An Introduction to the

Portable, Extensible Toolkit for Scientific Computation
PETSc

http://acts.nersc.gov

ACTS Toolkit

acts-support@nersc.gov

Tony Drummond
Lawrence Berkley National Laboratory

NPACI - All Hands-Meeting 2002, San Diego, CA

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

** The material used in the preparation of this tutorial comes from the PETSc User's Guide, on-line man pages, tutorials and examples prepared by the PETSc development team*

ACTS Toolkit

An Introduction to the

Portable, Extensible Toolkit for Scientific Computation

PETSc

OUTLINE

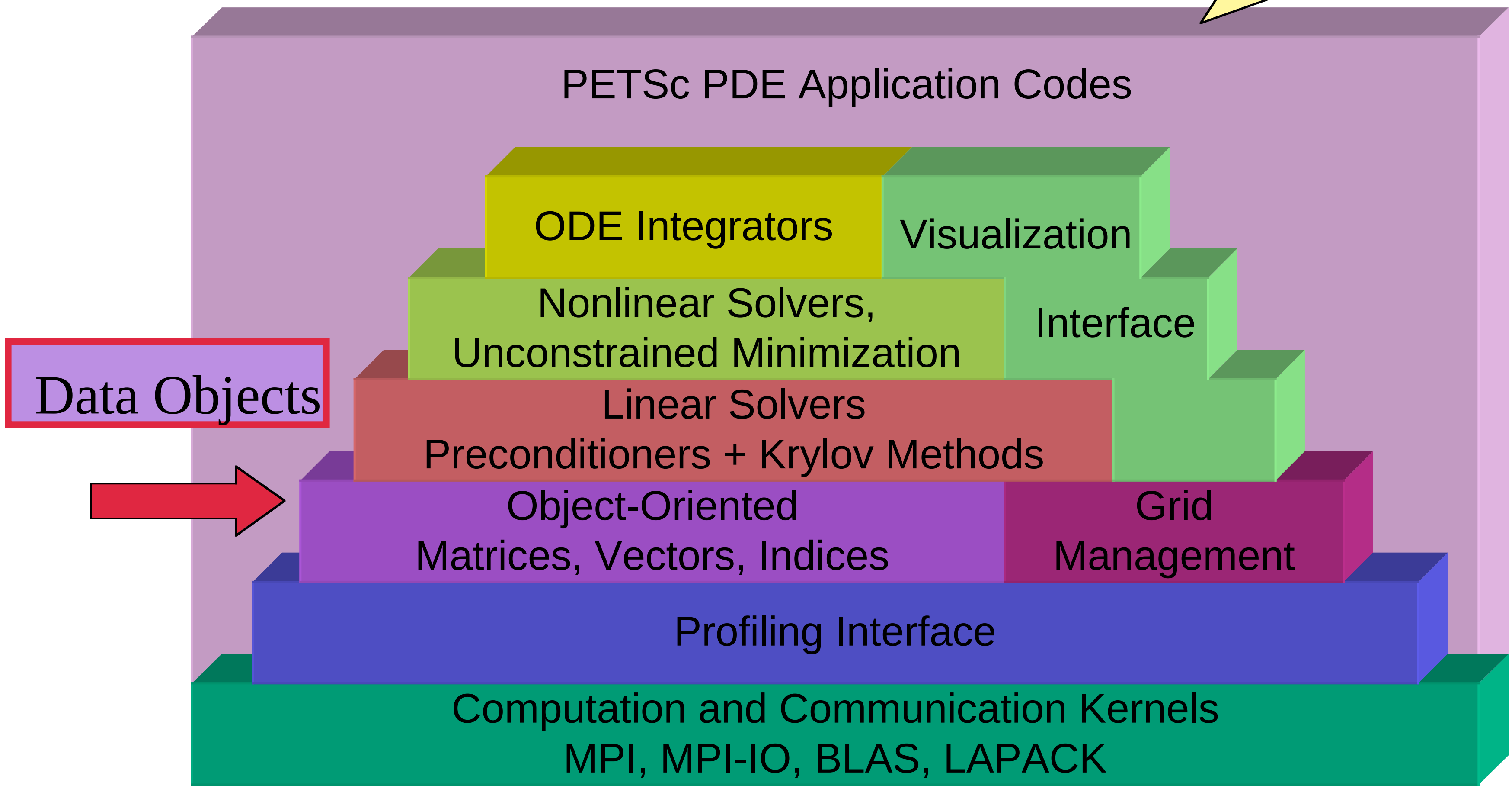
- Basic Concepts
- Vectors, Matrices, Viewers + examples
- Linear Solvers + examples
- Non-Linear Solvers + examples
- Timesteping Solvers
- Structured and Unstructured Meshes
- Applications
- Other Packages

