

Robust and High Performance Tools for
Scientific Computing
The DOE ACTS Collection



Tony Drummond and Osni Marques
UC Berkeley - CS-267
October 10, 2002



What is the DOE ACTS Collection?



<http://acts.nersc.gov>

- Advanced CompuTational Software
- Tools for developing parallel applications
 - Developed (primarily) at DOE Labs
 - Separate projects originally
 - ~ 20 tools
- ACTS is an "umbrella" project
 - Leverage numerous independently funded projects
 - Collect tools in a toolkit



ACTS: *Project Goals*



- Extended support for *experimental software*
- Make ACTS tools available on DOE computers
- Provide technical support (*acts-support@nersc.gov*)
- Maintain ACTS information center
(*http://acts.nersc.gov*)
- Coordinate efforts with other supercomputing centers
- Enable large scale scientific applications
- Educate and train



Related Activities



- **Software Repositories:**
 - **Netlib:** <http://www.netlib.org>
 - **HPC-Netlib:** <http://www.nhse.org/hpc-netlib>
 - **National HPCC Software Exchange NHSE:** <http://www.nhse.org>
 - **Guide to Available Mathematical Software:** <http://gams.nist.gov>
 - **MGNet:** <http://www.mgnet.org>
 - **NEOS:** <http://www-fp.mcs.anl.gov/otc/Guide>
 - **OO Numerics:** <http://oonumerics.org/oon>
- **Portable timing routines, tools for debugging, compiler technologies:**
 - **Ptools:** <http://www.ptools.org>
 - **Center for Programming Models for Scalable Parallel Computing:** <http://www.pmodels.org>
- **Education:**
 - **Computational Science Educational Project:** <http://csep1.phy.ornl.gov>
 - **UCB's Applications of Parallel Computers:** http://www.cs.berkeley.edu/~demmel/cs267_Spr99
 - **MIT's Applied Parallel Computing:** <http://www.mit.edu/~cly/18.337>
 - **Dictionary of algorithms, data structures and related definitions:** <http://www.nist.gov/dads>



Why is ACTS unique?



- Extended support for tools
- Accumulates the expertise and user feedback on the use of the software tools and scientific applications that used them:
 - independent software evaluations
 - participation in the developer user groups e-mail list
 - presentation of a gallery of applications
 - leverage between tool developers and tool users
 - workshops and tutorials
 - tool classification



ACTS: *levels of support*



- ***High***
 - Intermediate level
 - Tool expertise
 - Conduct tutorials
- ***Intermediate***
 - Basic level
 - Provide a higher level of support to users of the tool
- ***Basic***
 - Basic knowledge of the tools
 - Help with installation
 - Compilation of user's reports (acts-support@nslsc.gov)

<http://acts.nersc.gov>



The DOE ACTS Collection



[Tools](#)

[News](#)

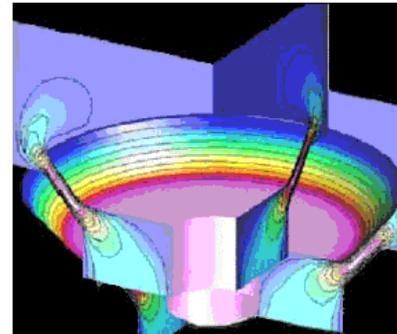
[Project](#)

[Center](#)

[Search](#)

The DOE ACTS (Advanced Computational Software) Collection is a set of DOE-developed software tools that make it easier for programmers to write high performance scientific applications for parallel computers. This site is the central information center for the ACTS Collection and is brought to you by NERSC and the [Mathematical, Information, and Computational Sciences](#) (MICS) Division of DOE. Correspondence regarding the collection (including requests for support) should be directed to acts-support@nersc.gov.

click on the image below to see other applications that have benefited from ACTS Tools



The image shows pressure and velocity around a moving valve in a diesel engine. The flow here was found as part of a CFD effort to simulate the flow within the complex 3D geometry of a diesel engine. The computation was carried out using the Overture Framework and the PADRE library for parallel data distribution.

[Tools](#)

[News](#)

[Project](#)

[Center](#)

[Search](#)

Tool descriptions, installation details, examples, etc

Agenda, accomplishments, conferences, releases, etc

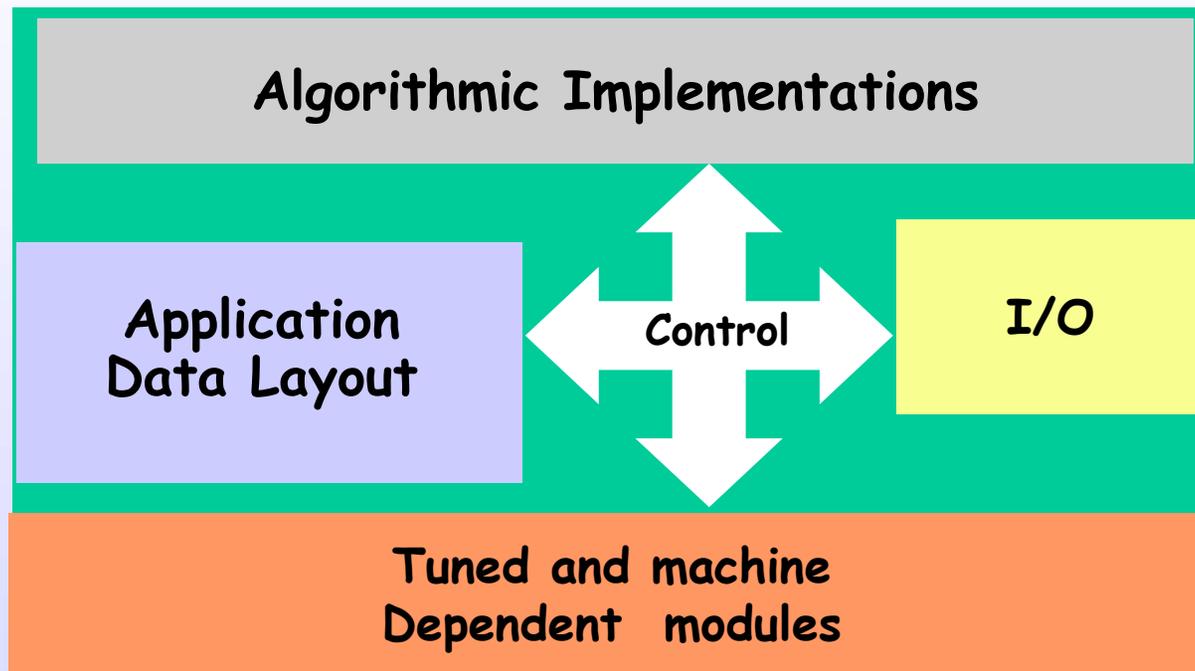
Goals and other relevant information

Points of contact

Search engine



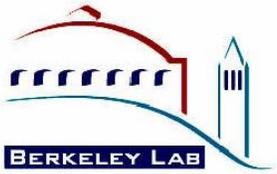
Large Scientific Codes: *A Common Programming Practice*





