, multirate apps in SciDAC portfolio are insatiable, exacerbated by "curse of dimensionality":**
  - **each doubling of resolution requires 8-fold increase in memory and 16-fold increase in compute power to stay even, modulo adaptivity and implicitness**
  - **SciDAC apps are eager for the petascale and beyond**
  - **UQ, sensitivity and optimization add another "dimension" to the curse**
- **Complexity of "out-of-the-box" solvers is typically superlinear in problem size; optimal solvers are required ⇒**
  - **Towards *Optimal* Petascale Simulations (T*O*PS)**
- **At highest level, solver tasks are generic across many apps, amortizing software:**
  - $Ax = b$, $Ax = \lambda Bx$, $F(x) = 0$, $x' = f(x, p, t)$, $\min_p C(x, p)$ subject to $F(x, p) = 0$, $G(p) \geq 0$
- **For performance, customized tuning is required for each app**

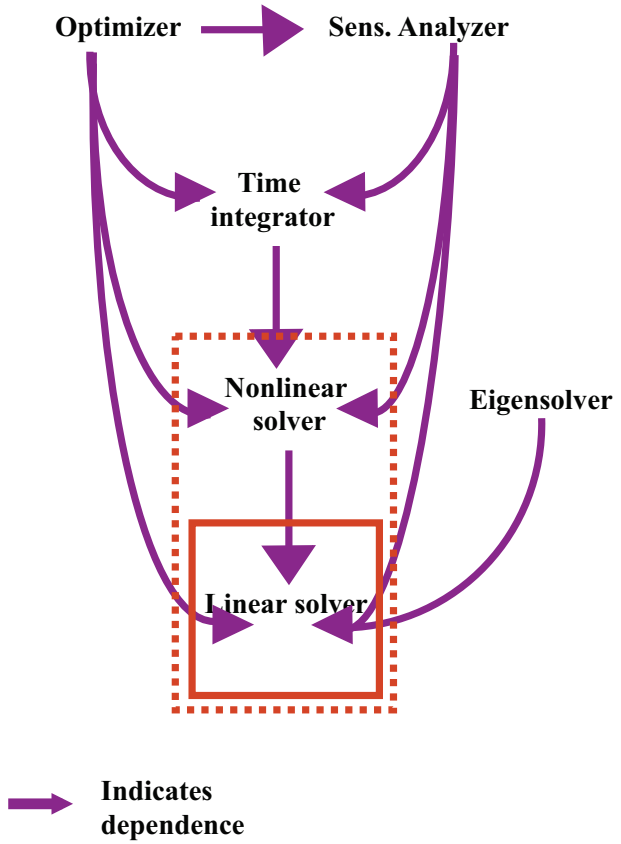
TOPS solvers are "green"

- Optimality is a necessary goal for indefinite scalability
  - of interest in order to follow DOE's exascale roadmap
  - TOPS is tracking the International Exascale Software Project (IESP)*
- Short of optimality, a constant factor improvement on a suboptimal algorithm today is also valuable
  - today's highest end systems are power-intensive and should spend as much time as possible on the physics
- Solving $Ax = b$ once by dense Gaussian elimination on LANL's Roadrunner for the Top 500 ScaLAPACK benchmark costs an estimated 2 barrels of oil [See 2009 WTEC report to NSF]
  - burning a barrel of oil generates about a half-ton of CO_2
  - just a factor of 2 in solver execution for a system this size would keep a half-ton of CO_2 out of the atmosphere
- Fruit is surprisingly "low hanging" on average; TOPS #1 product is communication of *existing* know-how


* See www.exascale.org

TOPS is building a toolchain of interoperable, proven solver components

- We aim to carry users from "one-off" *solutions* to the full scientific agenda of *sensitivity*, *stability*, and *optimization* (from heroic *point studies* to systematic *parametric studies*) all in one software suite
- TOPS solvers are nested, from applications-hardened linear solvers outward, leveraging common distributed data structures
- Communication and performance-oriented details are hidden so users deal with mathematical objects throughout



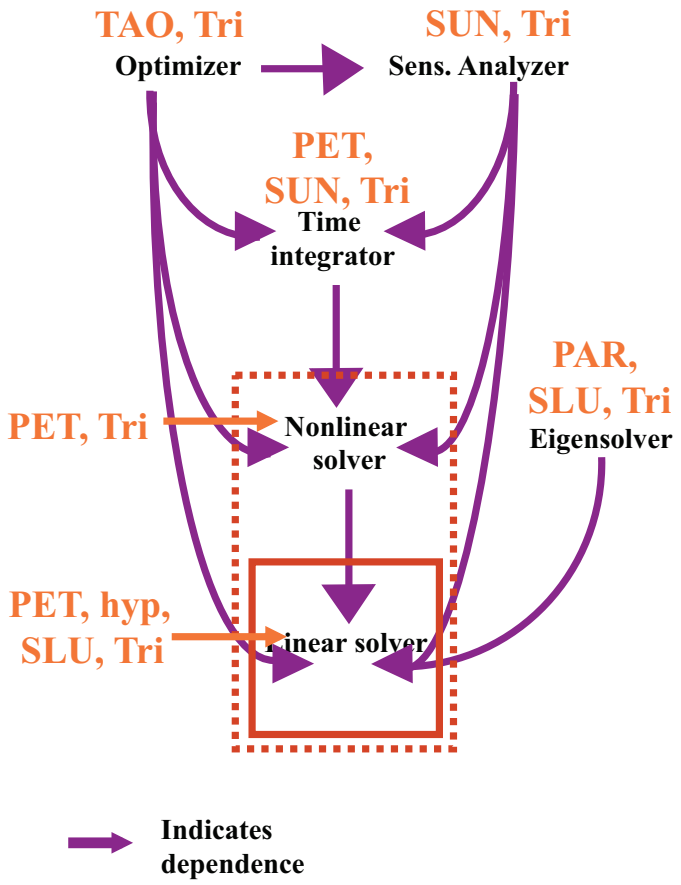
Software stack supported by TOPS

- TOPS features these trusted packages, whose principal functions are added to the chart at the right:

  hypre, [PARPACK], PETSc,
  SUNDIALS, SuperLU,
  TAO, Trilinos

  These are in use and actively debugged in dozens of high-performance computing environments, in dozens of applications domains, by thousands of user groups around the world