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 2.1.1

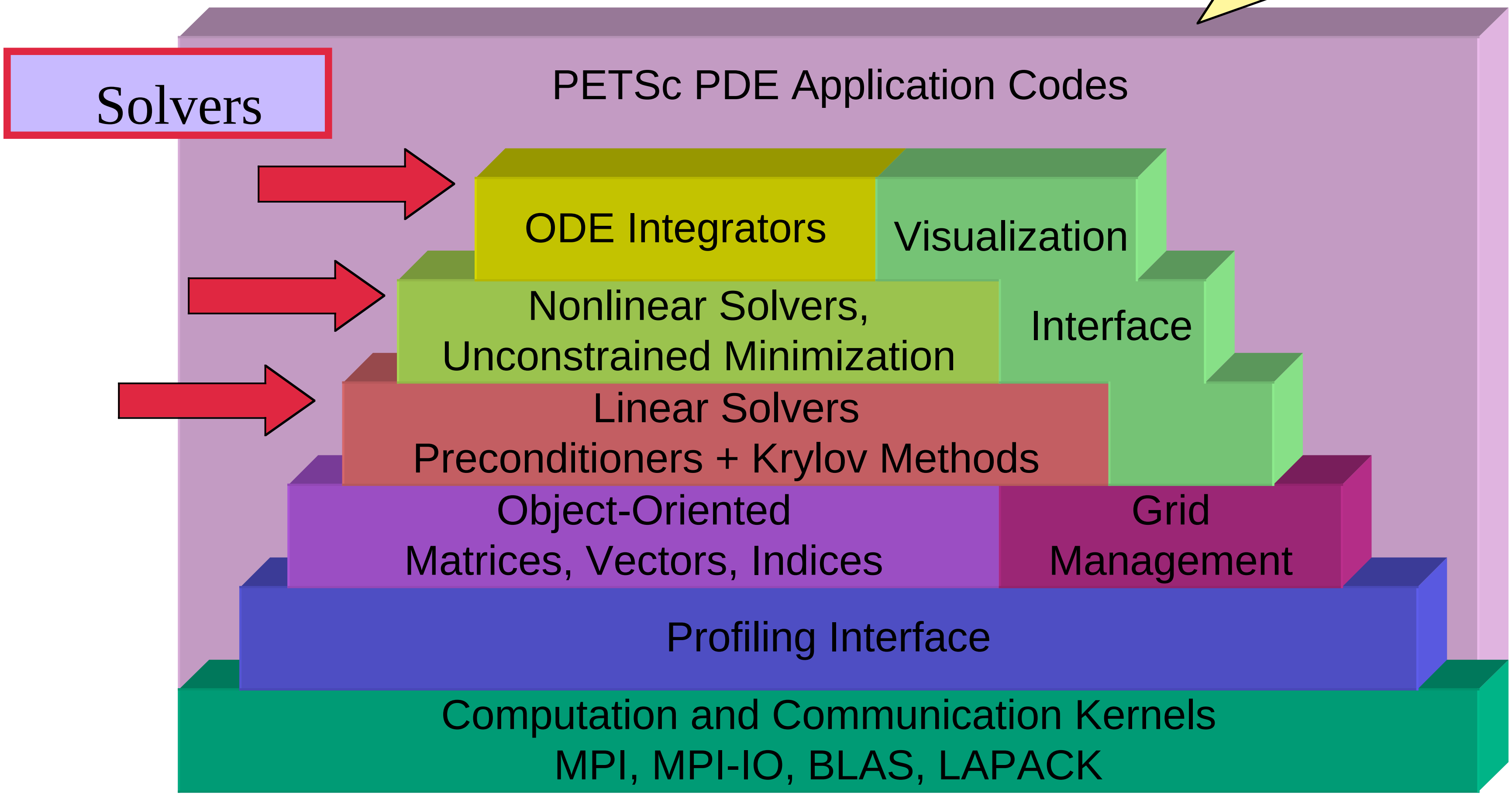
Structure of PETSc

How to specify the mathematics of the problem?



Structure of PETSc

How to solve the problem?

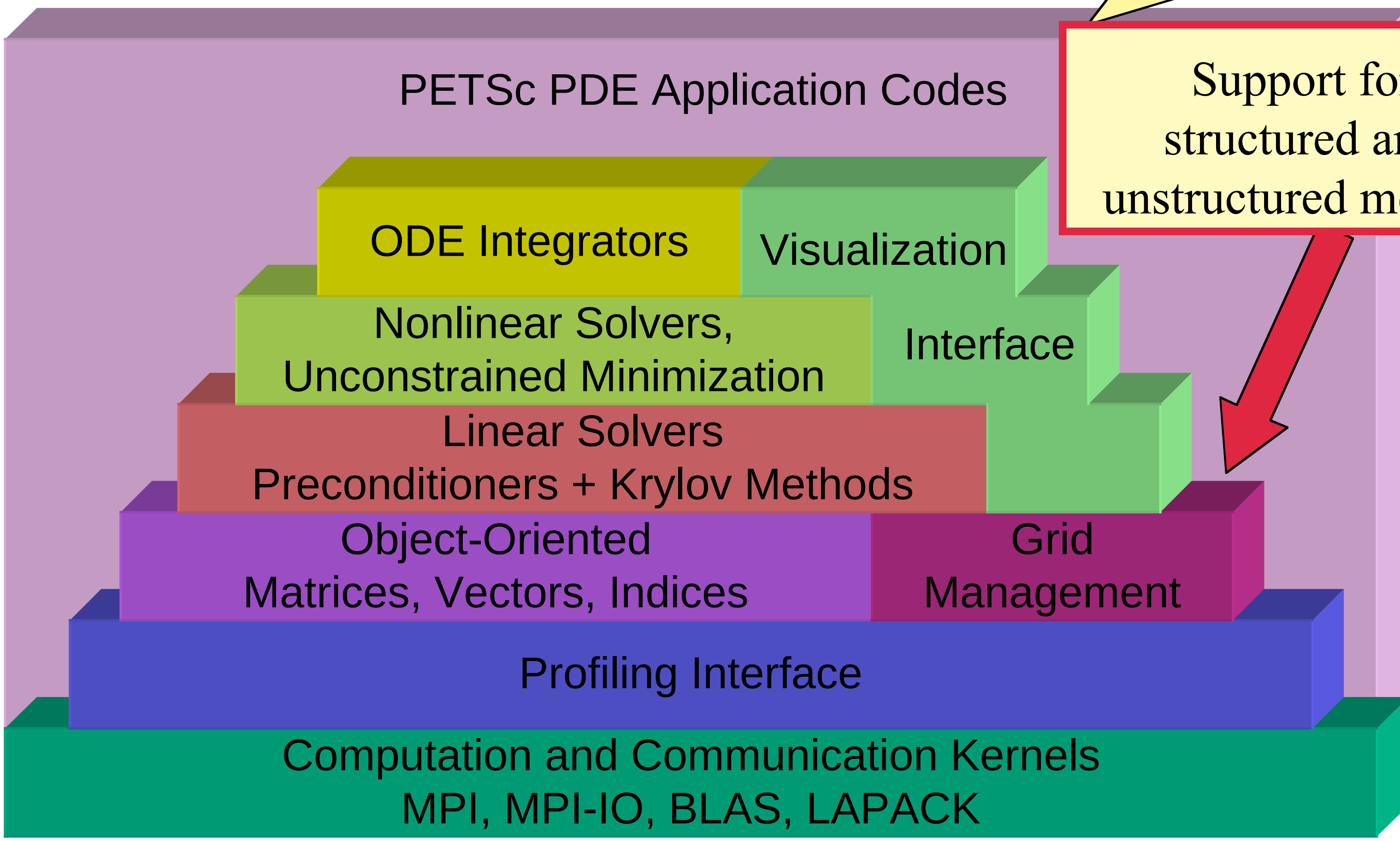


ACTS Toolkit

Structure of PETSc

How to handle Parallel computations?