Using today's hardware to tackle today's Grand Challenges

Q. Why is it **still** difficult to obtain
High Performance?



Some common and *interesting* answers



- Technology
- Memory latency
- Algorithms
- Programming Practices

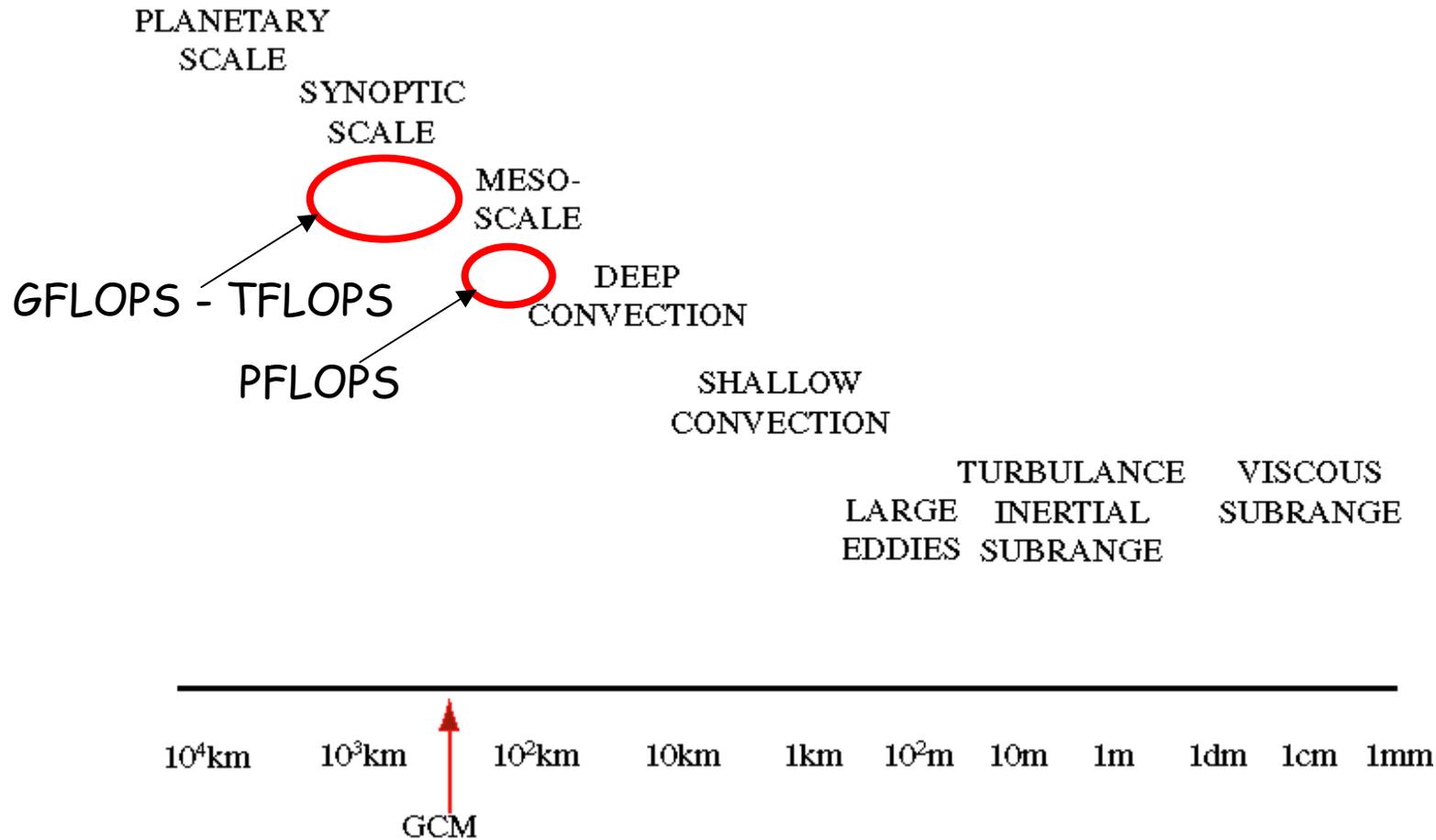
•
•
•



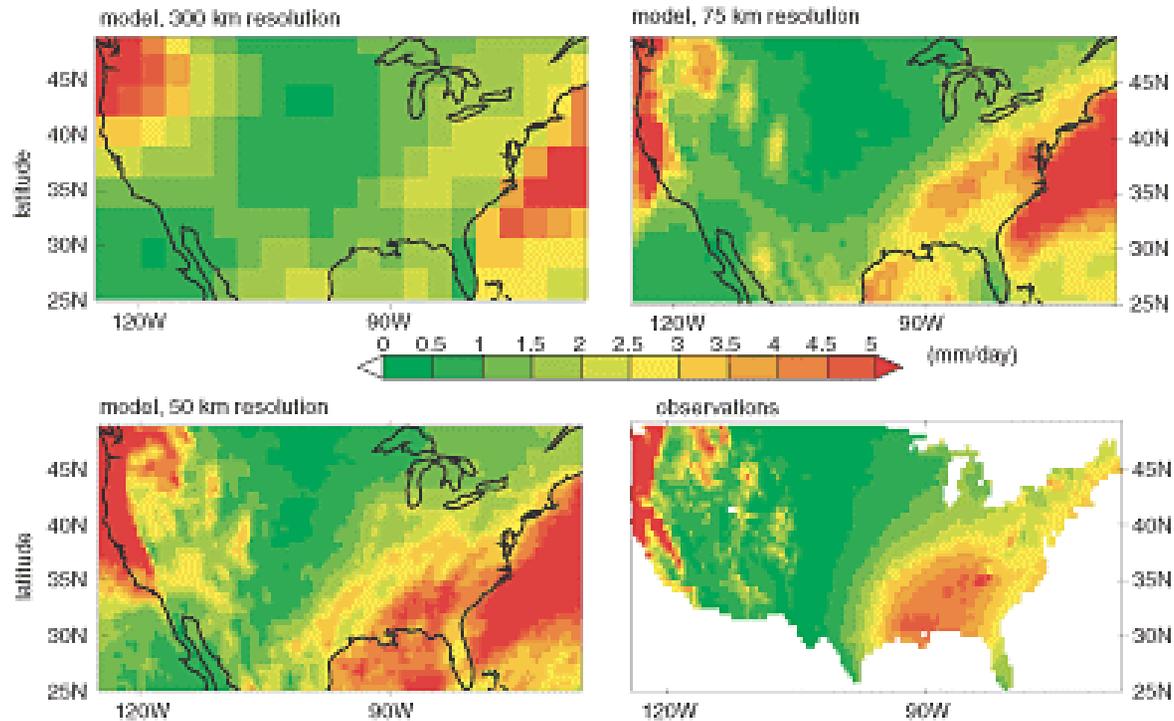
Motivation - Example I



SPECTRUM OF ATMOSPHERIC PHENOMENA



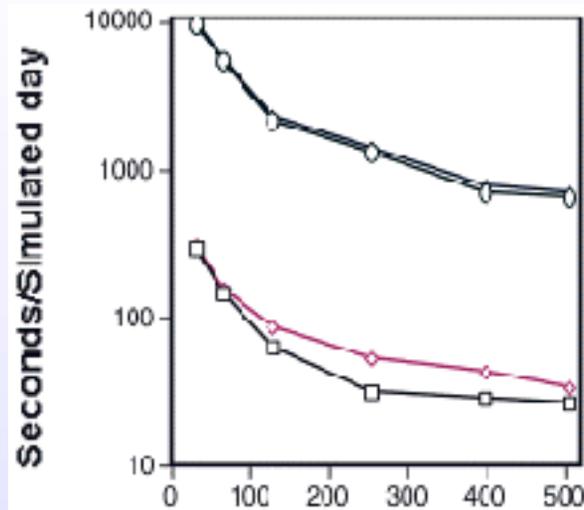
Motivation - Example I



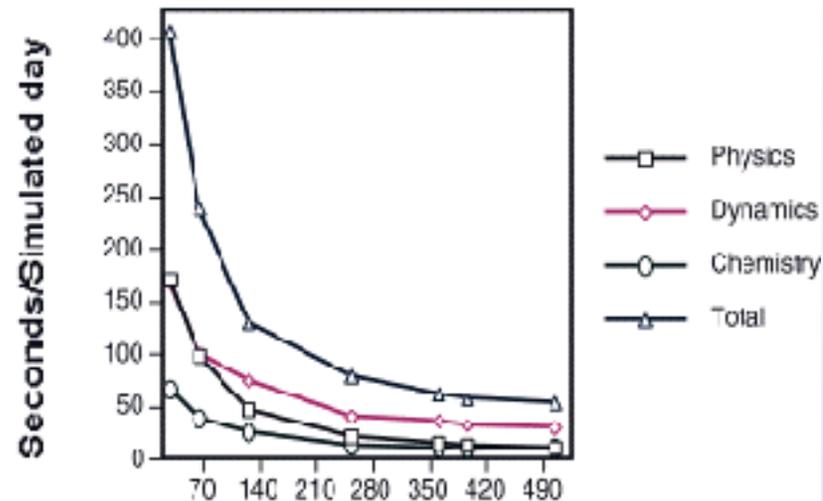
Duffy et. al.,
Lawrence Livermore National Laboratory

- CCM3 - spectral truncations of T170 and T239
- 50 Km spatial resolution is 32 times more grid cells and takes roughly **200 times longer** to run

Motivation - Example II



AGCM/ACM
2.5 long x 2 lat, 30 layers
25-chemical species



AGCM/ACM
2.5 long x 2 lat, 30 layers
2-chemical species

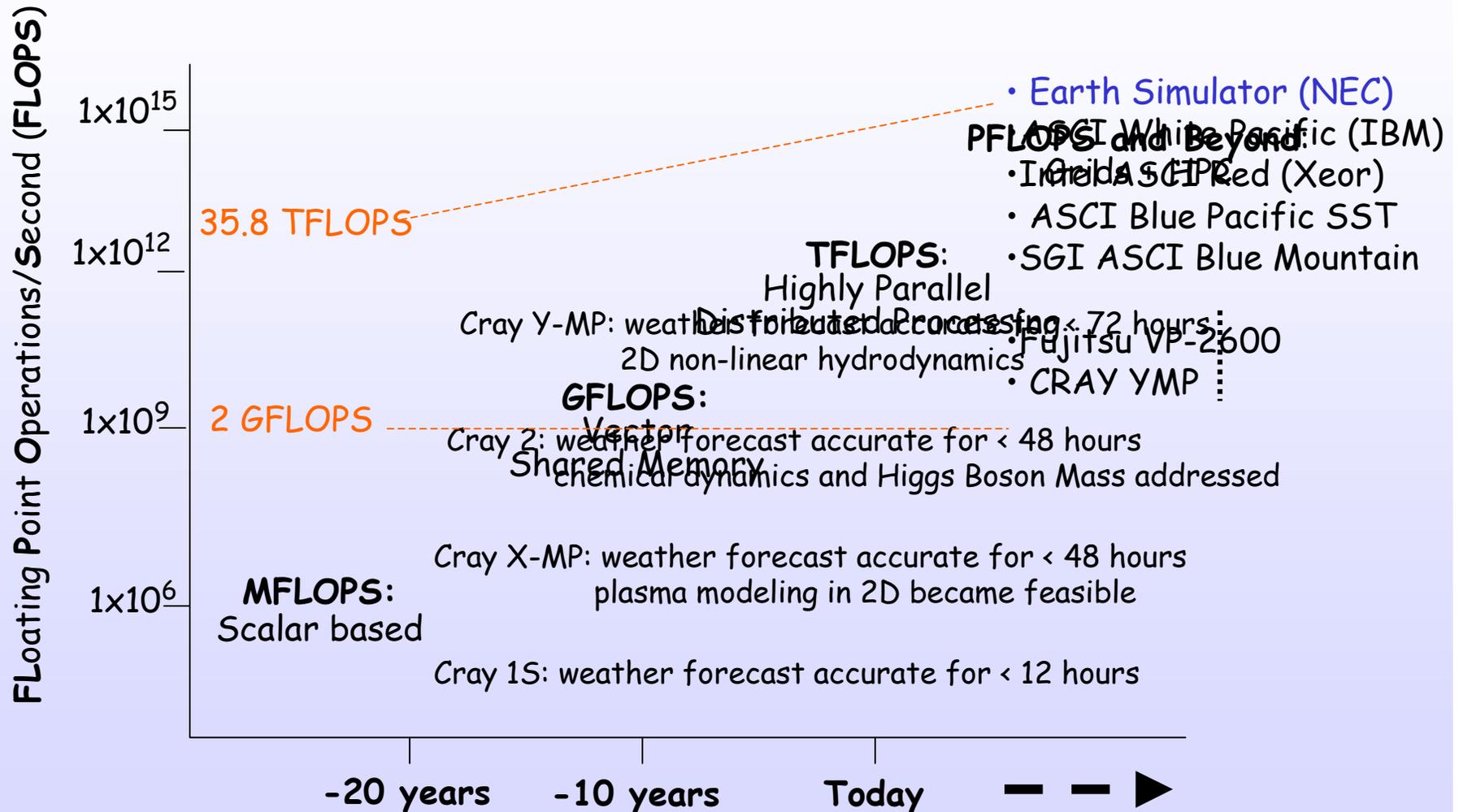
- Non-linear demand for resources (CPU - Memory)
- Multi-disciplinary application is more demanding



The Hardware

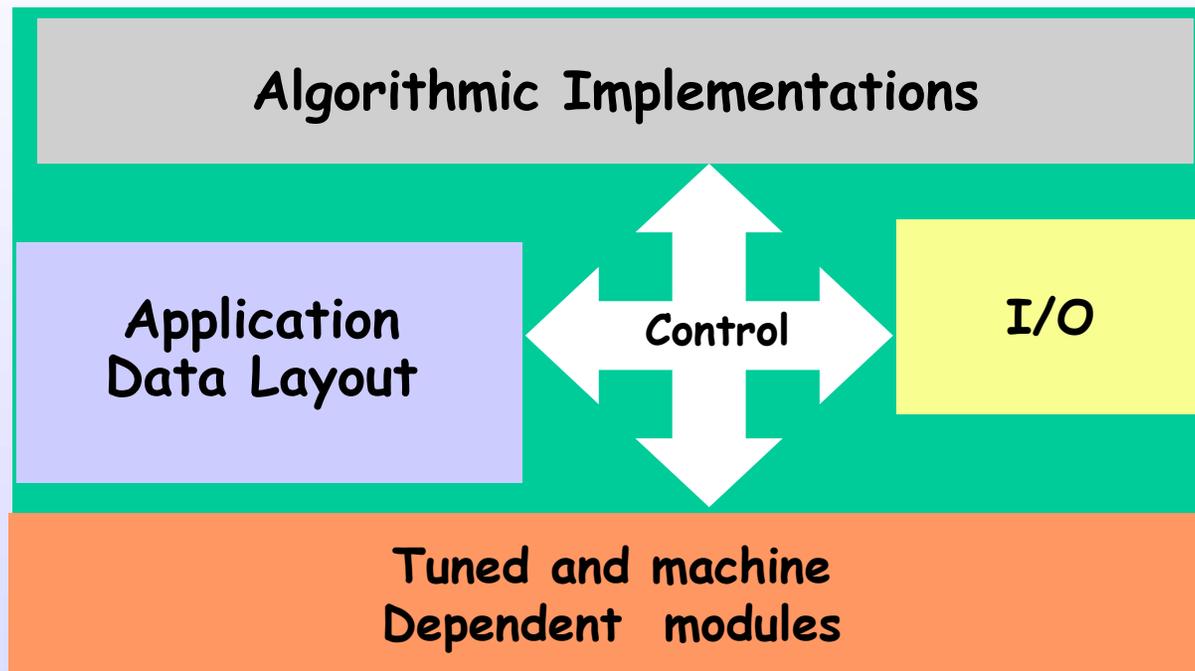


Evolution of High Performance Computers





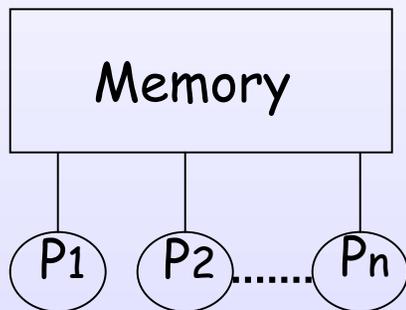
Large Scientific Codes: *A Common Programming Practice*