- TOPS maintains about *half* of the software presented at the ACTS toolkit tutorials



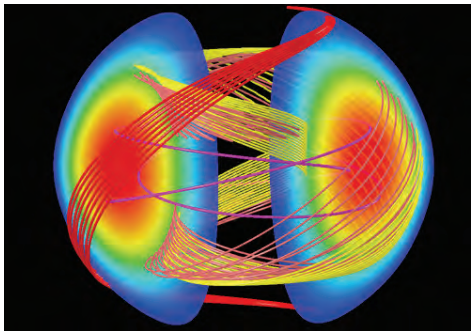
TOPS goals

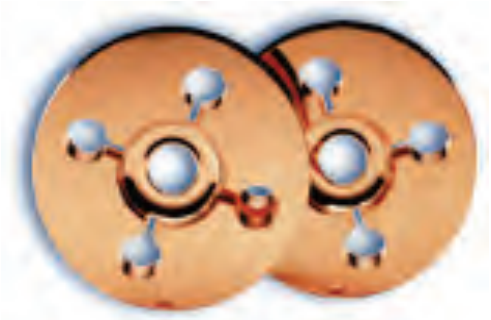
- Develop, demonstrate, and disseminate solver software
- Focus on systems modeled by PDEs
- Exploit mathematical structure for optimality
- Deploy computational mathematicians to specific SciDAC projects and help others as much as possible
- Raise standards for scientific software engineering
  - interface design
  - test harness
  - distributed development
  - plug-and-play interoperability
  - portability
  - extensibility
- Build large user/developer community, in and beyond SciDAC

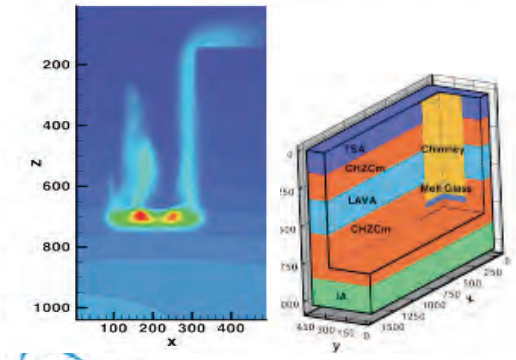

Desiderata for TOPS solver interface

- **Simplicity**
  - as few required concepts as possible, implementation details hidden
- **Generality**
  - richness of instances of each concept
- **Language independence**
  - callable from anything users want, but easy for TOPS to maintain
- **High performance**
  - no forced provision or recopying of data beyond what best method needs
- **Extensibility**
  - something designers can live with as new algorithms come along and have to be integrated underneath

Some applications requiring scalable solvers – in conventional and progressive ways







- **Magnetically confined fusion**
  - **Poisson problems**
  - ***nonlinear coupling of multiple physics codes***
- **Accelerator design**
  - **Maxwell eigenproblems**
  - ***shape optimization subject to PDE constraints***
- **Porous media flow**
  - **div-grad Darcy problems**
  - ***parameter estimation***

presenting symptoms

actual ailments

A *leitmotif:* nonlinear implicitness

- Virtually of the desirable developments in PDE-based simulation are achievable given first the ability to approximate the inverse action of the Jacobian matrix of the PDE with respect to the state variables, on an arbitrary vector
- A key is to take the original PDE solver nonlinearly implicit
  - often legacy codes are explicit, partially implicit with each equation, or fully implicit within each equation but decoupled
  - these are starting points for nonlinearly implicit formulations

Explicit methods can be unphysically oscillatory – nonlinear example ("profile stiffness")