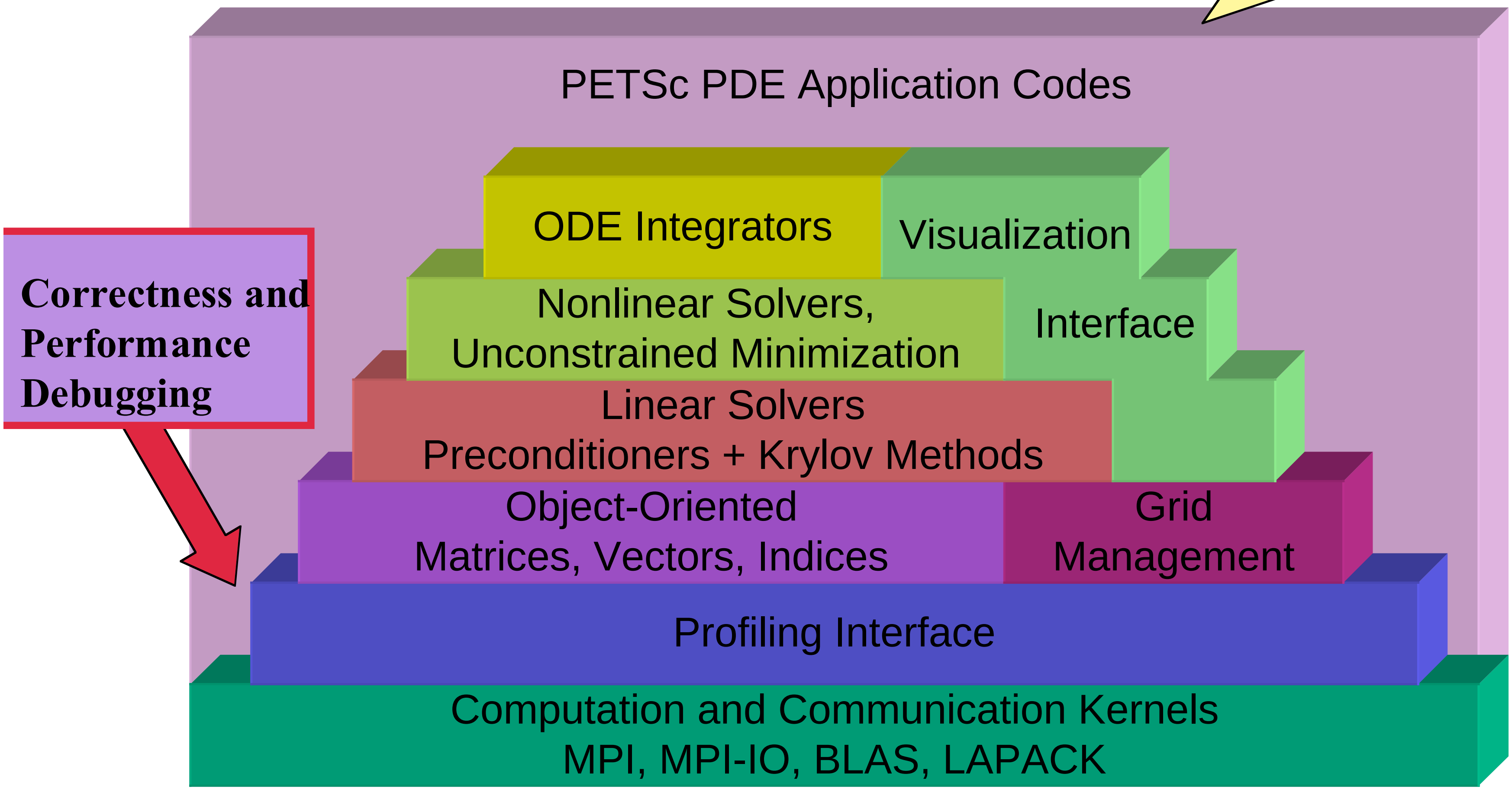
Support for structured and unstructured meshes



ACTS Toolkit

Structure of PETSc

What debugging and monitoring aids it provides?



ACTS Toolkit

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	Chebychev	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

Index Sets

Indices	Block Indices	Stride	Other

Vectors

A simple C example using PETSc
Hello World

```
#include "petsc.h"
int main( int argc, char *argv[] )
{
  PetscInitialize(&argc,&argv);
  PetscPrintf(PETSC_COMM_WORLD,"Hello World\n");
  PetscFinalize();
  return 0;
}
```

ACTS Toolkit

A simple FORTRAN example using PETSc

Hello World

```
    program main
    integer ierr, rank
#include "include/finclude/petsc.h"
    call PetscInitialize( PETSC_NULL_CHARACTER, ierr )
    call MPI_Comm_rank( PETSC_COMM_WORLD, rank, ierr )
    if (rank .eq. 0) then
        print *, ‘Hello World’
    endif
    call PetscFinalize(ierr)
    end
```

A fancier C example using PETSc
Hello World

```
#include "petsc.h"
int main( int argc, char *argv[] )
{
  int rank;
  PetscInitialize(&argc,&argv);
  MPI_Comm_rank(PETSC_COMM_WORLD,&rank );
  PetscSynchronizedPrintf(PETSC_COMM_WORLD,
                          "Hello World from %d\n",rank);
  PetscSynchronizedFlush(PETSC_COMM_WORLD);
  PetscFinalize();
  return 0;
}
```

Vectors and Matrices

Vectors

Fundamental objects for storing field solutions, right-hand sides, etc.

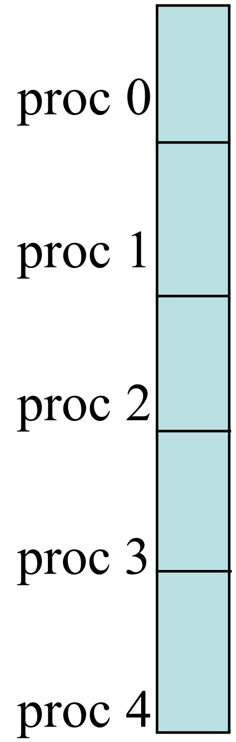
Matrices

Fundamental objects for storing linear operators (e.g., Jacobians)

ACTS Toolkit

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 *)

ACTS Toolkit

PETSc - Vector Example (in C)

```
#include petscvec.h
int main(int argc,char **argv)
 {
     Vec          x;                    /* array of vectors */
     int         n = 20,m=4, ierr;
     PetscInitialize(&argc,&argv);

      VecCreate(PETSC_COMM_WORLD,&x);
      VecSetSizes(x,PETSC_DECIDE,n);
      VecSetFromOptions(x);
         <-- perform some array operations -->

    PetscFinalize();
    return 0;
}
```

ACTS Toolkit

PETSc - Vector Example (in C)

```
#include petscvec.h
int main(int argc,char **argv)
 {
    Vec        x;                  /* array of vectors */
    int        n = 20,m=4, ierr;
    PetscInitialize(&argc,&argv);


       VecCreateMPI(PETSC_COMM_WORLD, m, n, x);


       <-- perform some array operations -->

   PetscFinalize();
   return 0;
}
```

ACTS Toolkit

Selected Vector Operations

Function Name	Operation
VecAXPY(Scalar *a, Vec x, Vec y)	$y = y + a*x$
VecAYPX(Scalar *a, Vec x, Vec y)	$y = x + a*y$
VecWAXPY(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)	$r = \|\|x\|\|$