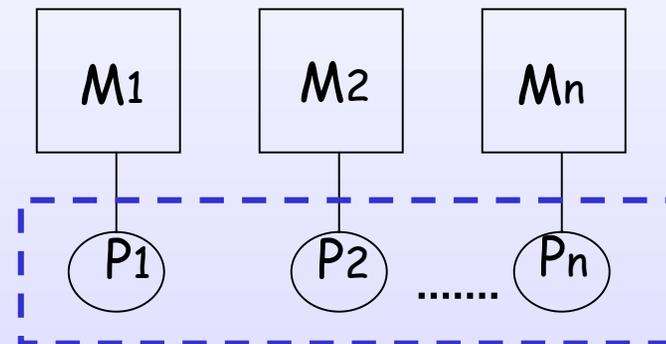
Memory Latency

Hybrid-Model

Shared
Memory



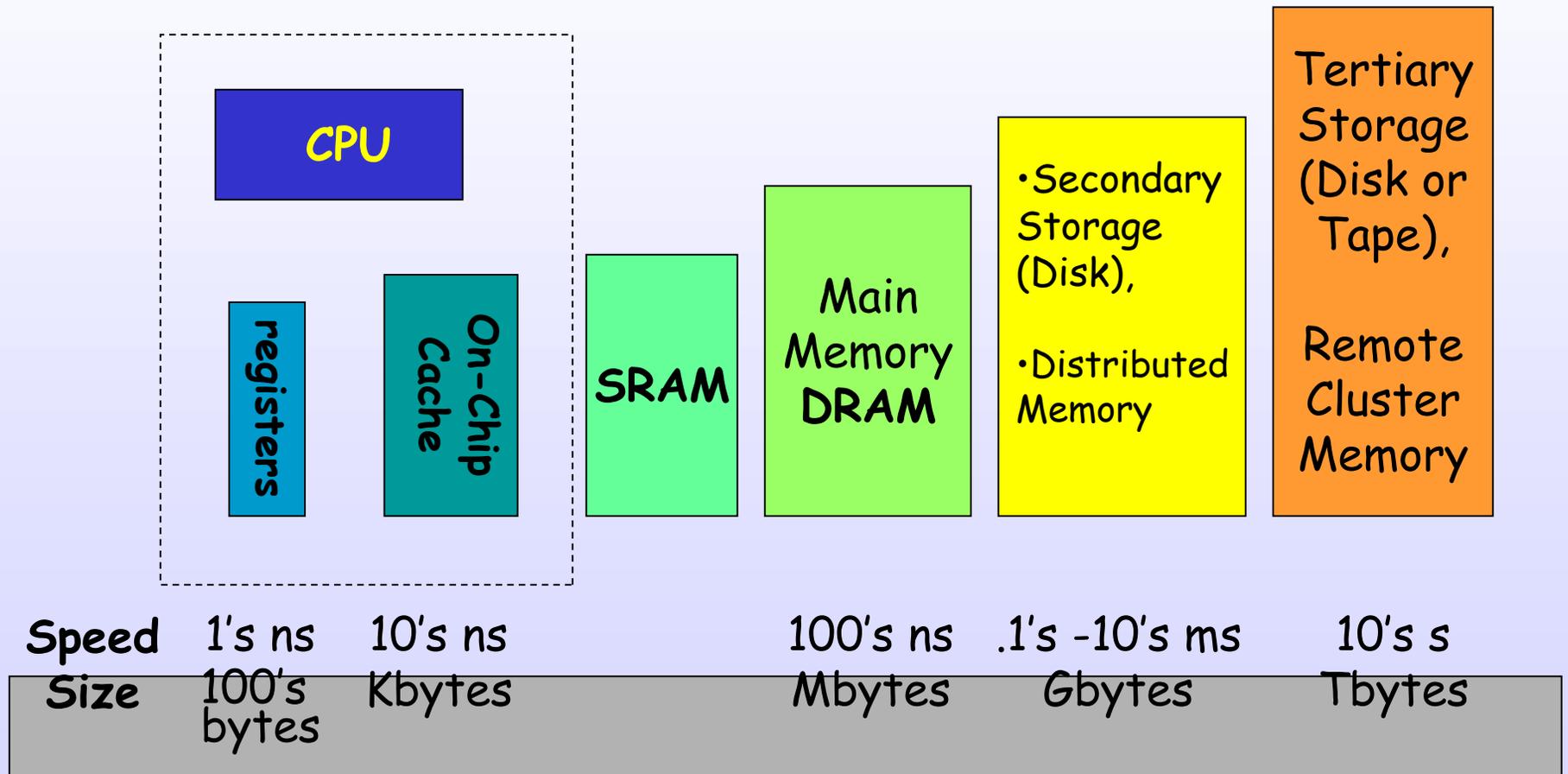
Distributed
Memory



Different interconnection
mechanisms

Memory Hierarchy

- *Where is the data? Why is data locality important?*

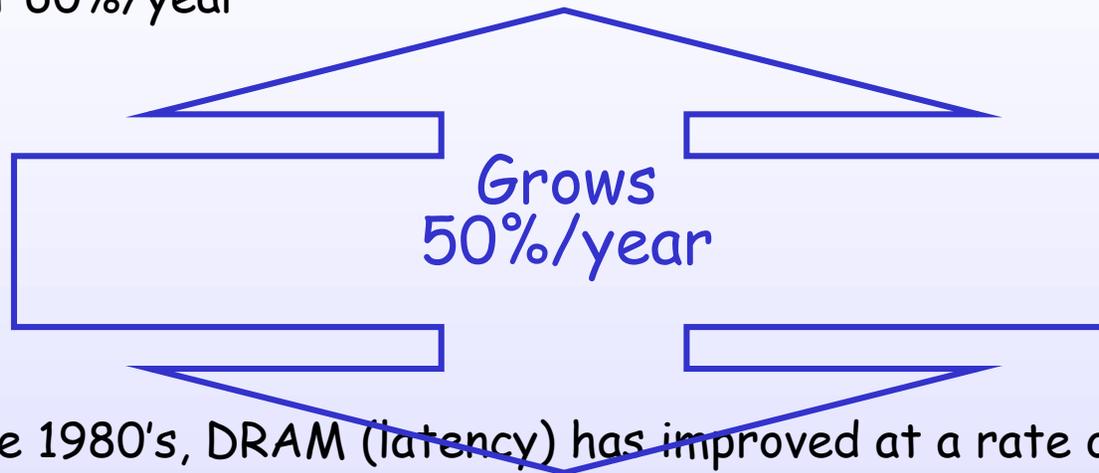




CPU vs. DRAM Performance

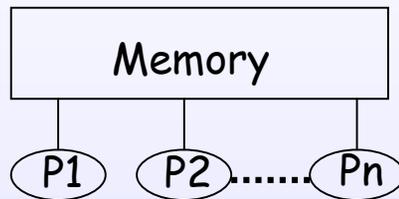


- Since 1980's, μ Procs performance has increased at a rate of almost 60%/year

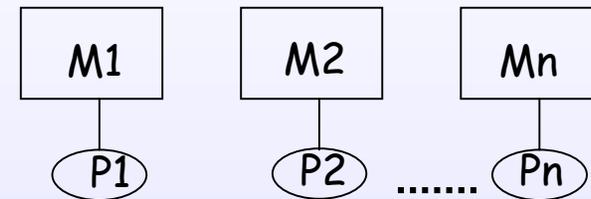


- Since 1980's, DRAM (latency) has improved at a rate of almost 9%/year

Shared Memory

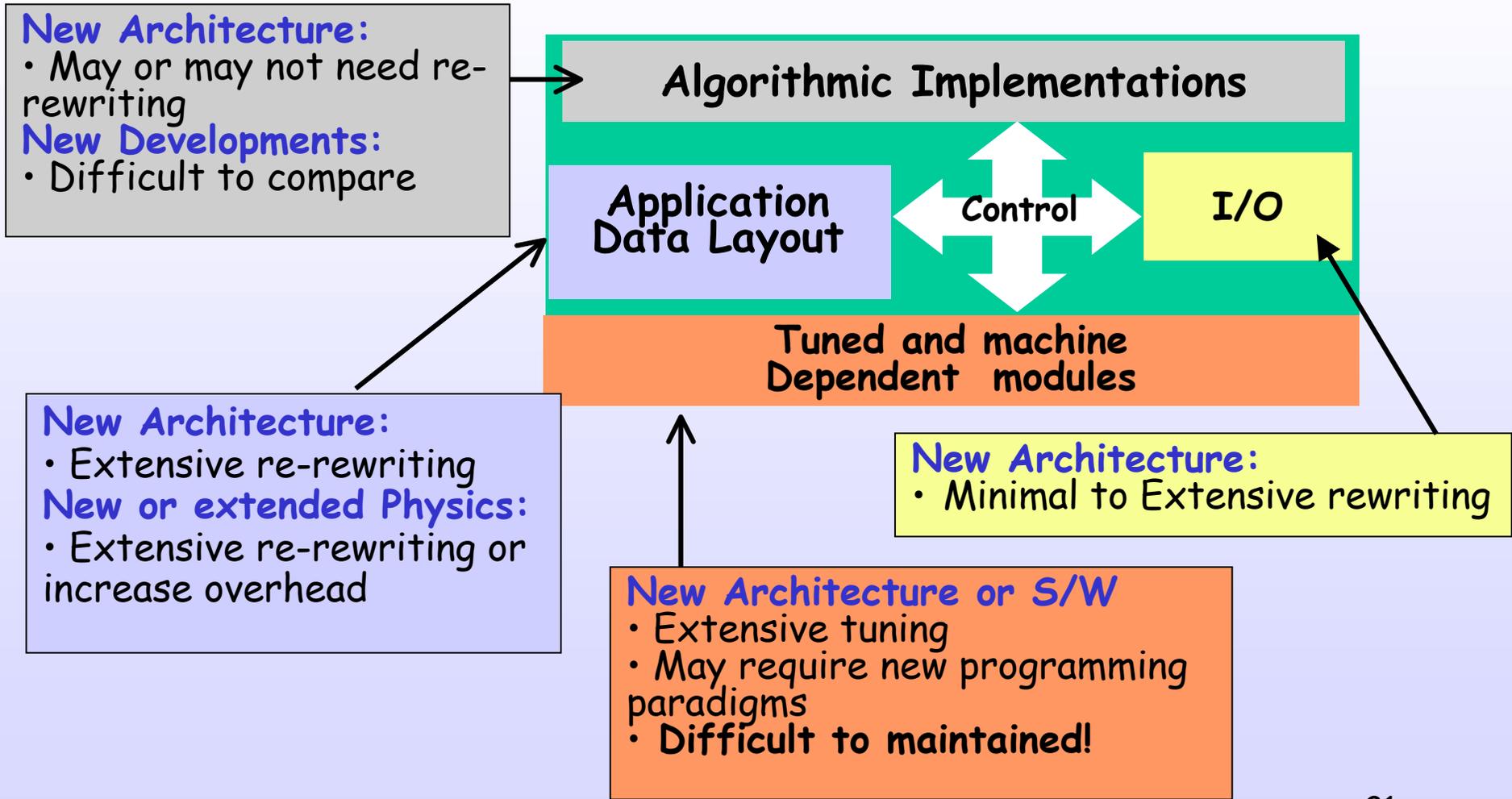


Distributed Memory



- Data parallelism
- easier to implement
- shared memory space
- mutual exclusion, contention
- shared area is use for sending and receiving data
- Message Passing
- virtual shared memory
- data is implicitly available to all
- Implicit mutual exclusion
- Only explicit synch
- Depends on Memory Hierarchy and Network

Shortcomings

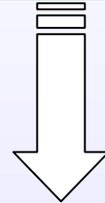
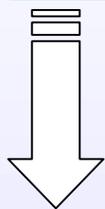




Alternative Programming Approach



USER'S APPLICATION CODE
(Main Control)



AVAILABLE

Application
Data Layout

LIBRARIES & PACKAGES

AVAILABLE

Algorithmic
Implementations

LIBRARIES & PACKAGES

AVAILABLE

I/O

LIBRARIES



**Tuned and machine
Dependent modules**



Software Development Levels of abstraction



- Scientific or engineering context
- Domain expertise

- Simulation codes
- Data Analysis codes

- Templates
- Scientific Computing Tools
- General Purpose Libraries

- Data Structures
- Algorithms
- Code Optimization
- Programming Languages
- O/S - Compilers

Hardware - Middleware - Firmware



What needs to be computed?



J. Demmel (16)

ScaLAPACK

J. Demmel(18)

Aztec/Trilinos

SuperLU

$$Ax = b$$

$$Az = \lambda z$$

$$A = U\Sigma V^T$$

J. Meza(21)

$$\min \left\{ \frac{1}{2} \|r(x)\|^2 : x_l \leq x \leq x_u \right\}$$

OPT++

PDEs

ODEs

TAO

SUNDIALS

PETSc

Hypre

Today

Today



What codes are being developed?



Global Arrays

Overture

Parallel programs that use large distributed arrays

PAWS

Coupling distributed applications

Support for Grids and meshes

Infrastructure for distributed computing

On-line visualization and computational steering

Language Interoperability

Performance analysis and monitoring

Chasm

CUMULVS

Globus

TAU



Portable, Extensible Toolkit for Scientific
Computation
PETSc



The PETSc Development Team

Argonne National Laboratory
Mathematics and Computer Science Division

<http://www-fp.mcs.anl.gov/petsc/>

- Satish Balay
- Kris Buschelman
- Bill Gropp
- Dinesh Kaushik
- Mathew Knepley
- Lois Curfman-McInnes
- Barry Smith
- Hong Zhang

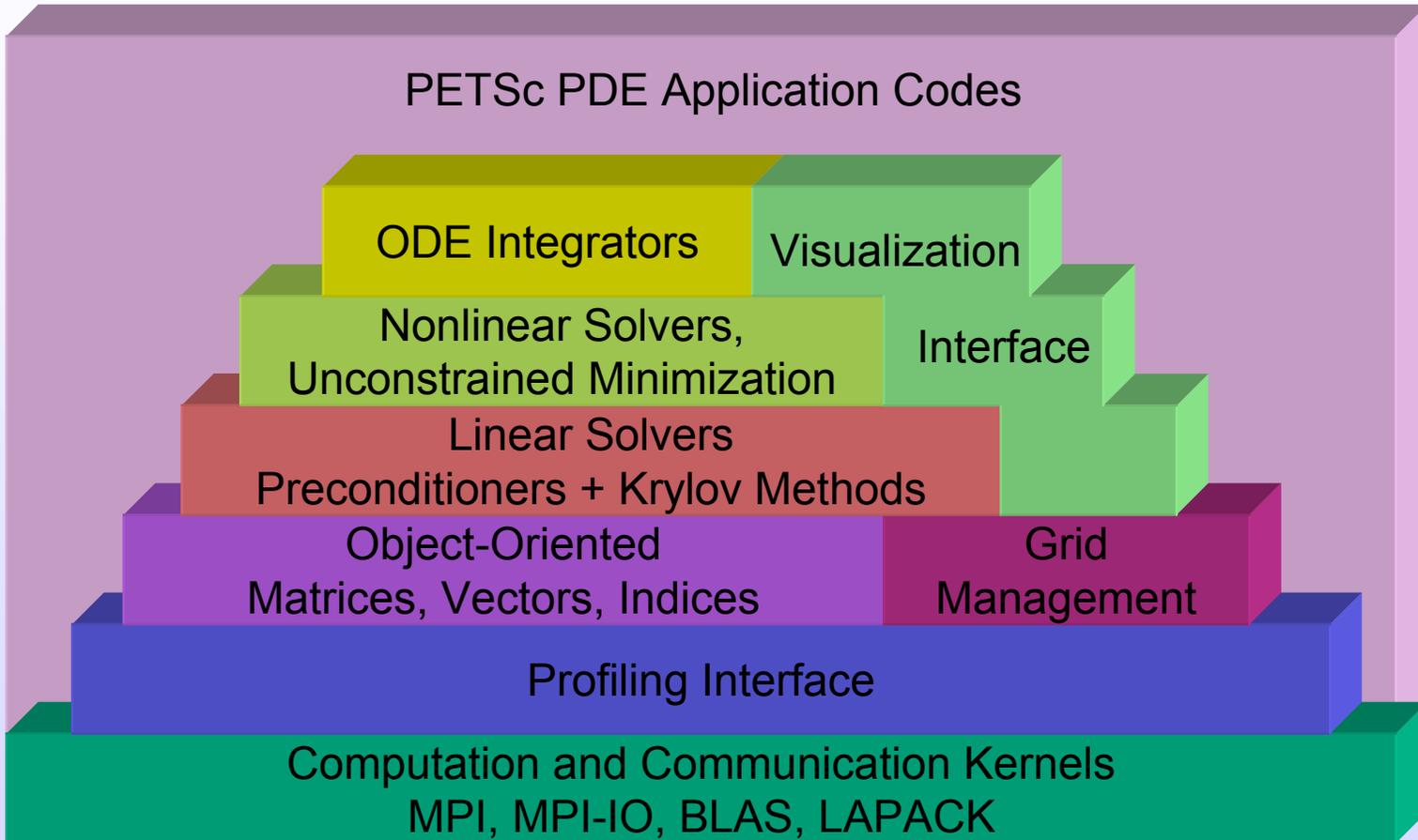


What is PETSc?