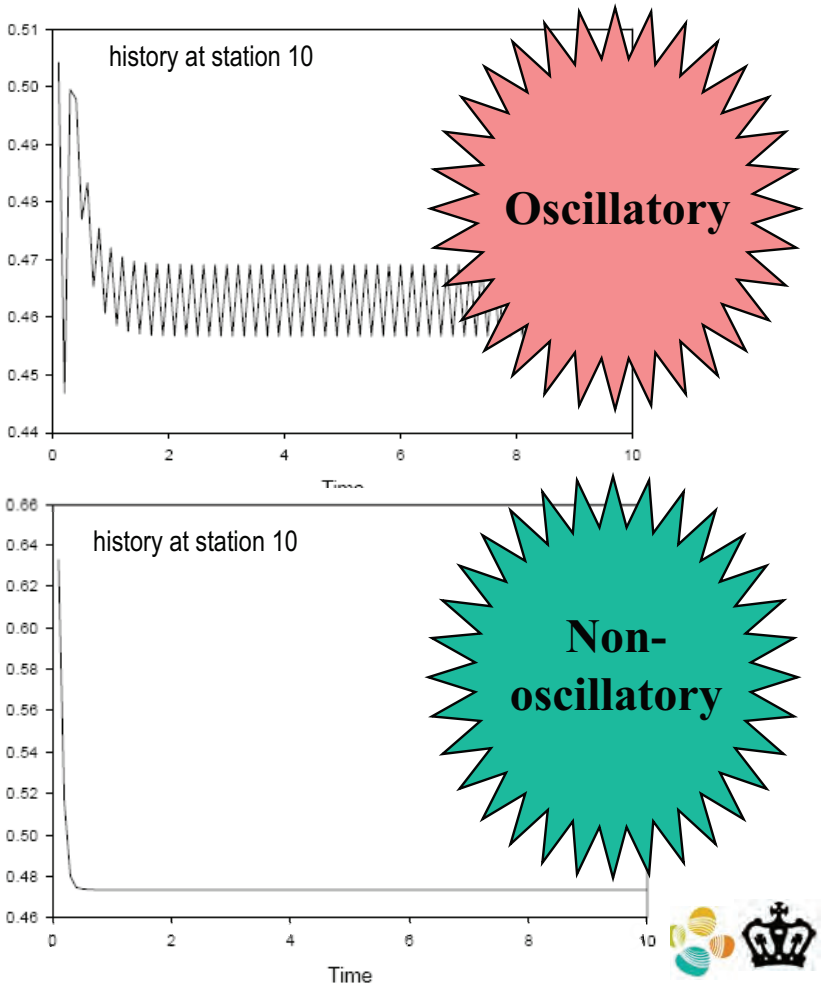
$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(\alpha(x,u_x)\frac{\partial u}{\partial x}\right); \quad \alpha \equiv x\kappa(u_x); \quad \kappa(s) \equiv \begin{cases} \kappa_0, & |s| \le s_{crit} \\ \kappa_0 + \kappa_1(|s| - s_{crit})^{1/2}, & |s| > s_{crit} \end{cases}$$

Linearly implicit, nonlinearly explicit:

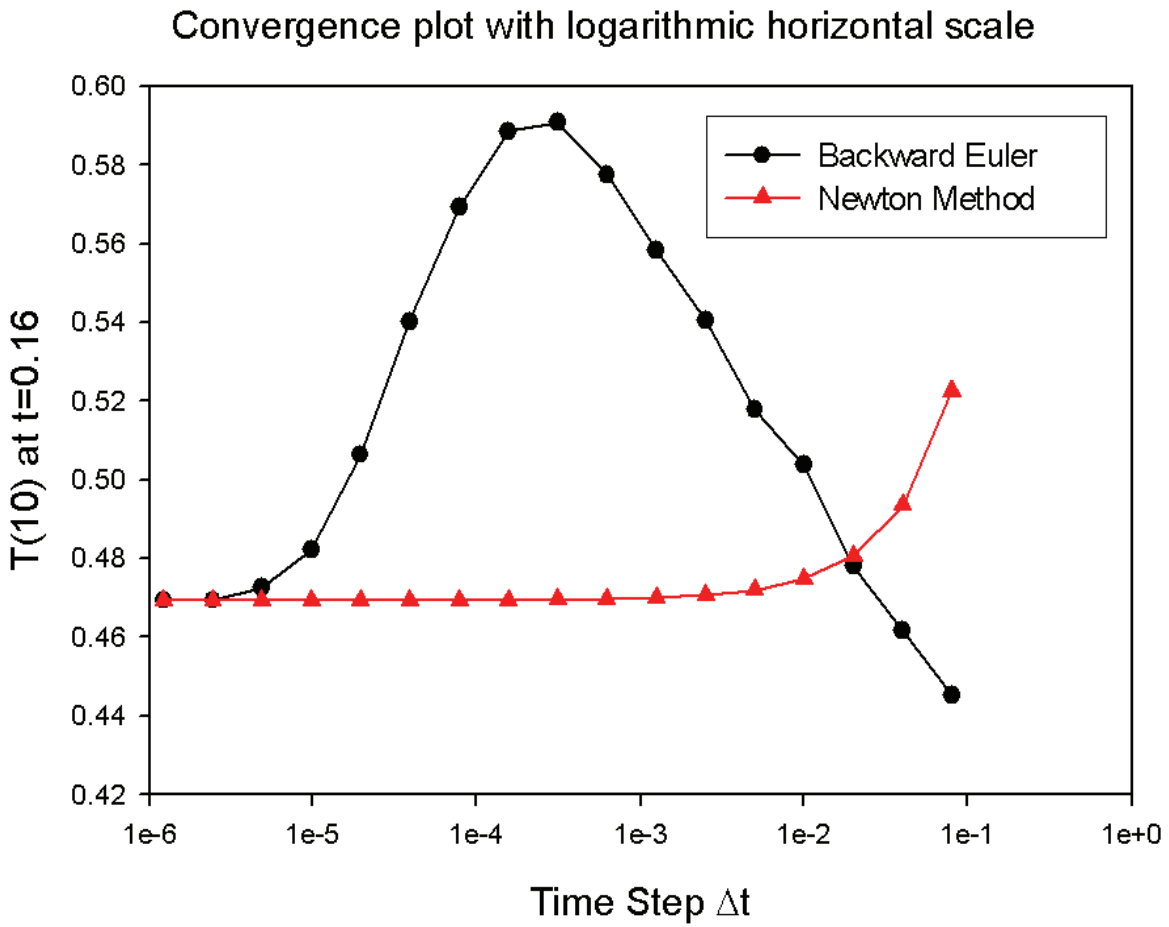
$$U_j^{n+1} - \nu[\alpha_{j+1/2}^{n}(U_{j+1}^{n+1} - U_j^{n+1}) - \alpha_{j-1/2}^{n}(U_j^{n+1} - U_{j-1}^{n+1})] = U_j^n$$