ACTS Toolkit

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)

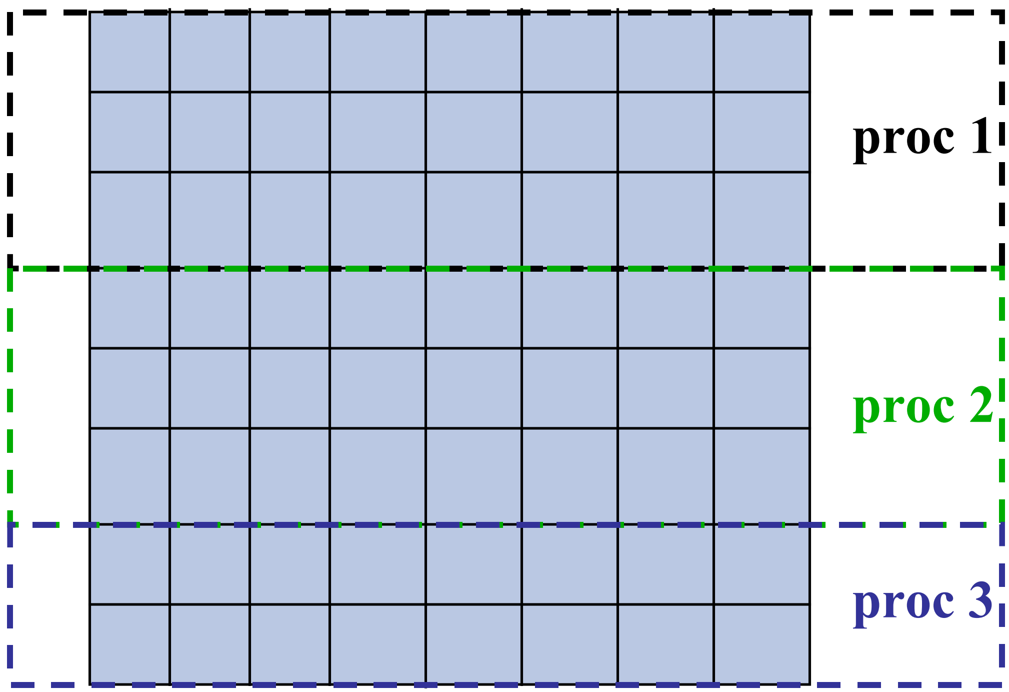
ACTS Toolkit

Matrices (Polymorphism)

- 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.

Parallel Matrix (Global vs. Local)

Each process locally owns a submatrix of contiguously numbered global rows.



M=8,N=8,m=3,n=8
rstart=0,rend=4

M=8,N=8,m=3,n=8
rstart=3,rend=6

M=8,N=8,m=2,n=8
rstart=6,rend=8

MatGetOwnershipRange(Mat A, int *rstart, int *rend)

- rstart: first locally owned row of global matrix
- rend -1: last locally owned row of global matrix

PETSc Example - Matrix and Vector manipulations

```
int main(int argc,char **args)
  {
    Vec         x, b, u;      /* approx solution, RHS, exact solution */
    Mat         A;            /* linear system matrix */
    int         ierr,i,n = 10,col[3],its,size;
    PetscScalar neg_one = -1.0,one = 1.0,value[3];

    PetscInitialize(&argc,&args,(char *)0,help);
    MPI_Comm_size(PETSC_COMM_WORLD,&size);
    if (size != 1) SETERRQ(1,"This is a uniprocessor example only!");
   PetscOptionsGetInt(PETSC_NULL,"-n",&n,PETSC_NULL);

 VecCreate(PETSC_COMM_WORLD,&x);
 VecSetSizes(x,PETSC_DECIDE,n);
 VecSetFromOptions(x);
 VecDuplicate(x,&b);
 VecDuplicate(x,&u);
```

ACTS Toolkit

PETSc Example - Matrix and Vector manipulations (cont.)

```
MatCreate(PETSC_COMM_WORLD,PETSC_DECIDE,PETSC_DECIDE,n,n,&A);
MatSetFromOptions(A);
value[0] = -1.0; value[1] = 2.0; value[2] = -1.0;
for (i=1; i<n-1; i++) {
 col[0] = i-1; col[1] = i; col[2] = i+1;
  MatSetValues(A,1,&i,3,col,value,INSERT_VALUES);
 }
i = n - 1; col[0] = n - 2; col[1] = n - 1;
MatSetValues(A,1,&i,2,col,value,INSERT_VALUES);
i = 0; col[0] = 0; col[1] = 1; value[0] = 2.0; value[1] = -1.0;
MatSetValues(A,1,&i,2,col,value,INSERT_VALUES);
MatAssemblyBegin(A,MAT_FINAL_ASSEMBLY);
MatAssemblyEnd(A,MAT_FINAL_ASSEMBLY);
 VecSet(&one,u);
 MatMult(A,u,b);
          <… Perform some other calculations see next example . .>
 VecDestroy(x); VecDestroy(u);  VecDestroy(b); MatDestroy(A);
PetscFinalize();
}
```

PETSc Viewers

- Information about PETSc objects
  - runtime choices for solvers, nonzero info for matrices, etc.
- Data for later use in restarts or external tools
  - vector fields, matrix contents
  - various formats (ASCII, binary)
- Visualization
  - *simple* x-window graphics
    - vector fields
    - matrix sparsity structure

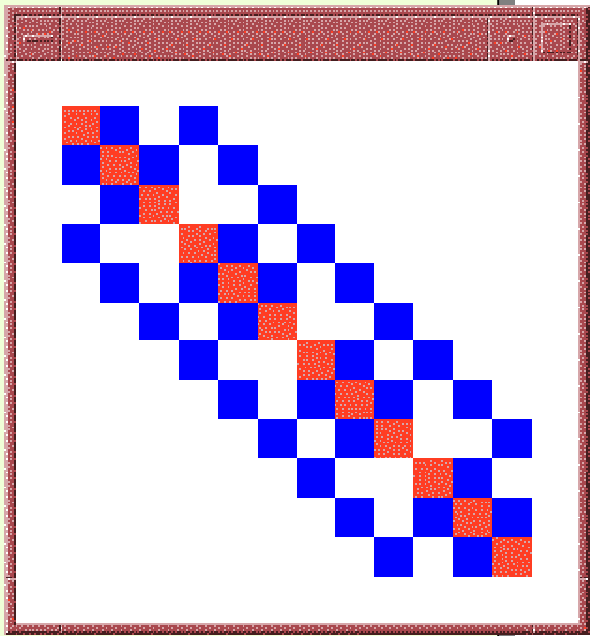
PETSc Viewers
Viewing a PETSc Matrix

- MatView(Mat A, PetscViewer v);
- Runtime options available after matrix assembly
  - -mat_view_info
    - info about matrix assembly
  - -mat_view_draw
    - sparsity structure
  - -mat_view
    - data in ASCII
  - etc.