- A toolkit that eases the difficulties of developing parallel, non-trivial PDE solvers that deliver high performance (not a PDE solver black-box!)
- Freely available (well documented + lots examples and tutorials!)
- Portable to any parallel system supporting MPI
- Begun in 1991. Over 8,500 downloads. Current version 2.1.3

Structure of PETSc

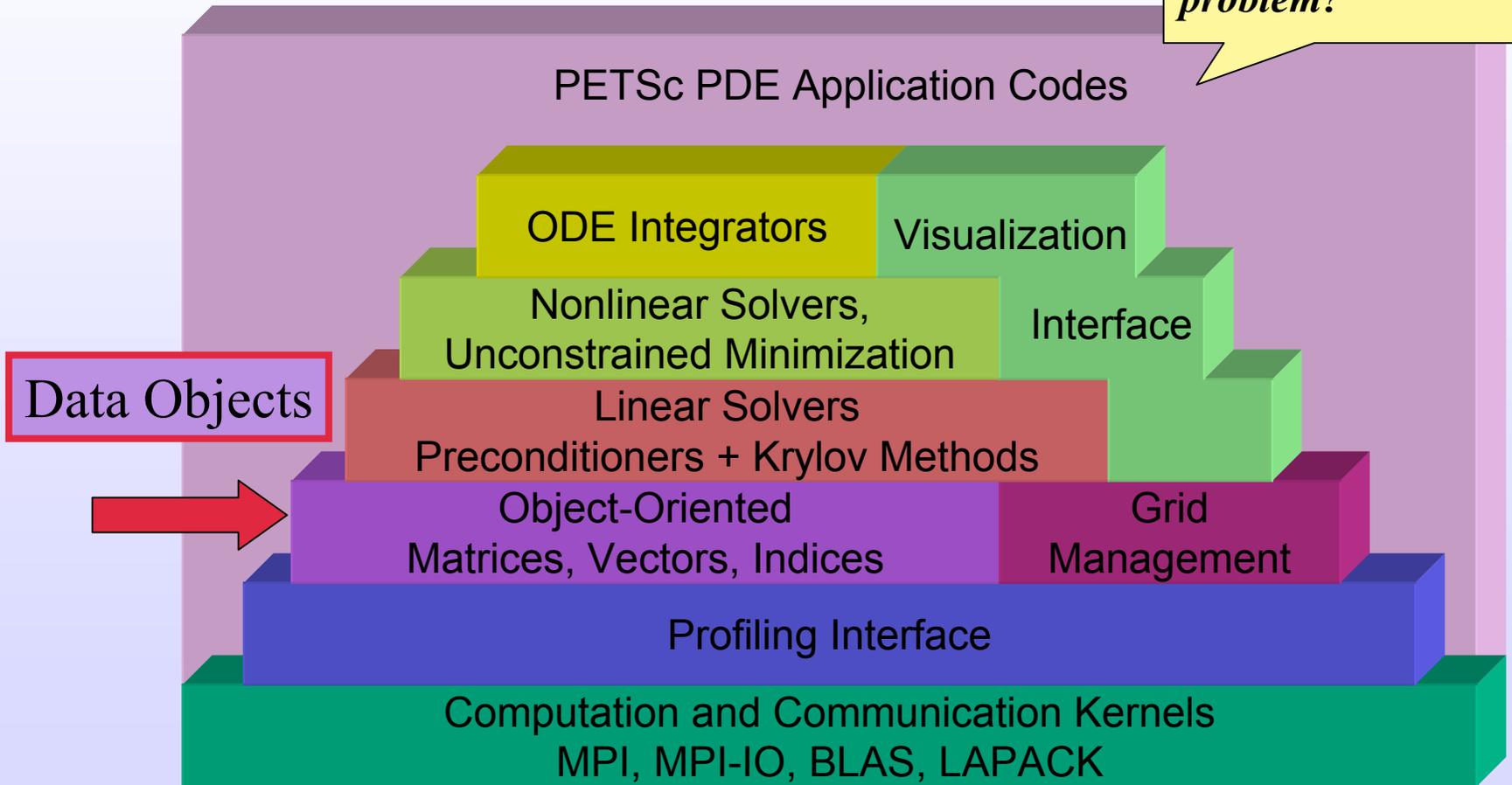




Structure of PETSc



How to specify the mathematics of the problem?

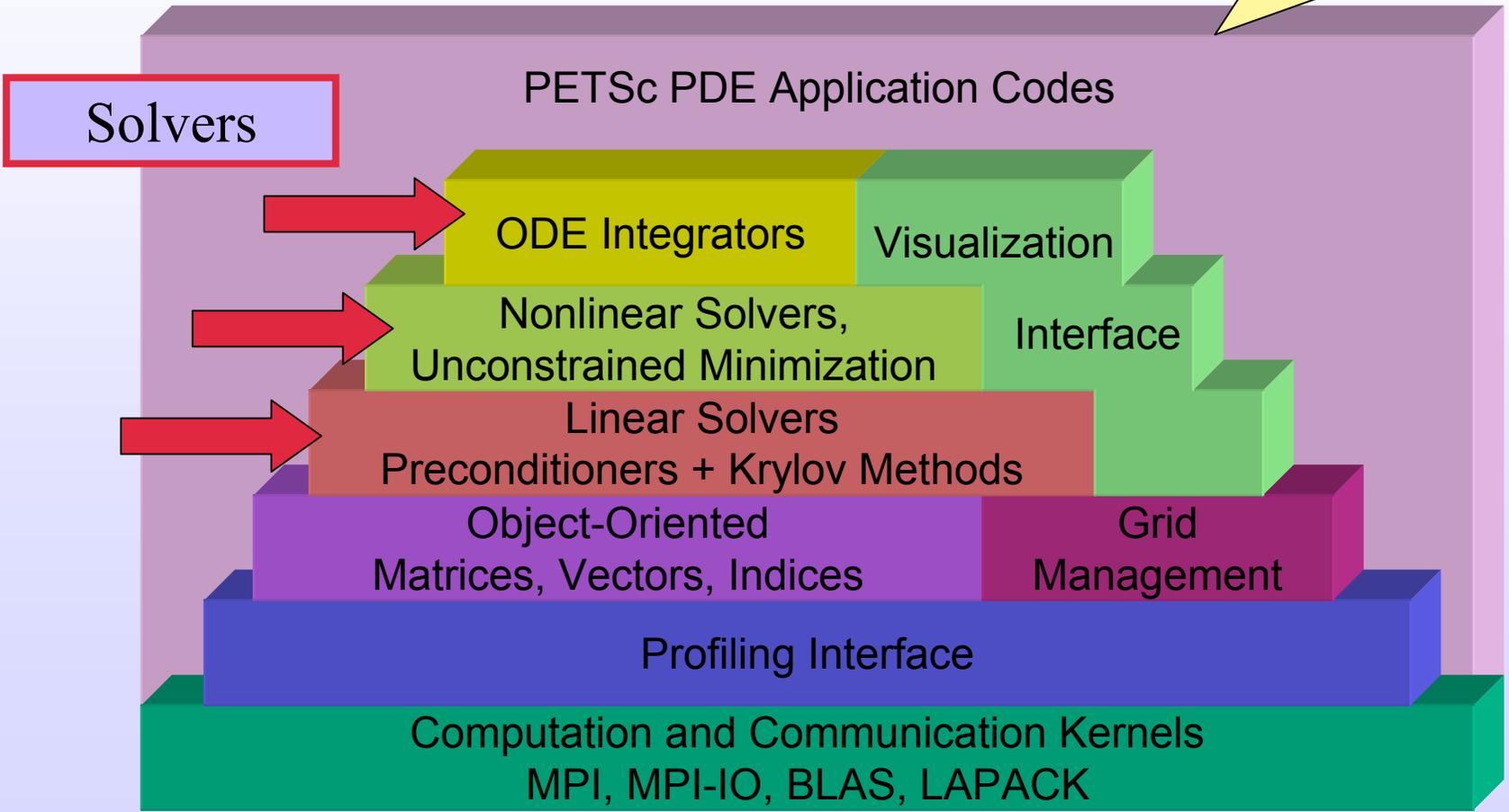




Structure of PETSc



How to solve the problem?



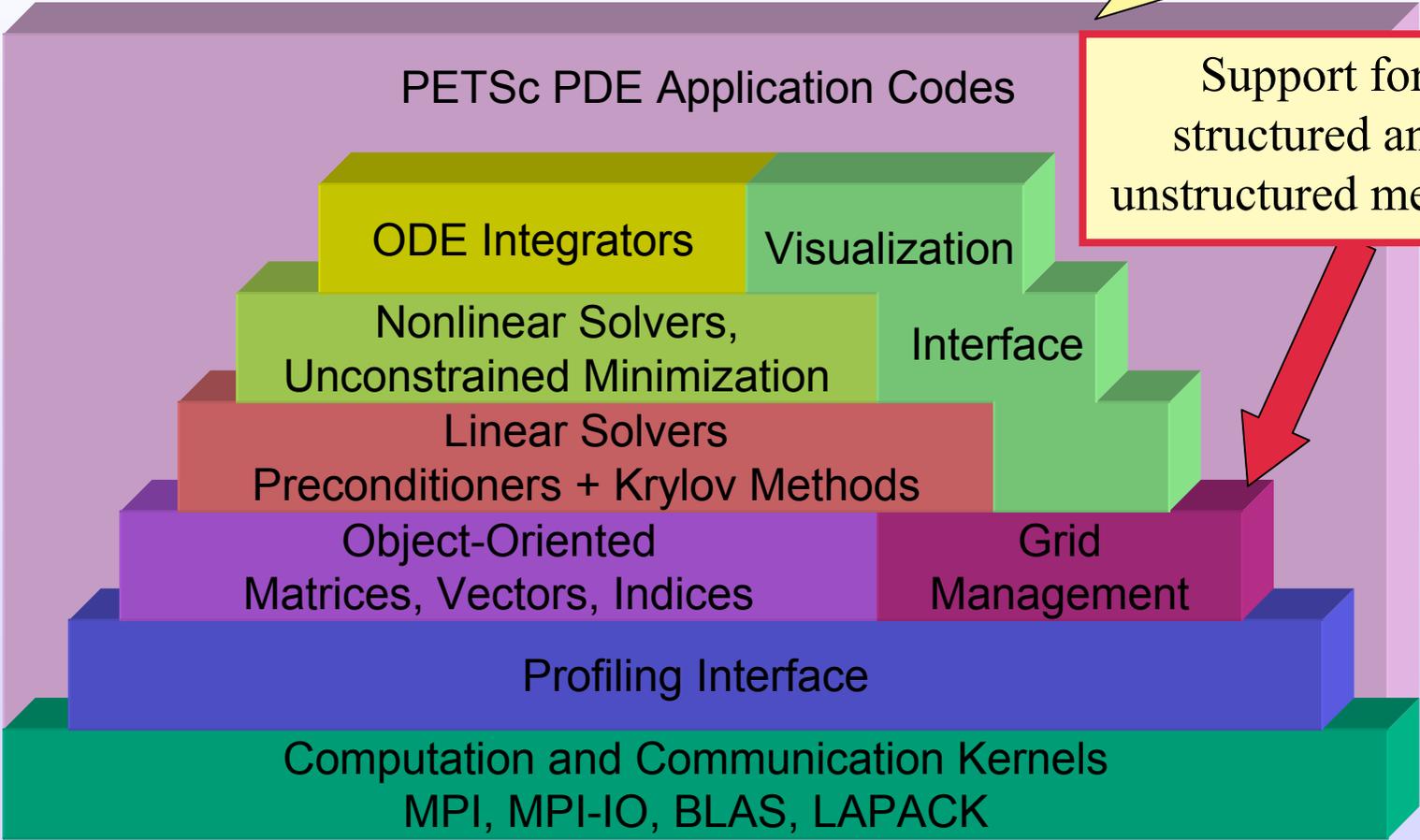


Structure of PETSc



How to handle Parallel computations?