Linearly *and* nonlinearly implicit:

$$U_j^{n+1} - \nu[\alpha_{j+1/2}^{n+1}(U_{j+1}^{n+1} - U_j^{n+1}) - \alpha_{j-1/2}^{n+1}(U_j^{n+1} - U_{j-1}^{n+1})] = U_j^n$$



Timesteps for equivalent accuracy – GLF23 with gradient-dependent diffusivity



Example from fusion collaboration: for sufficiently small timestep, the nonlinearly implicit and linearly implicit with lagged diffusivity converge on the same result, but the nonlinear implicit permits timesteps 10^4 times larger with same accuracy

c/o Steve Jardin, PPPL

However –
implicit methods can be unruly and expensive

	Explicit	Naïve Implicit
Reliability	robust when stable	uncertain
Performance	predictable	data-dependent
Concurrency	$O(N)$	limited
Synchronization	once per step	many times per step
Communication	nearest neighbor*	global, in principle
Workspace	$O(N)$	$O(N^w)$, e.g., w=5/3
Complexity	$O(N)$	$O(N^c)$, e.g., c=7/3

* plus the estimation of the stable step size

Examples of scale-separated features of multiscale problems, ripe for implicitness

- Gravity surface waves in global climate
- Alfvén waves in tokamaks
- Acoustic waves in aerodynamics
- Fast transients in detailed kinetics chemical reaction
- Bond vibrations in protein folding (?)

Explicit methods are restricted to marching out the long-scale dynamics on short scales. Implicit methods can "step over" or "filter out" with equilibrium assumptions the dynamically irrelevant short scales, ignoring stability bounds. (Accuracy bounds must still be satisfied; for long time steps, one can use high-order temporal integration schemes!)

The common practice of operator splitting can destabilize multiphysics

- **Model problem** $\dot{u} = -\lambda u + u^2, \quad u(0) = u_0, \quad t > 0$
- **Exact solution**

$$u(t) = \frac{u_0 \exp(-\lambda t)}{1 + \frac{u_0}{\lambda}(\exp(-\lambda t) - 1)}$$

Well defined for all time if $\lambda > u_0$

- **Numerical approx.** $U_k \approx u(t_k), \quad t_k = k\Delta t, \quad k = 0, 1, \ldots$
- **Phase 1 ("R")** $\dot{u}_R = u_R^2, \quad u_R(t_k) = U_k, \quad t_k < t \le t_{k+1}$
- **Phase 2 ("D")** $\dot{u}_D = -\lambda u_D, \quad u_D(t_k) = u_R(t_{k+1}), \quad t_k < t \le t_{k+1}$
- **Overall advance** $U_{k+1} = u_D(t_{k+1})$
- **Phase 1 solution**

$$u_R(t) = \frac{U_k}{1 - U_k(t - t_k)}$$

- **Phase 2 solution** $u_D(t) = u_R(t_{k+1}) \exp(-\lambda(t - t_k))$
- **Overall advance**

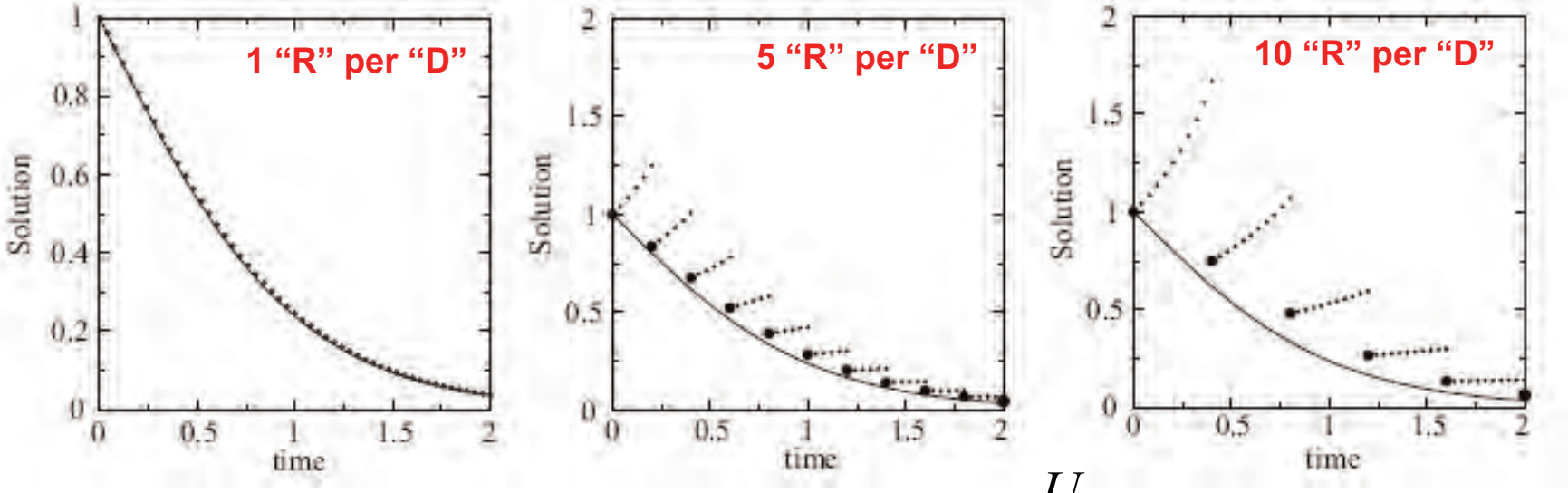
$$U_{k+1} = U_k \frac{\exp(-\lambda \Delta t)}{1 - U_k \Delta t}$$

Can blow up in finite time!