ACTS Toolkit

PETSc Viewers
Viewing a PETSc Matrix

- MatView(Mat A, PetscViewer v);
- Runtime options available after matrix assembly
  - -mat_view_info
    - info about matrix assembly
  - -mat_view_draw
    - sparsity structure
  - -mat_view
    - data in ASCII
  - etc.



ACTS Toolkit

PETSc Linear Solvers

Goal: Support the solution of linear systems,

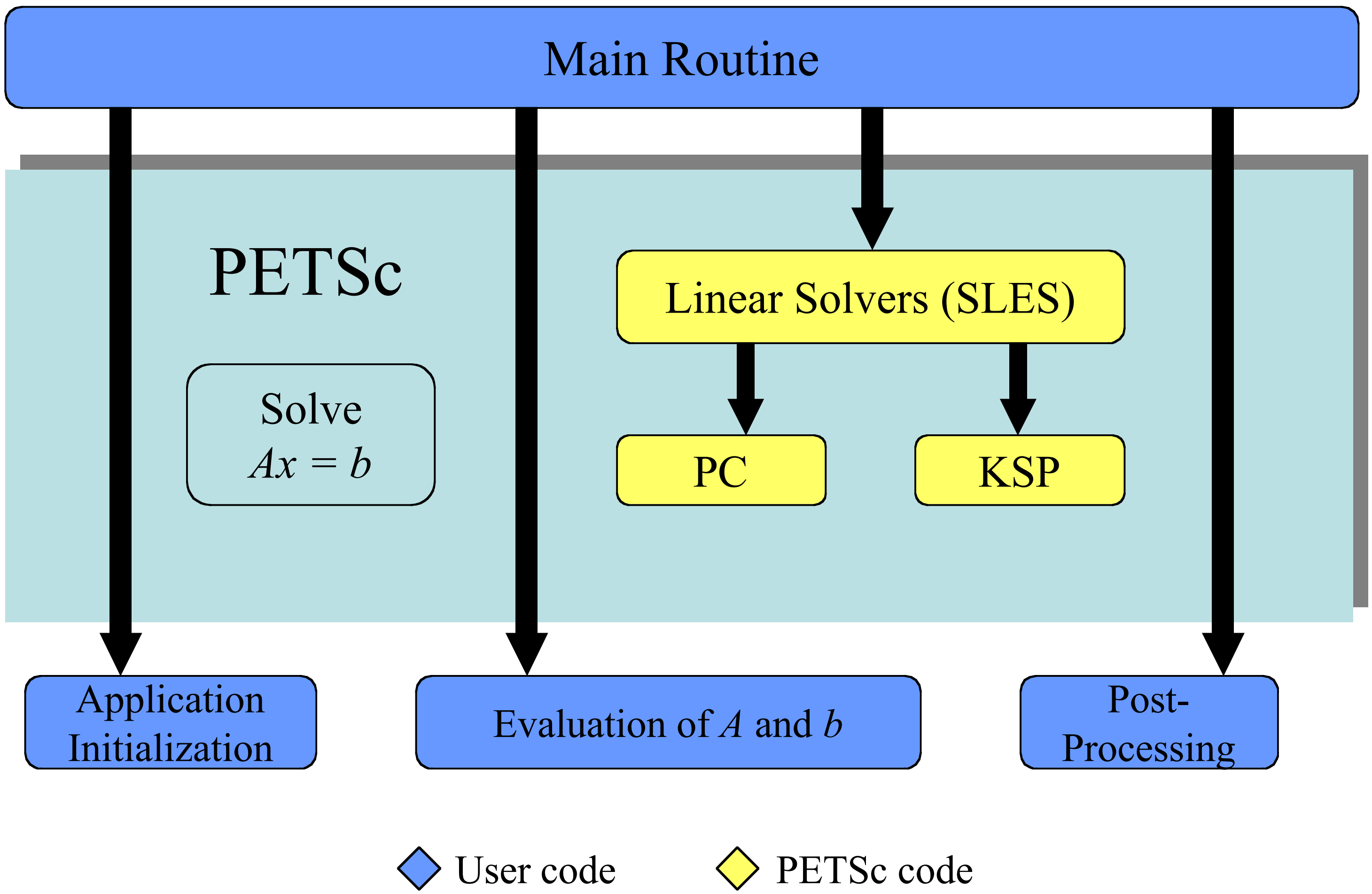
$$Ax=b,$$

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

User provides:

– Code to evaluate A, b

PETSc Linear Solvers (SLES)



ACTS Toolkit

Context Variables

- Are the key to solver organization
- Contain the complete state of an algorithm, including
  - parameters (e.g., convergence tolerance)
  - functions that run the algorithm (e.g., convergence monitoring routine)
  - information about the current state (e.g., iteration number)

ACTS Toolkit

Context Variables

- C/C++ version

```
ierr = SLESCreate(MPI_COMM_WORLD,&sles);
```

- Fortran version

```
call SLESCreate(MPI_COMM_WORLD,sles,ierr)
```

- Provides an **identical** user interface for all linear solvers
  - uniprocessor and parallel
  - real and complex numbers

ACTS Toolkit

Linear Solvers in PETSc 2.1.1

Krylov Methods (KSP)

- Conjugate Gradient
- GMRES
- CG-Squared
- Bi-CG-stab
- Transpose-free QMR
- etc.

Preconditioners (PC)

- Block Jacobi
- Overlapping Additive Schwarz
- ICC, ILU via BlockSolve95
- ILU(k), LU (sequential only)
- etc.