Support for structured and unstructured meshes



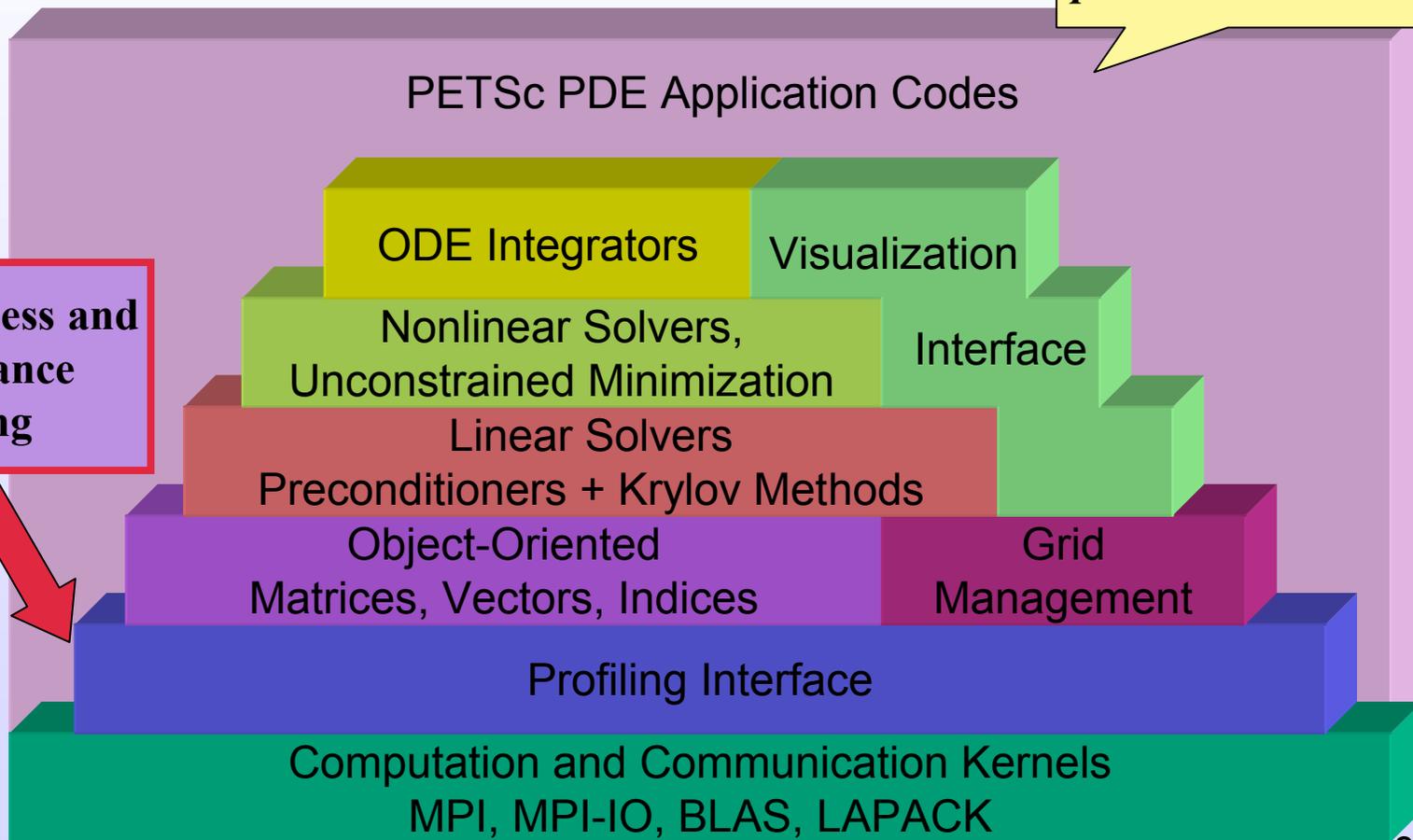


Structure of PETSc



What debugging and monitoring aids it provides?

Correctness and Performance Debugging





PETSc Numerical Components



Nonlinear Solvers		
Newton-based Methods		Other
Line Search	Trust Region	

Time Steppers			
Euler	Backward Euler	Pseudo Time Stepping	Other

Krylov Subspace Methods							
GMRES	CG	CGS	Bi-CG-STAB	TFQMR	Richardson	Chebyshev	Other

Preconditioners						
Additive Schwartz	Block Jacobi	Jacobi	ILU	ICC	LU (Sequential only)	Others

Matrices					
Compressed Sparse Row (AIJ)	Blocked Compressed Sparse Row (BAIJ)	Block Diagonal (BDIAG)	Dense	Matrix-free	Other

Distributed Arrays

Vectors

Index Sets			
Indices	Block Indices	Stride	Other



Vectors (basic operations)



- Each process locally owns a subvector of contiguously numbered global indices
- Types: Sequential, MPI or SHARED

–VecCreate(MPI_Comm Comm, Vec * v)
• **comm** - MPI_Comm of processors that share the vector
• **v** = vector

–VecSetType(Vec, VecType)
• Where VecType is
–VEC_SEQ, VEC_MPI, or VEC_SHARED

–VecSetSizes(Vec *v, int n, int N)
▪ Where **n** or **N** (not both) can be PETSC_DECIDE

– VecDestroy(Vec *)





Selected Vector Operations



Function Name	Operation
VecAXPY(Scalar *a, Vec x, Vec y)	$y = y + a*x$
VecAYPX(Scalar *a, Vec x, Vec y)	$y = x + a*y$
VecWXPY(Scalar *a, Vec x, Vec y, Vec w)	$w = a*x + y$
VecScale(Scalar *a, Vec x)	$x = a*x$
VecCopy(Vec x, Vec y)	$y = x$
VecPointwiseMult(Vec x, Vec y, Vec w)	$w_i = x_i * y_i$
VecMax(Vec x, int *idx, double *r)	$r = \max x_i$
VecShift(Scalar *s, Vec x)	$x_i = s + x_i$
VecAbs(Vec x)	$x_i = x_i $
VecNorm(Vec x, NormType type, double *r)	$r = x $



Matrices (basic operations)



- Fundamental objects for storing linear operators (e.g., Jacobians)
- Types:
 - default sparse AIJ: MPIAIJ, SEQAIJ
 - block sparse AIJ (for multi-component PDEs): MPIAIJ, SEQAIJ
 - symmetric block sparse AIJ: MPISBAIJ, SAEQSBAIJ
 - block diagonal: MPIBDIAG, SEQBDIAG
 - dense: MPIDENSE, SEQDENSE
 - matrix-free

–MatCreate(MPI_Comm Comm, m,n,M,N, Mat * Mat)

- MPI_Comm - processors that share the matrix
- number of local (m x n)/global (M x N) rows and columns

–MatSetType(Mat,MatType)
–MatDestroy(Mat)



Matrices (basic ops)



- Single user interface, e.g.,
 - Matrix assembly
 - MatSetValues()
 - Matrix-vector multiplication
 - MatMult()
 - Matrix viewing
 - MatView()
- Multiple underlying implementations
 - AIJ, block AIJ, symmetric block AIJ, block diagonal, dense, matrix-free, etc.



PETSc Linear Solvers



Goal: Support the solution of linear systems,

$$Ax=b,$$

particularly for sparse, parallel problems arising within PDE-based models

User provides:

- Code to evaluate A, b



PETSc Linear Solvers (Preconditioning)



Given the linear system of equations:

$$Ax=b \quad (1)$$

Krylov Projection Methods (KSP) are strongly dependent on the spectrum of A . Use of preconditioning techniques usually accelerate the convergence rate of the iterative techniques.

$$(M_L^{-1}AM_R^{-1})(M_Rx) = M_L^{-1}b,$$

For $M_L=I$

$$r \equiv b - AM_R^{-1}M_Rx$$

For $M_R=I$

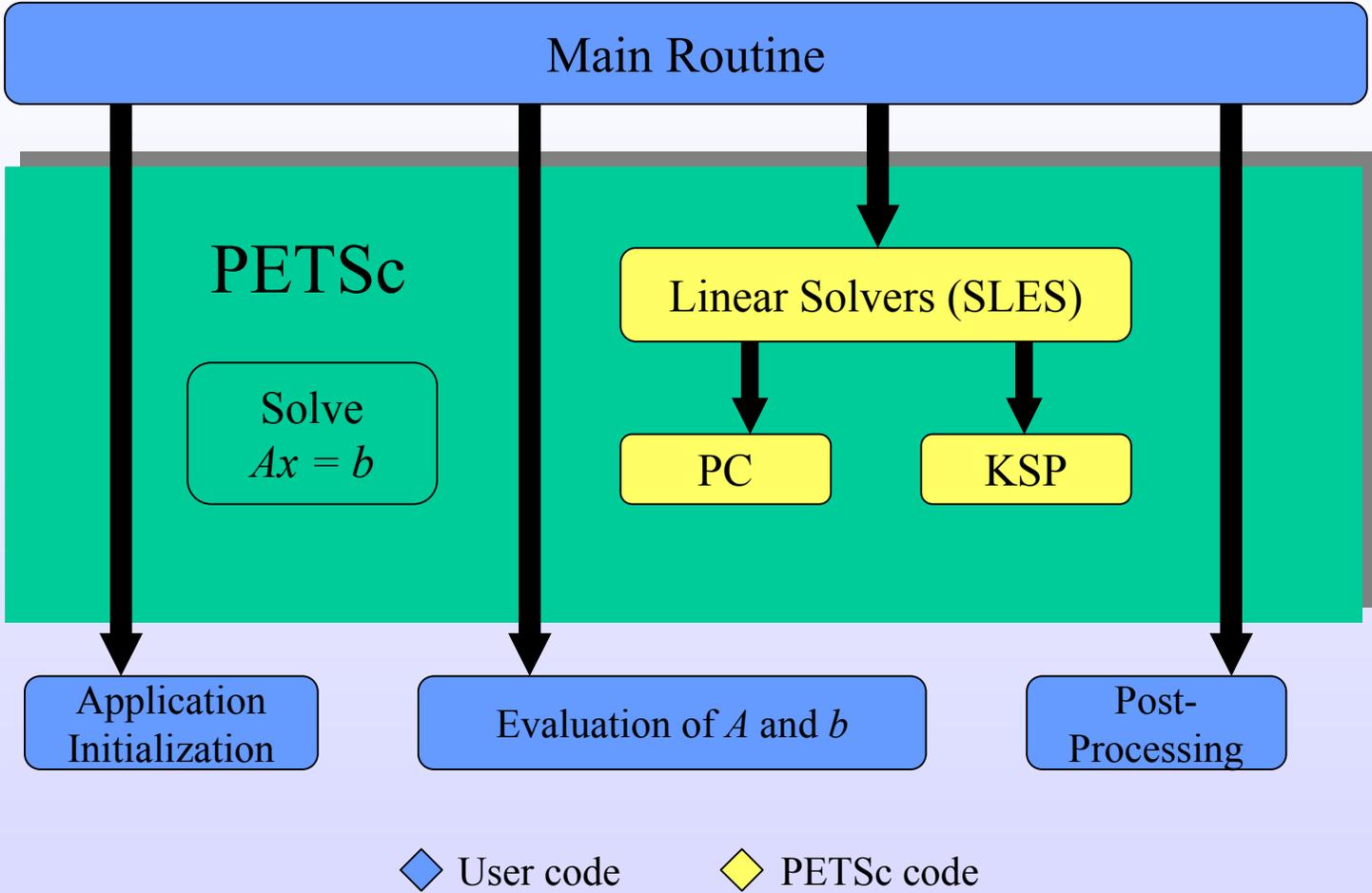
$$r_L \equiv M_L^{-1}b - M_L^{-1}Ax = M_L^{-1}r$$

**PETSC
Default**

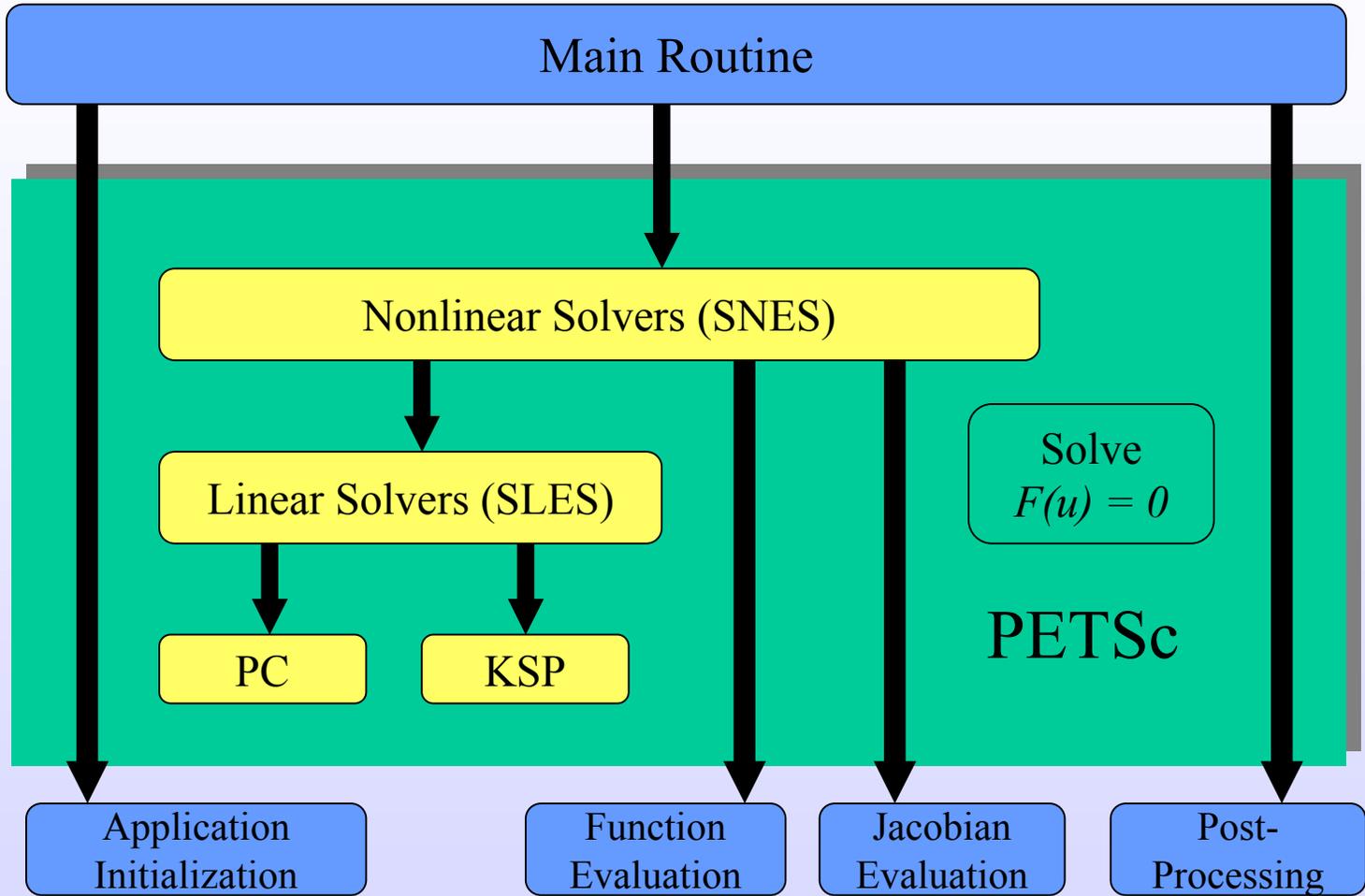
KRYLOV SUBSPACE METHODS + PRECONDITIONERS

R. Freund, G. H. Golub, and N. Nachtigal. *Iterative Solution of Linear Systems*, pp 57-100. *ACTA Numerica*. Cambridge University Press, 1992.