DOE CSGF

The common practice of operator splitting can destabilize multiphysics, cont.

- **Example from *Estep et al.* (2007),** $\lambda = 2,\ u_0 = 1$
- **50 time steps, phase 1 subcycled inside phase 2**



$$\dot{u} + \lambda u = u^2, \quad u(0) = u_0, \quad t > 0$$

$$u(t) = \frac{u_0 \exp(-\lambda t)}{1 + \frac{u_0}{\lambda}(\exp(-\lambda t) - 1)}$$

$$u_R(t) = \frac{U_k}{1 - U_k(t - t_k)}$$

$$u_D(t) = u_R(t_{k+1}) \exp(-\lambda(t - t_k))$$

$$U_{k+1} = \frac{U_k}{1 - U_k \Delta t} \exp(-\lambda \Delta t)$$

This is a prototype for a reaction-diffusion PDE

$$u_t - a u_{xx} = u^2, \quad u(0,x) = u_0(x), \quad t > 0$$

- Diffusive time-scale is constant in time (for each wave number), whereas reactive time-scale changes with solution magnitude
- Besides opening the possibility of finite-time blow-up for a problem that is well defined for all time, operator splitting leaves a first-order error, independent of integration errors for the two phases
- Splitting a single equation is just the simplest example
- Other types of multiphysics (multiple equations in one domain, multiple domains) similarly treatable (see *D. Estep, et al.* 2007)

Another use for a full Jacobian -- adjoints "probe" uncertain problems efficiently

- "Forward" operator equation $L u_k = f_k, k \text{ samples}$
- Desired functional of solution $\ell(u) = \langle \ell, u \rangle$
- Define adjoint operator $L^* v = g$

$$\langle L^* v, u \rangle \equiv \langle v, Lu \rangle$$

- If we can solve for v given ℓ $L^* v = \ell$
- Then desired output …

$$\ell(u_k) = \langle \ell, u_k \rangle = \langle L^* v, u_k \rangle = \langle v, Lu_k \rangle = \langle v, f_k \rangle$$

… reduces to an inner product for each forcing f

Inverse problems also benefit from implicit methods for the forward solver

- **Inverse problems can be formulated as PDE-constrained optimization problems**
  - **objective function (mismatch of model output and "true" output)**
  - **equality constraints (PDE)**
  - **possible inequality constraints, in addition**
- **Cast as nonlinear rootfinding problem**
  - **Form (augmented) Lagrangian**
  - **Take gradient of Lagrangian with respect to design variables, state variables, and Lagrange multipliers**
  - **Obtain large nonlinear rootfinding problem**
- **Solving with Newton requires Jacobian of gradient, or Hessian of Lagrangian**
  - **Major blocks are Jacobian of PDE system and its adjoint** ☑

DOE CSGF

Components of scalable solvers for PDEs

- **Subspace solvers**
  - elementary smoothers
  - incomplete factorizations
  - full direct factorizations
- **Global linear preconditioners**
  - Schwarz and Schur methods
  - multigrid
- **Linear accelerators**
  - Krylov methods
- **Nonlinear rootfinders**
  - Newton-like methods

alone unscalable; either too many iterations or too much fill-in

opt. combins. of subspace solvers

mat-vec algs.

vec-vec algs. + linear solves

Newton-Krylov-Schwarz: a PDE applications "workhorse"

$$F(u) \approx F(u_c) + F'(u_c)\delta u = 0$$
$$u = u_c + \lambda\, \delta u$$

$$J\delta u = -F$$
$$\delta u = \underset{x \in V \equiv \{F, JF, J^2F, \cdots\}}{\arg\min} \{Jx + F\}$$

$$M^{-1}J\delta u = -M^{-1}F$$
$$M^{-1} = \sum_i R_i^T (R_i J R_i^T)^{-1} R_i$$

Newton
nonlinear solver
asymptotically quadratic

Krylov
accelerator
spectrally adaptive

Schwarz
preconditioner
parallelizable

Nonlinear implicitness is an easy add-on to linear implicitness

- **Linear versus nonlinear problems**
  - **Solving linear systems often constitutes 90% of the running time of a large PDE simulation**
  - **The nonlinearity is often a fairly straightforward outer loop, in that it introduces no new types of messages or synchronizations not present in Krylov-Schwarz, and has overall many fewer synchronizations than the preconditioned Krylov method or other linear solver inside it**
- **We can wrap Newton, Picard, fixed-point or other iterations outside, linearize, and apply what we know**
- **We consider both Newton-outside and Newton-inside methods**

"Secret sauce" #1: iterative correction w/ each step $O(N)$

- The most basic idea in iterative methods for $Ax = b$

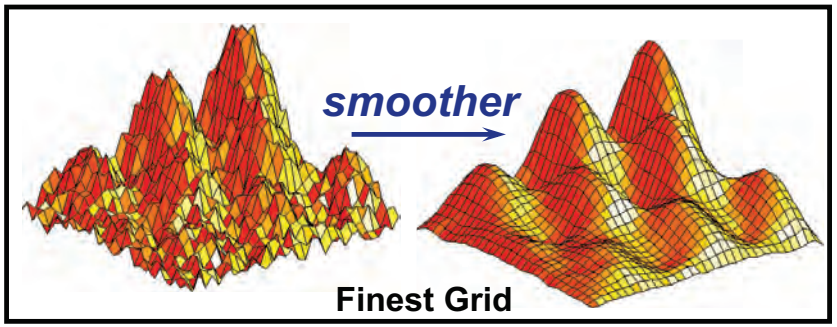
$$x \leftarrow x + B^{-1}(b - Ax)$$

- Evaluate residual accurately, but solve approximately, where B^{-1} is an approximate inverse to A
- A sequence of complementary solves can be used, e.g., with B_1 first and then B_2 one has