ACTS Toolkit

PETSc Example - Basic Linear Solver in C

```
SLES   sles;          /*  linear solver context  */
Mat    A;             /*  matrix  */
Vec    x, b;          /*  solution, RHS vectors  */
int    n, its;        /*  problem dimension, number of iterations  */

MatCreate(MPI_COMM_WORLD,PETSC_DECIDE,PETSC_DECIDE,
          n,n,&A);                          /* assemble matrix */
VecCreate(MPI_COMM_WORLD,PETSC_DECIDE,n,&x);
VecDuplicate(x,&b);                         /* assemble RHS vector */

SLESCreate(MPI_COMM_WORLD,&sles);
SLESSetOperators(sles,A,A,DIFFERENT_NONZERO_PATTERN);
SLESSetFromOptions(sles);
SLESSolve(sles,b,x,&its);
SLESDestroy(sles);
```

ACTS Toolkit

PETSc Example - Basic Linear Solver in Fortran

```
SLES     sles
Mat      A
Vec      x, b
integer  n, its, ierr

call  MatCreate(MPI_COMM_WORLD,PETSC_DECIDE,n,n,A,ierr)
call  VecCreate(MPI_COMM_WORLD,PETSC_DECIDE,n,x,ierr)
call  VecDuplicate(x,b,ierr)

<… Ensamble matrix and right-hand side (see previous example) . .>

call  SLESCreate(MPI_COMM_WORLD,sles,ierr)
call  SLESSetOperators(sles,A,A,DIFFERENT_NONZERO_PATTERN,ierr)
call  SLESSetFromOptions(sles,ierr)
call  SLESSolve(sles,b,x,its,ierr)
call  SLESDestroy(sles,ierr)
```

ACTS Toolkit

Setting SLES Parameters Inside The Program

- SLESGetKSP(SLES sles,KSP *ksp)
  - KSPSetType(KSP ksp,KSPType type)
  - KSPSetTolerances(KSP ksp,PetscReal rtol, PetscReal atol,PetscReal dtol, int maxits)
  - many more (see manual)

- SLESGetPC(SLES sles,PC *pc)
  - PCSetType(PC pc,PCType)
  - PCASMSetOverlap(PC pc,int overlap)
  - etc....

ACTS Toolkit

Setting SLES Parameters at Run Time

- -ksp_type [cg,gmres,bcgs,tfqmr,…]
- -pc_type [lu,ilu,jacobi,sor,asm,…]

More advanced:

- -ksp_max_it <max_iters>
- -ksp_gmres_restart <restart>
- -pc_asm_overlap <overlap>
- -pc_asm_type [basic,restrict,interpolate,none]
- Many more (see manual)

Setting SLES Parameters at Run Time

- -ksp_monitor - Prints preconditioned residual norm
- -ksp_xmonitor - Plots preconditioned residual norm

- -ksp_truemonitor - Prints true residual norm || b-Ax ||
- -ksp_xtruemonitor - Plots true residual norm || b-Ax ||

- User can also define their own monitors using call backs

ACTS Toolkit

Summary of Setting SLES Parameters

Functionality	Procedural Interface	Runtime Option
Set Krylov method	KSPSetType( )	-ksp_type [cg,gmres,bcgs, tfqmr,cgs,…]
Set monitoring routine	KSPSetMonitor( )	-ksp_monitor, –ksp_xmonitor, -ksp_truemonitor, -ksp_xtruemonitor
Set convergence tolerances	KSPSetTolerances( )	-ksp_rtol <rt> -ksp_atol <at> -ksp_max_its <its>
Set GMRES restart parameter	KSPGMRESSetRestart( )	-ksp_gmres_restart <restart>
Set orthogonalization routine for GMRES	KSPGMRESSet Orthogonalization( )	-ksp_unmodifiedgramschmidt -ksp_irorthog

ACTS Toolkit

Nonlinear Solver

Goal: For problems arising from PDEs, support the general solution of

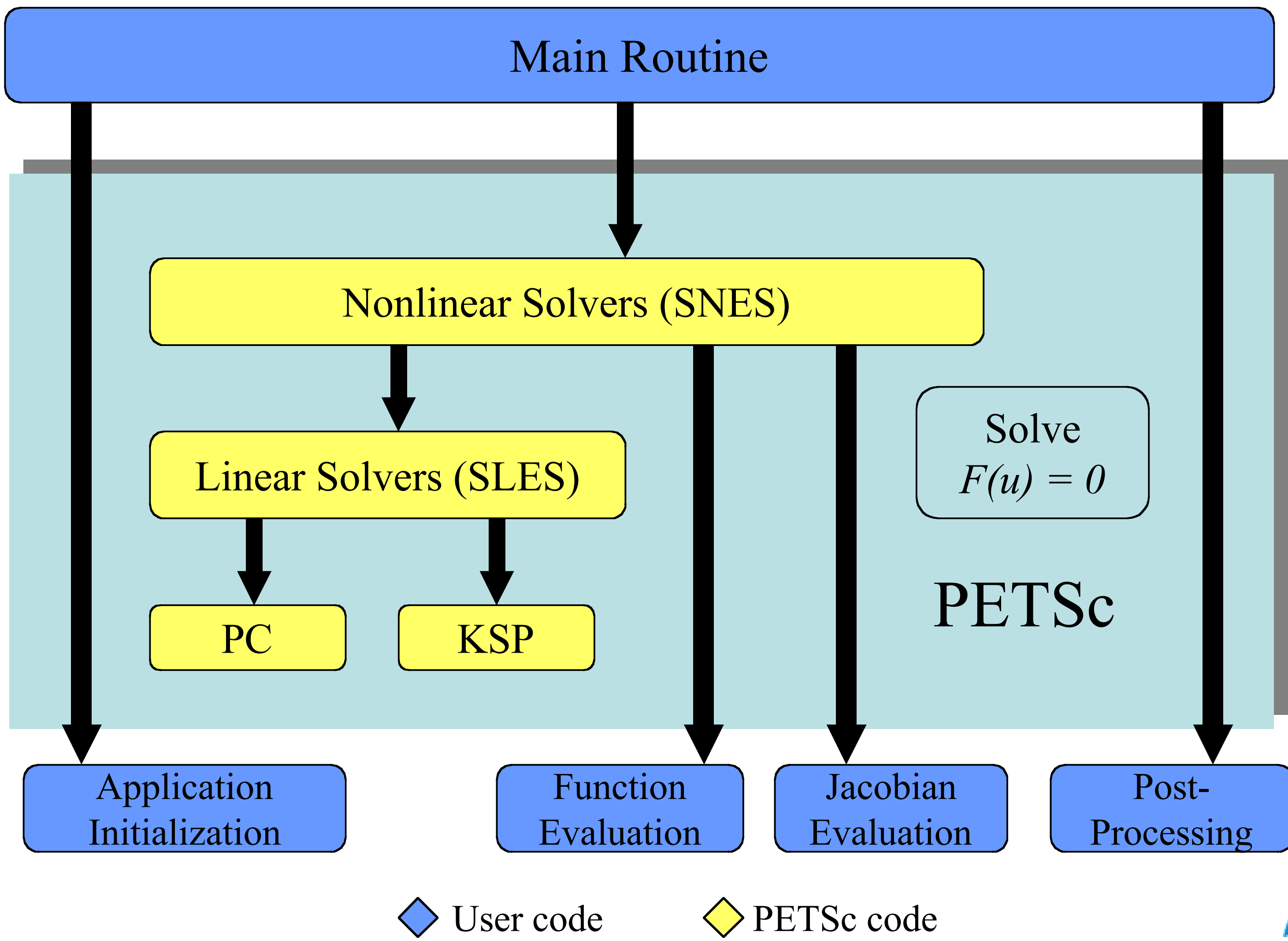
$$F(u) = 0$$

User provides:

- Code to evaluate *F(u)*
- Code to evaluate Jacobian of *F(u)* (optional)
  - or use sparse finite difference approximation
  - or use automatic differentiation
    - AD support via collaboration with P. Hovland and B. Norris
    - Via automated interface to ADIFOR and ADIC (see http://www.mcs.anl.gov/autodiff)

ACTS Toolkit

PETSc Non-Linear Solvers (SLES)



ACTS Toolkit

Nonlinear Solver

- Newton-based methods, including
  - Line search strategies
  - Trust region approaches
  - Pseudo-transient continuation
  - Matrix-free variants
- User can customize all phases of the solution process

ACTS Toolkit

PETSc Example - Basic Nonlinear Solver in C

```
SNES   snes;                          /*  nonlinear solver context  */
Mat      J;                           /*  Jacobian matrix  */
Vec      x, F;                        /*  solution, residual vectors  */
int       n, its;                     /*  problem dimension, number of iterations  */
ApplicationCtx  usercontext;          /*  user-defined application context  */

...
MatCreate(MPI_COMM_WORLD,PETSC_DECIDE,PETSC_DECIDE,n,n,&J);
VecCreate(MPI_COMM_WORLD,PETSC_DECIDE,n,&x);
VecDuplicate(x,&F);

SNESCreate(MPI_COMM_WORLD,SNES_NONLINEAR_EQUATIONS,&snes);
SNESSetFunction(snes,F,EvaluateFunction,usercontext);
SNESSetJacobian(snes,J,EvaluateJacobian,usercontext);
SNESSetFromOptions(snes);
SNESSolve(snes,x,&its);
SNESDestroy(snes);
```

ACTS Toolkit

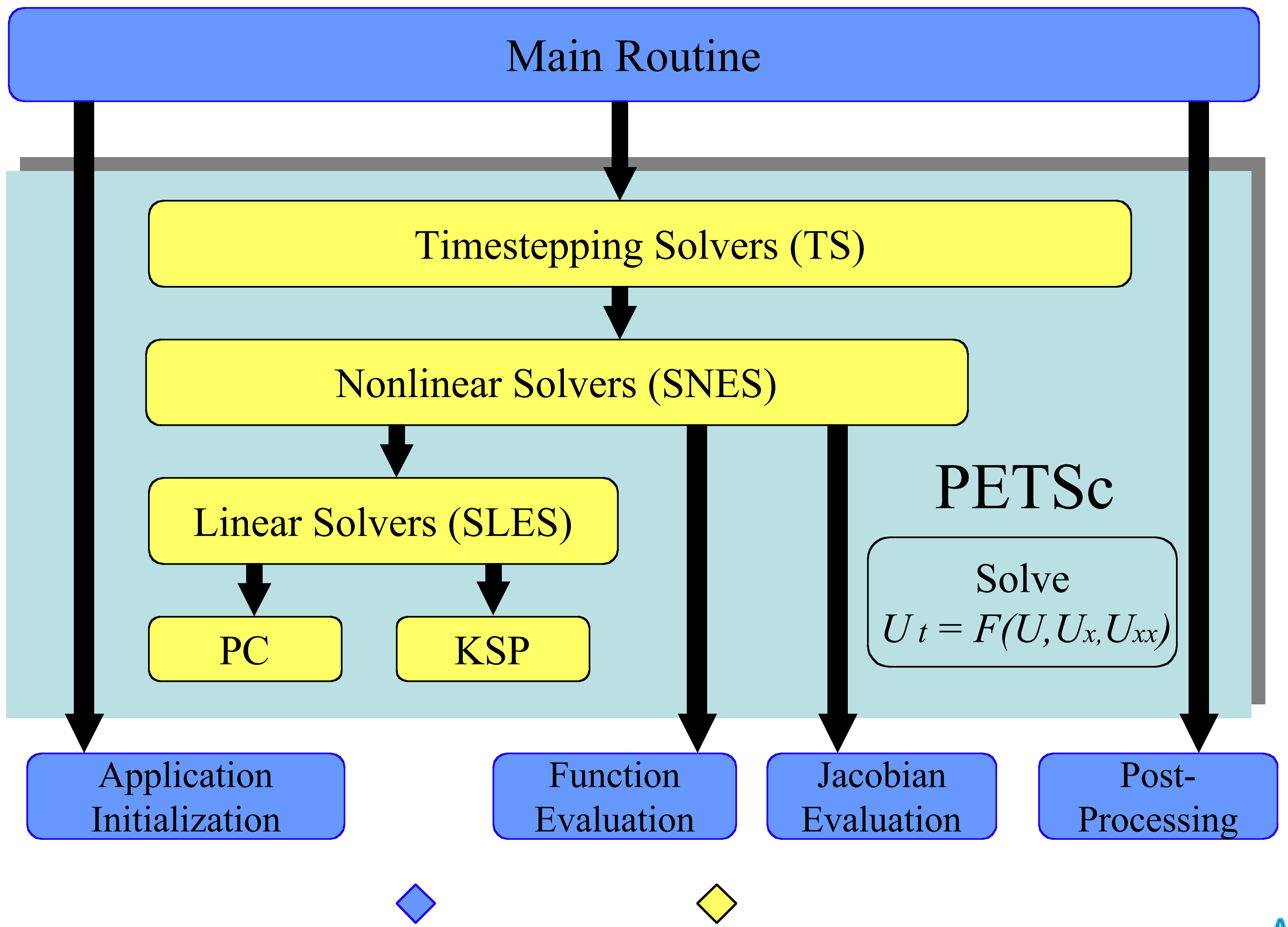
PETSc Example - Basic Nonlinear Solver in Fortran

```
    SNES   snes
    Mat    J
    Vec    x, F
    int    n, its

      ...
    call  MatCreate(MPI_COMM_WORLD,n,n,J,ierr)
    call  VecCreate(MPI_COMM_WORLD,n,x,ierr)
    call  VecDuplicate(x,F,ierr)

    call  SNESCreate(MPI_COMM_WORLD,SNES_NONLINEAR_EQUATIONS,
   &                          snes,ierr)
    call  SNESSetFunction(snes,F,EvaluateFunction,PETSC_NULL,ierr)
    call  SNESSetJacobian(snes,J,EvaluateJacobian,PETSC_NULL,ierr)
    call  SNESSetFromOptions(snes,ierr)
    call  SNESSolve(snes,x,its,ierr)
    call  SNESDestroy(snes,ierr)
```

PETSc Nonlinear Solver (based on callbacks)

- User provides routines to perform actions that the library requires. For example,
  - SNESSetFunction(SNES,...)
    - uservector - vector to store function values
    - userfunction - name of the user’s function
    - usercontext - pointer to private data for the user’s function
- Now, whenever the library needs to evaluate the user’s nonlinear function, the solver may call the application code directly with its own local state.
- usercontext: serves as an application context object. Data are handled through such opaque objects; the library never sees irrelevant application data.