PETSc Linear Solvers (SLES)



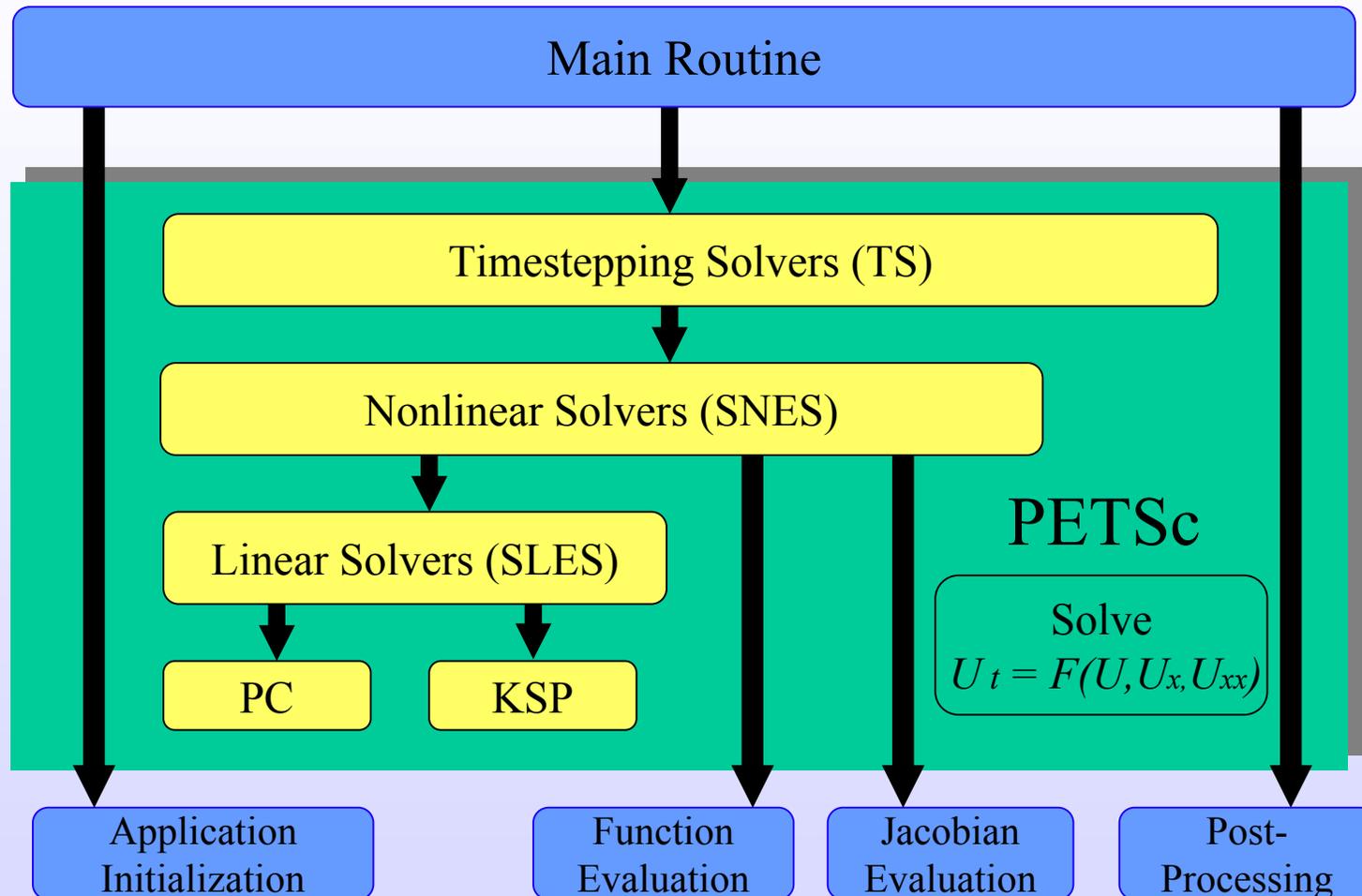
PETSc Non-Linear Solvers (SNLES)



◆ User code

◆ PETSc code

Time Dependent PDE Solution

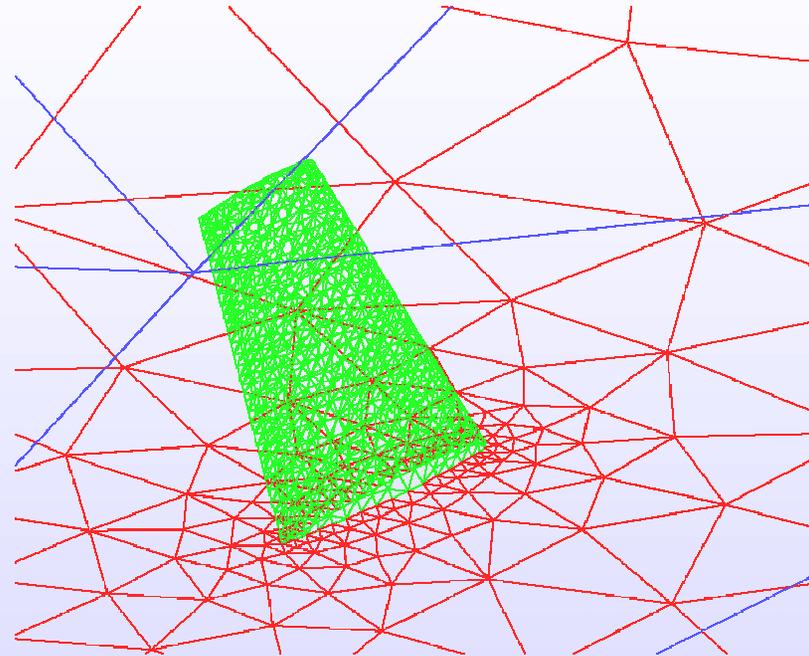




CFD Application using PETSc



- 3D incompressible Euler
- Tetrahedral grid
- Up to 11 million unknowns
- Based on a legacy NASA code, FUN3d, developed by W. K. Anderson
- Fully implicit steady-state
- Primary PETSc tools: nonlinear solvers (SNES) and vector scatters (VecScatter)



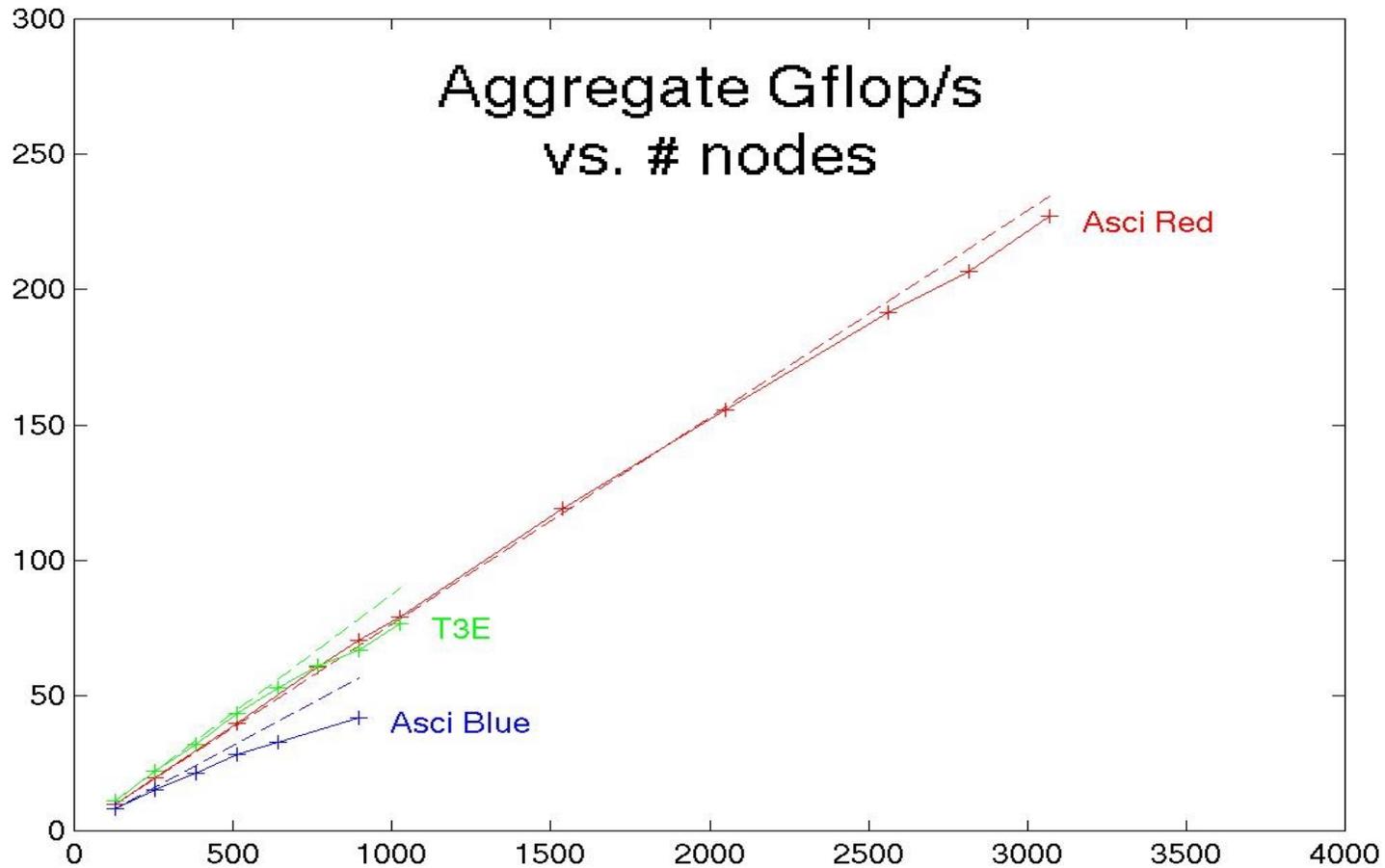
Results courtesy of Dinesh Kaushik and David Keyes, Old Dominion Univ., partially funded by NSF and ASCI level 2 grant



CFD Application using PETSc



Dimension=11,047,096



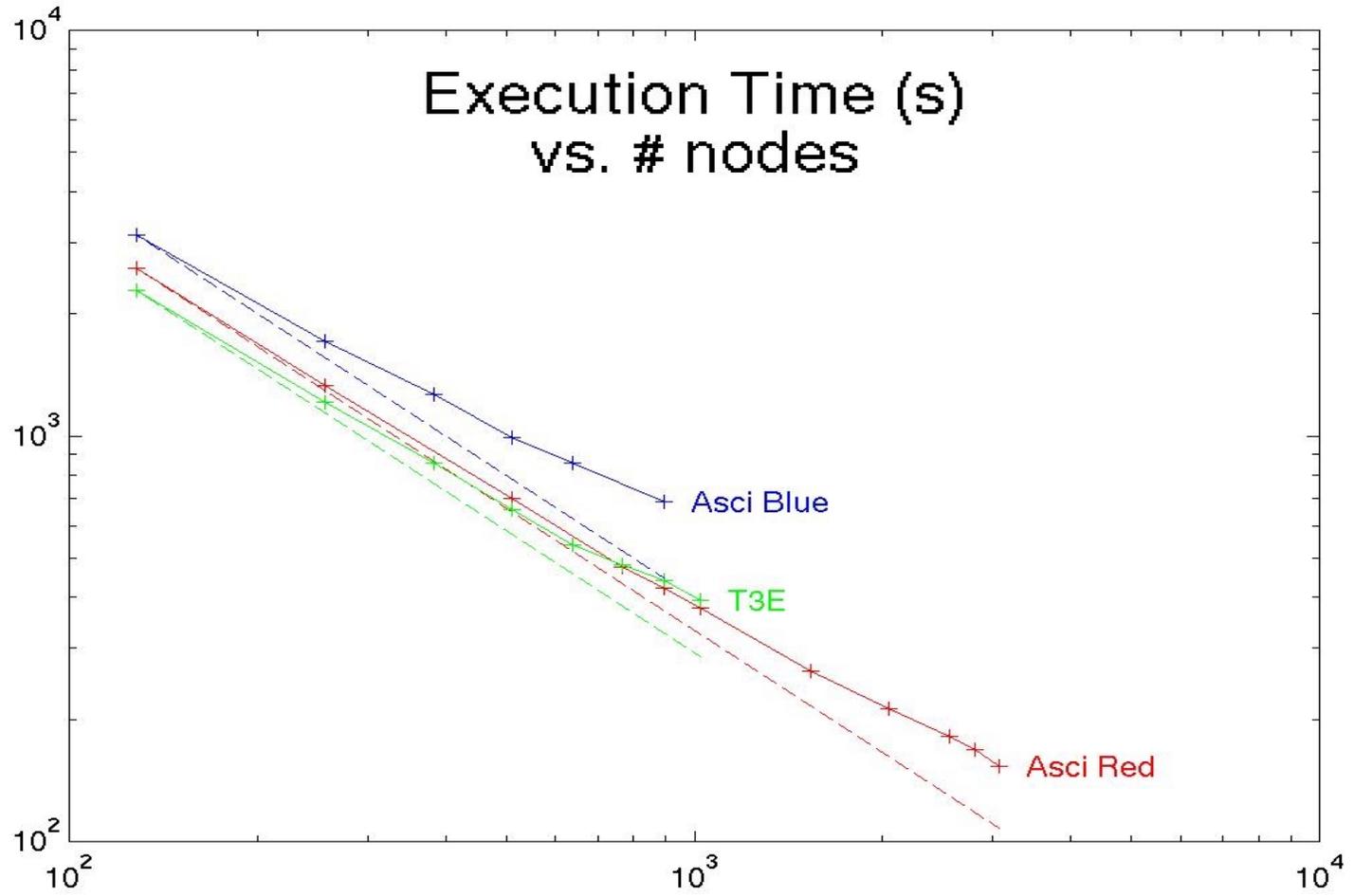
Fixed-size Parallel Scaling Results (GFlop/s)