$$x \leftarrow x + [B_1^{-1} + B_2^{-1} - B_2^{-1} A B_1^{-1}](b - Ax)$$

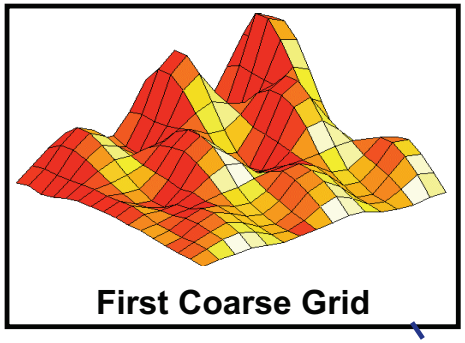
- Scale recurrence, e.g., with $B_2^{-1} = R^T (RAR^T)^{-1} R$, leads to *multilevel methods*
- Optimal polynomials of $(B^{-1}A)$ lead to various *preconditioned Krylov methods* (recall first lecture)


"Secret sauce" #2:
treat each error component in optimal subspace



A Multigrid V-cycle

Restriction
transfer from fine to coarse grid



coarser grid has fewer cells (less work & storage)

Prolongation
transfer from coarse to fine grid

Recursively apply this idea until we have an easy problem to solve

"Secret sauce" #3: skip the Jacobian

- In the Jacobian-Free Newton-Krylov (JFNK) method for $F(u) = 0$, a Krylov method solves the linear Newton correction equation, requiring Jacobian-vector products
- These are approximated by the Fréchet derivatives

$$J(u)v \approx \frac{1}{\varepsilon}[F(u + \varepsilon v) - F(u)]$$

(where ε is chosen with a fine balance between approximation and floating point rounding error) or automatic differentiation, so that the actual Jacobian elements are *never explicitly needed*

- One builds the Krylov space on a true $F'(u)$ (to within numerical approximation)

Carl Jacobi

Secret sauce #4: use the user's solver to precondition

- Almost any code to solve $F(u) = 0$ computes a residual and invokes some process to compute an update to u based on the residual
- Defines a weakly converging nonlinearly method