CFD Application using PETSc



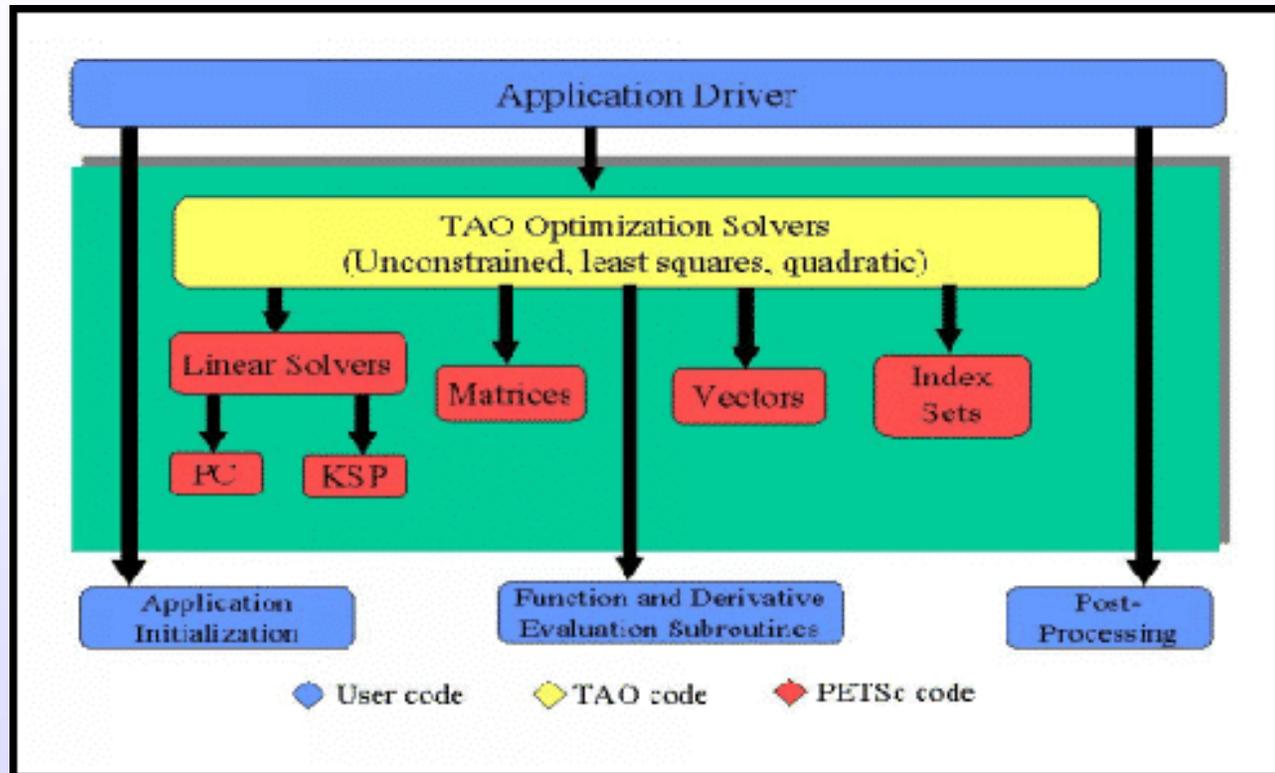
(Time in seconds)



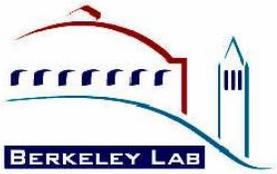
Fixed-size Parallel Scaling Results

TAO

Toolkit for Advanced Optimization



S. Benson, L. Curfman-McInnes, J. More and J. Sarich
Argonne National Laboratory



hypr

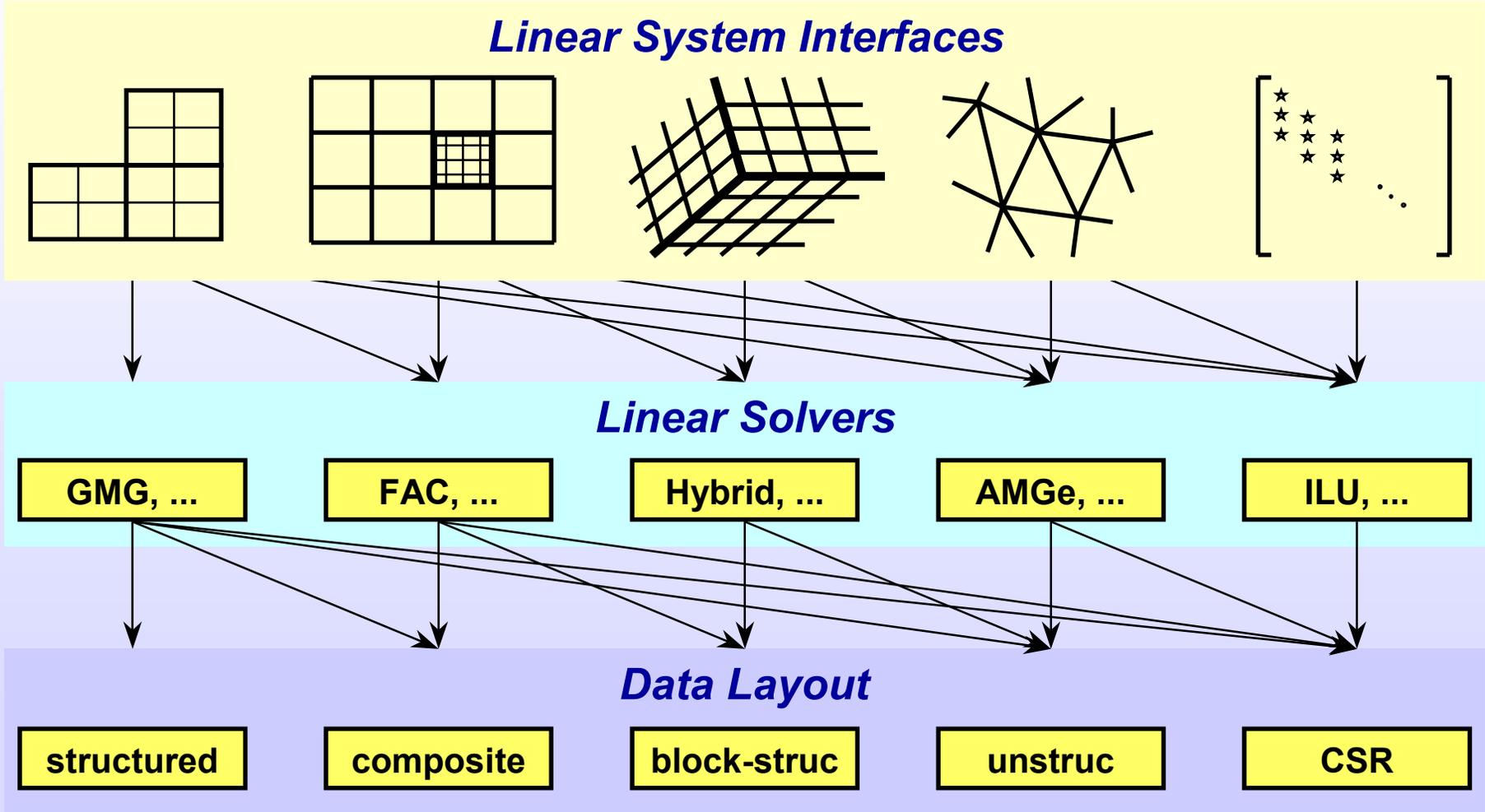


Hypre TEAM

Lawrence Livermore National Laboratory

- Rob Falgout (project leader)
- Guillermo Castilla
- Edmond Chow
- Andy Cleary
- Van Emden Henson
- Jim Jones
- Mike Lambert
- Barry Lee
- Jeff Painter
- Charles Tong
- Tom Treadway
- Panayot Vassilevski
- Ulrike Meier Yang

Hypre's Multiple interfaces





hypr conceptual interfaces



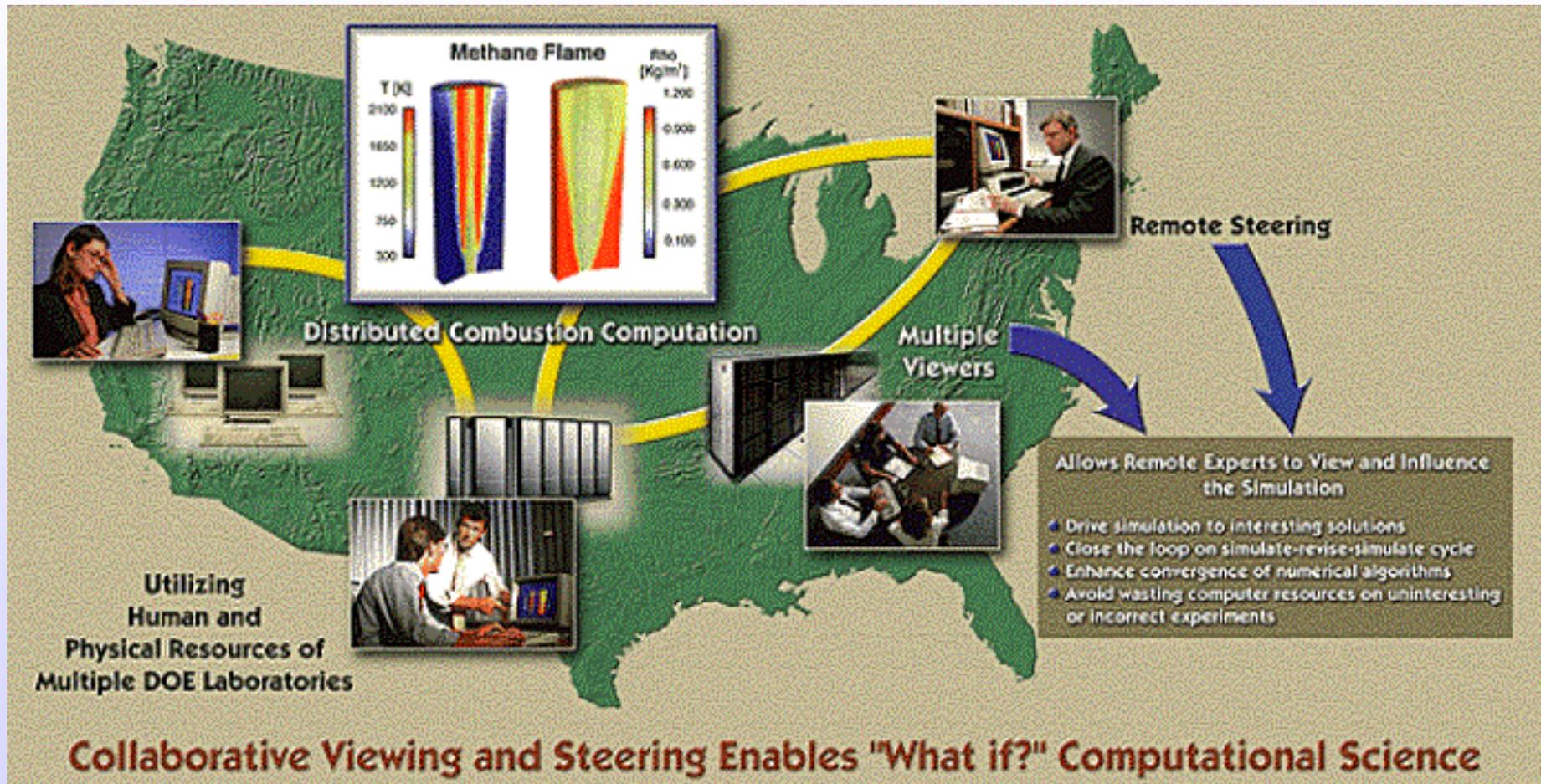
- Structured-Grid Interface (**Struct**)
 - *applications with logically rectangular grids*
- Semi-Structured-Grid Interface (**SStruct**)
 - *applications with grids that are mostly—but not entirely—structured (e.g., block-structured, structured AMR, overset)*
- Finite Element Interface (**FEI**)
 - *unstructured-grid, finite element applications*
- Linear-Algebraic Interface (**IJ**)
 - *applications with sparse linear systems*