$$M : F(u^k) \mapsto \delta u$$

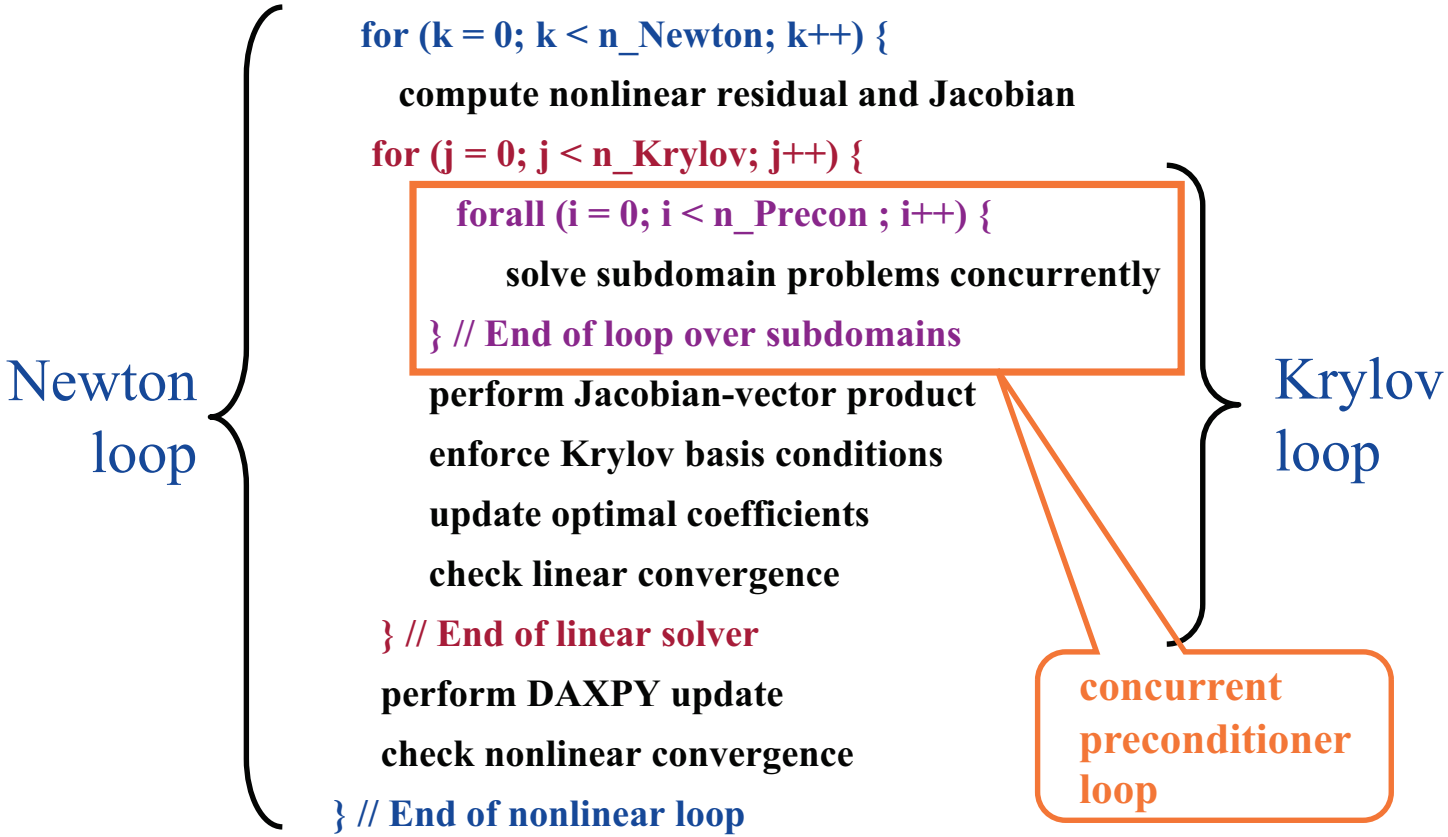
$$u^{k+1} \leftarrow u^k + \delta u$$

- M is, in effect, a preconditioner and can be applied directly within a Jacobian-free Newton context
- This is the "physics-based preconditioning" strategy discussed in the DOE "E^3 report" (2007); see D. Knoll, V. Mousseau and colleagues at INL for numerous examples

Scalable solvers for PDEs

- Linear preconditioners
  - Domain-decomposition methods
  - Multigrid
- Linear accelerators
  - Krylov methods
- Nonlinear rootfinders
  - Newton-like methods
- Hybrids (nonlinear Schwarz, FAS multigrid) have implications for multiphysics coupling, as well as domain-decomposed parallelism

Newton-Krylov-Schwarz



Outer loops (not shown): continuation, implicit timestepping, optimization

TOPS' wishlist for collaborations — "Asymptopia"

- **Engage at a higher-level than *Ax=b***
  - **implicit solvers directly on coupled nonlinear system**
- **Sensitivity analyses**
  - **validation studies**
- **Stability analyses**
  - **"routine" outer loop on steady-state solutions**
- **Optimization**
  - **parameter identification**
  - **design of facilities (accelerators, tokamaks, power plants, etc.)**
  - **control of experiments**

Seven leading questions TOPS asks users

Has your solver been unchanged for the past five or ten years?

Is your solver running at 1-10% of machine peak?

Do you spend more time in your solver than in your physics?

Is your discretization or model fidelity limited by the solver?

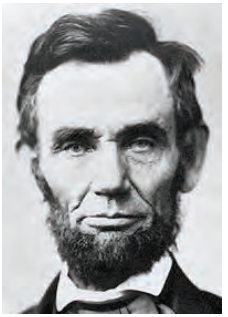
Is your time stepping limited by stability?

Are you running loops *around* your analysis code?

Do you care how sensitive to parameters your results are?

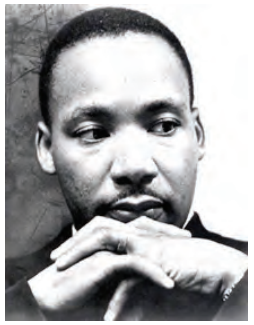
If the answer to any of these questions is "yes", you may be a customer!

TOPS is dedicated to the proposition that …



- **Not all problems are created equal**
  - **so a large variety of solvers should be callable from one interface**
- **Solver software can rarely be thrown over the wall**
  - **so we are committed to collaborations with applied users**
- **Discretization and solution rarely separate cleanly**
  - **so we are committed to collaborations with CET colleagues**
- **Desire for resolution will grow without bound**
  - **so we concentrate on solvers that scale well (in the "weak" sense)**
- **Solving the PDE well is only a beginning, not the end, in doing computational science**
  - **so we are providing a software "tool chain" of several links, which are implemented over common data structures and kernel functionality**

TOPS has a dream that users will…



- **Understand range of algorithmic options w/tradeoffs**

  ***e.g.*, memory *vs.* time, comp. *vs.* comm., inner iteration work *vs.* outer**

- **Try all reasonable options "easily"**

  without recoding or extensive recompilation

- **Know how their solvers are performing**

  with access to detailed profiling information

- **Intelligently drive solver research**

  ***e.g.*, publish joint papers with algorithm researchers**

- **Simulate *truly new physics* free from solver limits**

  ***e.g.*, finer meshes, complex coupling, full nonlinearity**

Expectations TOPS has of users

- Be willing to experiment with novel algorithmic choices – optimality is *rarely* achieved beyond model problems without interplay between physics and algorithmics!
- Adopt flexible, extensible programming styles in which algorithmic and data structures are not hardwired
- Be willing to let us play with the real code you care about, but be willing, as well to abstract out relevant compact tests
- Be willing to make concrete requests, to understand that requests must be prioritized, and to work with us in addressing the high priority requests
- If possible, *profile* before seeking help


TOPS software URLs

- **Hypre – scalable preconditioners**
  - https://computation.llnl.gov/casc/linear_solvers/sls_hypre.html
  - acts.nersc.gov/hypre
- **PETSc – scalable nonlinear and linear solvers**
  - www.mcs.anl.gov/petsc
- **SUNDIALS – scalable ODE and nonlinear solvers, forward and adjoint sensitivities**
  - https://computation.llnl.gov/casc/sundials
  - acts.nersc.gov/sundials
- **SuperLU – parallel direct sparse LU methods**
  - acts.nersc.gov/superlu
- **TAO – scalable general-purpose optimizers**
  - www.mcs.anl.gov/tao
- **Trilinos – scalable nonlinear, linear, and eigen solvers, with embedded nonlinear analysis tools**
  - trilinos.sandia.gov