CUMULVS



Development Team: Oak Ridge National Laboratory
Lead PIs: J. Kohl and Philip Papadopoulos

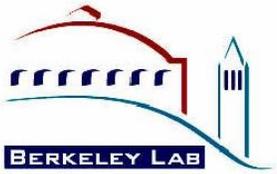




CUMULVS



- Collaborative User Migration, User Library for Visualization and Steering
- Enables parallel programming with the integration of:
 - Interactive visualization (local and remote)
 - Multiple views
 - Fault Tolerance
 - Computational Steering



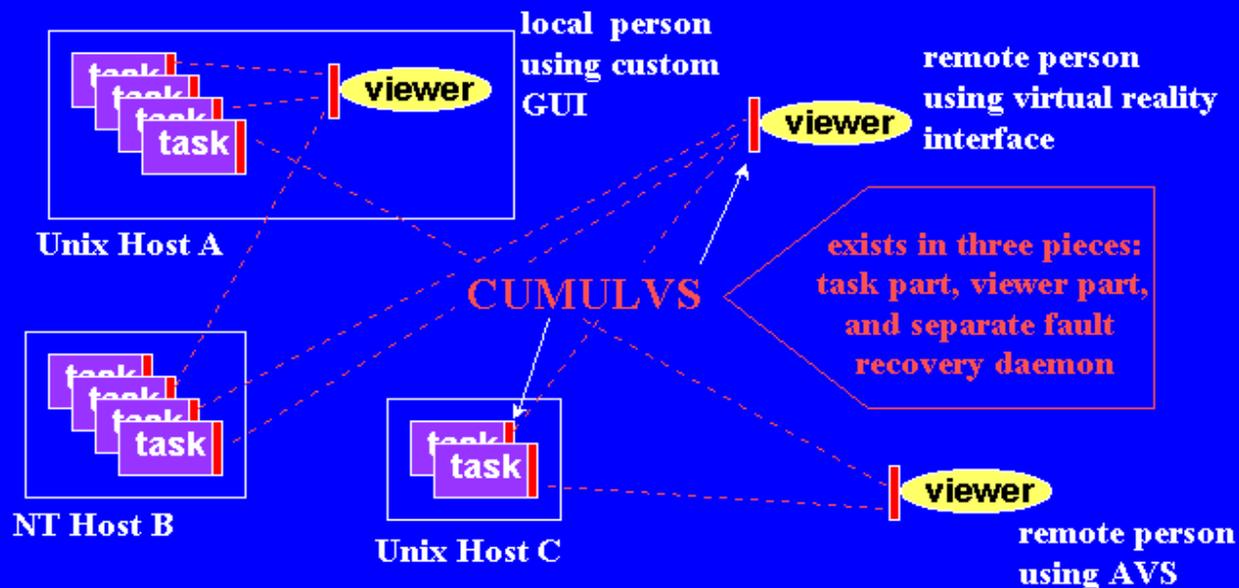
CUMULVS



- Setup input parameters to be steered
- Specify the nature and decomposition of the data fields to be visualized. Standard data decompositions: Block, Block-Cyclic, particle decompositions and User defined decompositions
- Use existing interfacing to visualization packages or define a custom viewer on top of other visualization tools.
- Setup checkpoint/restart mechanism

CUMULVS

coordinates the consistent collection and dissemination of information to/from parallel tasks to multiple viewers



distributed parallel application or simulation
supports most target platforms (PVM/MPI, Unix/NT, etc.)

Kohl-1999/13



The GRID



- A large pool of resources
 - Computers
 - Networks
 - Software
 - Databases
 - Instruments
 - people

Requirements from GRID implementation:

- Ubiquitous: ability to interface to the grid at any point and leverage whatever is available
- Resource Aware: manage heterogeneity of resources
- Adaptive: tailored to obtain maximum performance from resources

Globus



The diagram illustrates the architecture of the Globus project for online instruments. It features a central hub labeled "wide-area dissemination" and "archival storage" connected to three main components:

- Advanced Photon Source:** Represented by an aerial view of the facility, with the text "real-time collection" below it.
- tomographic reconstruction:** Represented by a server rack, with the text "tomographic reconstruction" below it.
- desktop & VR clients with shared controls:** Represented by a computer monitor displaying a 3D reconstruction, with the text "desktop & VR clients with shared controls" above it.

At the top left, the logo for "the globus project" is shown with the URL www.globus.org. The title "Online Instruments" is prominently displayed in orange. At the bottom, the text reads "DOE X-ray source grand challenge: ANL, USC/ISI, NIST, U.Chicago" and "User Tutorial (1.1-3) Introduction 6".



Globus



- High throughput computing
 - Schedule many tasks
 - Bag of resources for the Grid, offering:
 - Resource discovery
 - Data Access
 - Scheduling
 - Reservation
 - Security
 - Accounting
 - Code management



TAU



- Profiling of Fortran 90, C, C++, HPF, and HPC++ codes
- Detailed information (much more than *prof/gprof*)
- C++: per-class and per-instance profiling
- Graphical display of profiling results (built-in viewers, interface to Vampir)



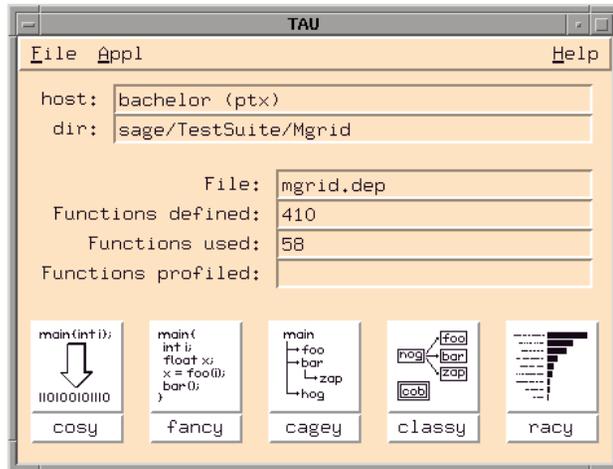
TAU



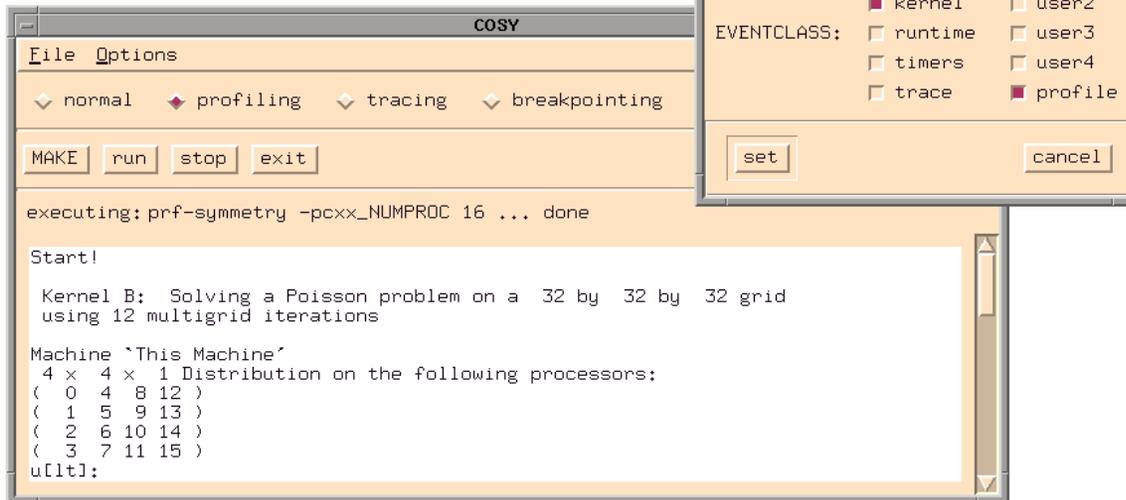
- Profiling of Fortran 90, C, C++, HPF, and HPC++ codes
- Detailed information (much more than *prof/gprof*)
- C++: per-class and per-instance profiling
- Graphical display of profiling results (built-in viewers, interface to Vampir)



TAU



- COSY: Compile manager Status display
- FANCY: File ANd Class display
- CAGEY: CALL Graph Extended display
- CLASSY: CLASS hierarchy browser
- RACY: Routine and data ACcess profile display
- SPEEDY: Speedup and Parallel Execution Extrapolation Display





TAU (Speedy)



The screenshot displays the SPEEDY software interface, which includes a main window and several sub-windows.

SPEEDY Main Window:

- Menu: File
- Buttons: Compile, View/Set Parameters, Run Experiment
- Varying parameter 1: Number of Processors (values: 1 2 4 8 16)
- Varying parameter 2: Latency (values: 0 200 400 600 800 1000)
- Graph: Execution Time [s] vs. Number of Processors. The graph shows execution time increasing with the number of processors for all latency values. Higher latency values result in higher execution times.

XtraP Parameter Viewer:

- General: Processor
- MipsRatio: 1.0
- ProcessMsgType: PredefinedAndPeriodicPolling
- Polling Period: 500.0 [us]
- Buttons: -100, -10, -1, +1, +10, +100
- Buttons: Reset, Load, Save, Close

Speedup Window:

- Graph: Speedup vs. Number of Processors. The graph shows speedup decreasing as the number of processors increases for all latency values. Higher latency values result in lower speedup.

Set Parameter 2: Latency:

- multuples of: 200 from 0 to 1000 include
- powers of: from to
- random sequence:

Set Parameter 1: Number of Processors:

- multuples of: from to include
- powers of: 2 from 1 to 16
- random sequence:



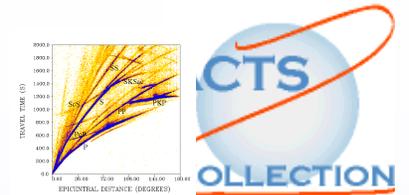
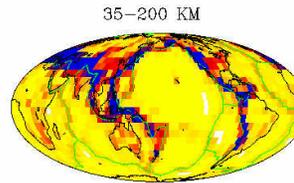
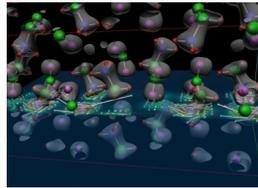
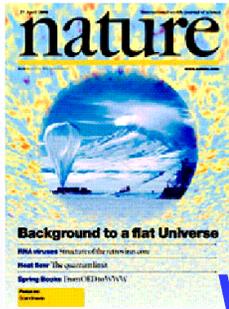
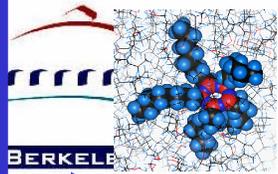
How much effort is involved in
using these tools?



Using the ACTS Collection

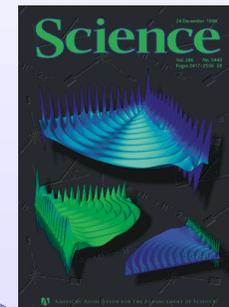
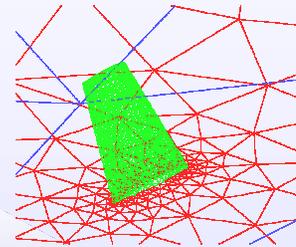
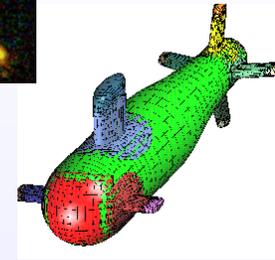
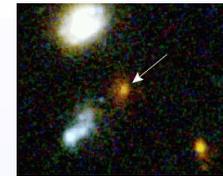


- Most of the tools provide interfaces (calling functions and subroutines) from Fortran and C
- Best approach is to start with examples for beginners!
- Several efforts are targeting Tool Interoperability!

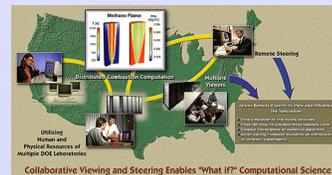
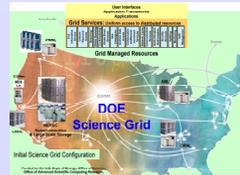
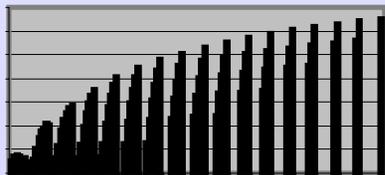


Workshop

ACTS COLLECTION 2002



- CCA
- ScaLAPACK
- Trilinos
- SuperLU
- OPT++
- TAO
- PETSc
- Hypre
- PhiPAC
- SUNDIALS
- Overture
- Chombo
- ATLAS
- TAU
- Globus
- CUMULVS
- Global Arrays
- PAWS



Robust and High Performing Tools for Scientific Computing
Lawrence Berkeley National Laboratory - September 4-7, 2002



To download electronic versions of the material covered in this workshop, please visit:

<http://acts.nersc.gov/events/Workshop2002/program.html>

The material includes pdf files with tutorials on the ACTS tools, example codes and uses. It also contains many references to applications that have benefited from the use of the ACTS Collection.

acts-support@neresc.gov

http://acts.neresc.gov