TOPS usage outside of SciDAC proper

Articles, proceedings, theses in:

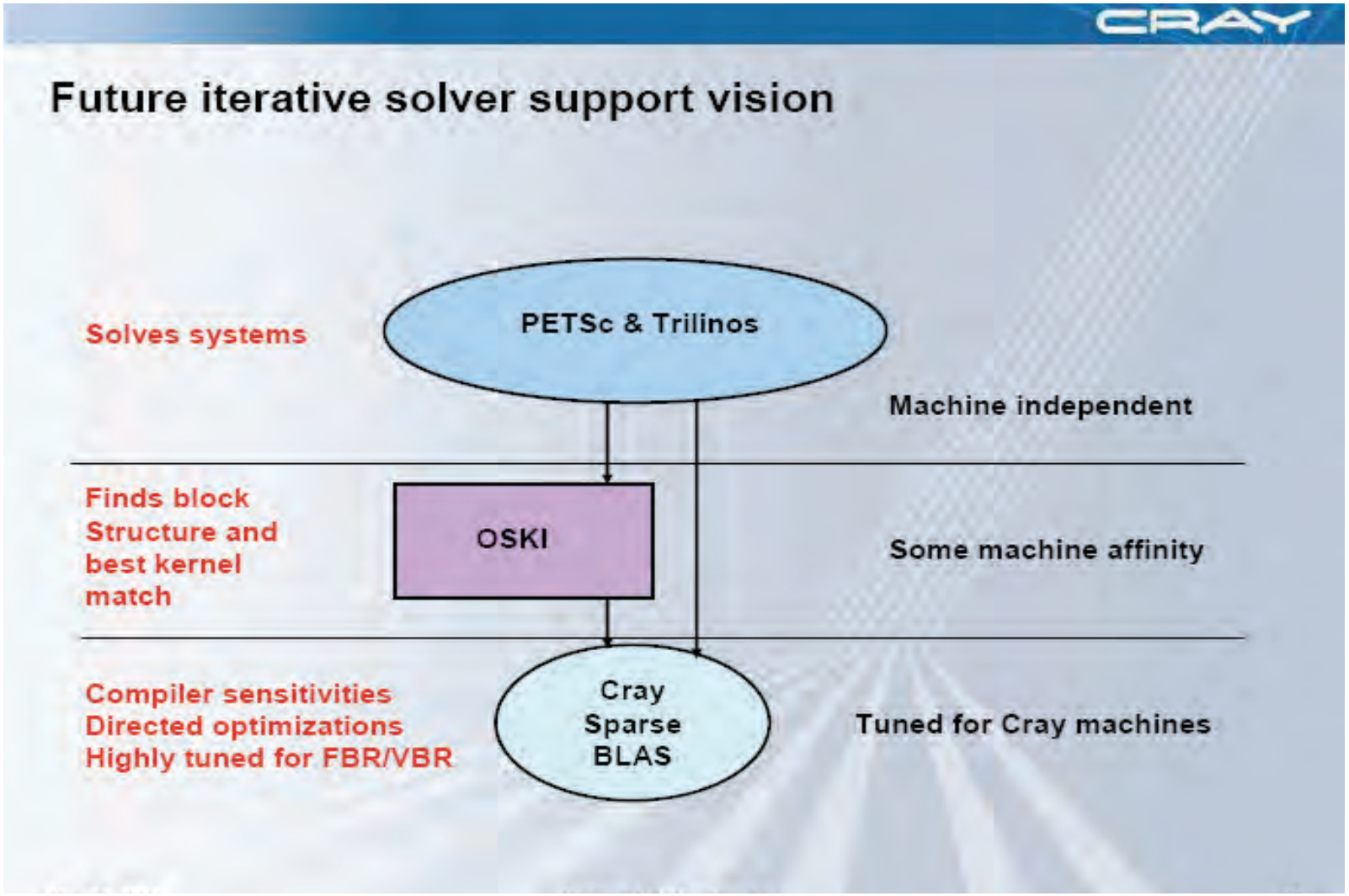
- Astronomy
- Biomechanics
- Chemistry
- Climate
- Cognitive Sciences
- Combustion
- Economics
- Electrical Engineering
- Finance
- Geosciences
- Hydrodynamics
- Materials Science
- Mechanics
- Medical
- Micromechanics/Nanotechnology
- Numerical Analysis
- Optics
- Porous Media
- Shape Optimization

Widely distributed software:

- Cray LibSci®
- deal.II (2007 Wilkinson Prize)
- Dspice
- EMSolve
- FEMLAB®
- FIDAP®
- GlobalArrays
- HP Mathematical Library®
- IMSL ®
- libMesh
- Magpar
- Mathematica®
- NAG ®
- NIKE
- Prometheus
- SCIRun
- SciPy
- SLEPc
- Snark

Thousands of groups around the world use TOPS software without directly collaborating

Cray plans to support TOPS software



A project like TOPS is needed in SciDAC because …

- **SciDAC applications are otherwise solver-bound**

  ***e.g.*, 90-95% of execution time in solver, limited to low dimensions**

- **SciDAC ambitions are too low, focused too near**

  concentrated on getting a few big runs, without enough "doing science," since iteration over the "forward" problem is costly

- **SciDAC community codes are hard to keep current**

  slow process to implement new algorithms, to port to new machines

- **SciDAC CS CETs need good stepping stone to apps**

  solvers are good target for research in components and performance

- **SciDAC Math CETs need good solvers, *too!***

  from scalable Poisson solves to mesh optimization, other CETs have subproblems for TOPS, for which they are not otherwise funded