Time Dependent PDE Solution



ACTS Toolkit

PETSc Timestepping Solver (TS)

Goal: Support the (real and pseudo) time evolution of PDE systems

$$U_t = F(U,Ux,Uxx,t)$$

User provides:

- Code to evaluate $F(U,Ux,Uxx,t)$
- Code to evaluate Jacobian of $F(U,Ux,Uxx,t)$
  - or use sparse finite difference approximation
  - or use automatic differentiation

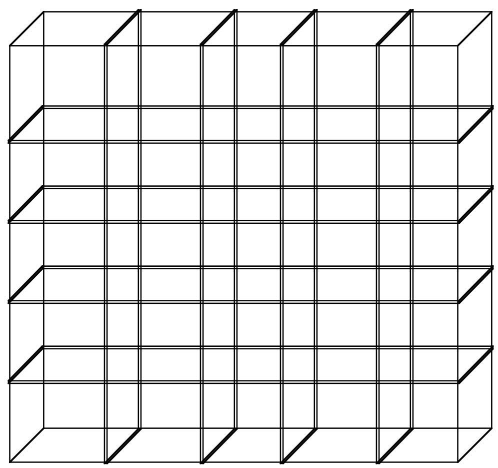
PETSc Timestepping Solvers (TS)

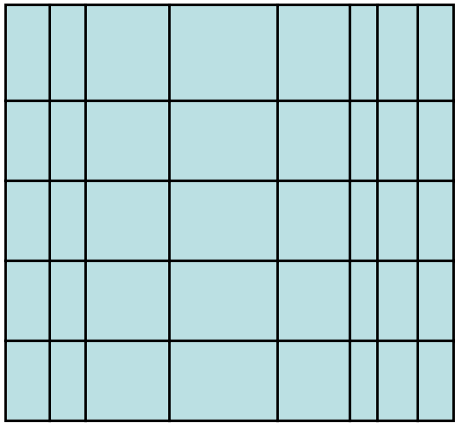
• TSCreate( )	- Create TS context
• TSSetRHSFunction( )	- Set function eval. routine
• TSSetRHSJacobian( )	- Set Jacobian eval. routine
• TSSetFromOptions( )	- Set runtime solver options for [TS,SNES,SLES,KSP,PC]
• TSSolve( )	- Run timestepping solver
• TSView( )	- View solver options actually used at runtime (alternative: -ts_view)
• TSDestroy( )	- Destroy solver

ACTS Toolkit

PETSc Support for Structured and Unstructured Meshes

- **Structured**: Determine neighbor relationships purely from logical I, J, K coordinates

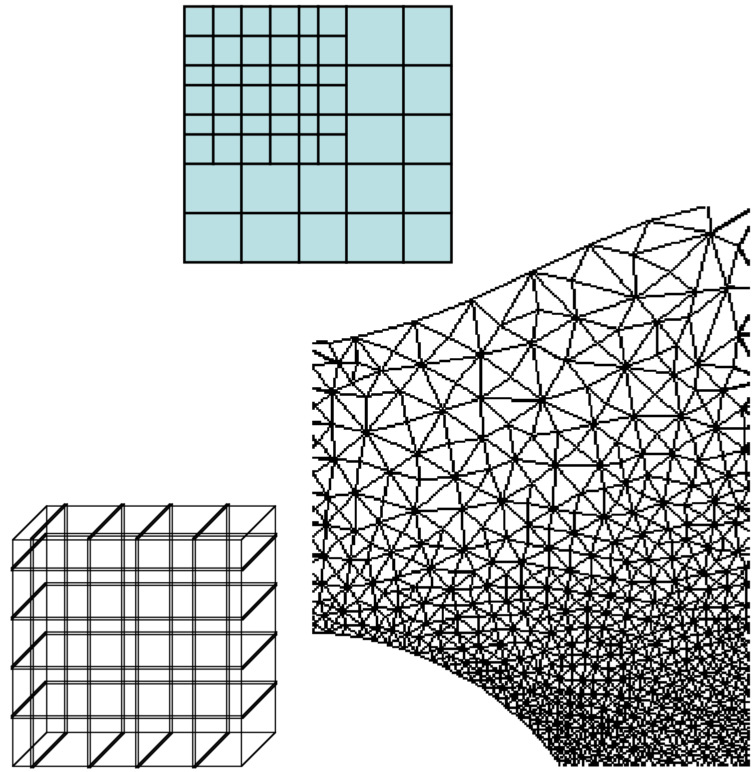




PETSc support provided via DA objects

PETSc Support for Structured and Unstructured Meshes

- **Unstructured**: Do not explicitly use logical I, J, K coordinates

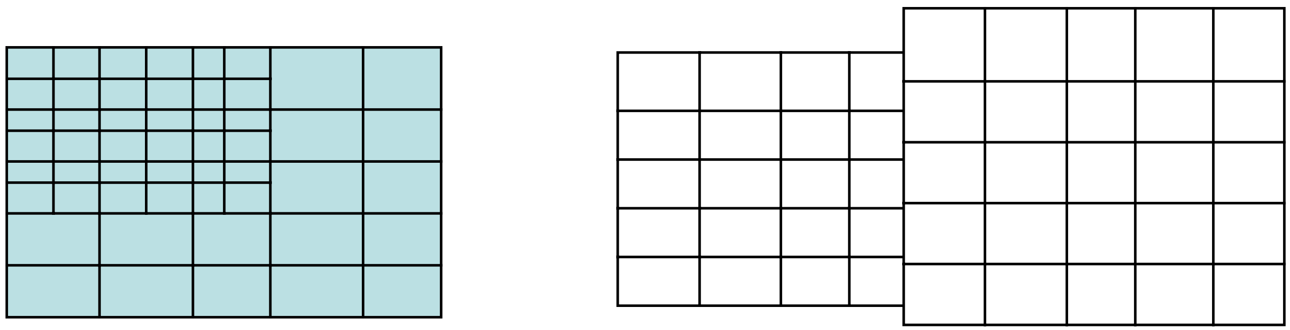


PETSc does not currently have high-level tools for managing such meshes (only support through VecScatter *objects)*

ACTS Toolkit

PETSc Support for Structured and Unstructured Meshes

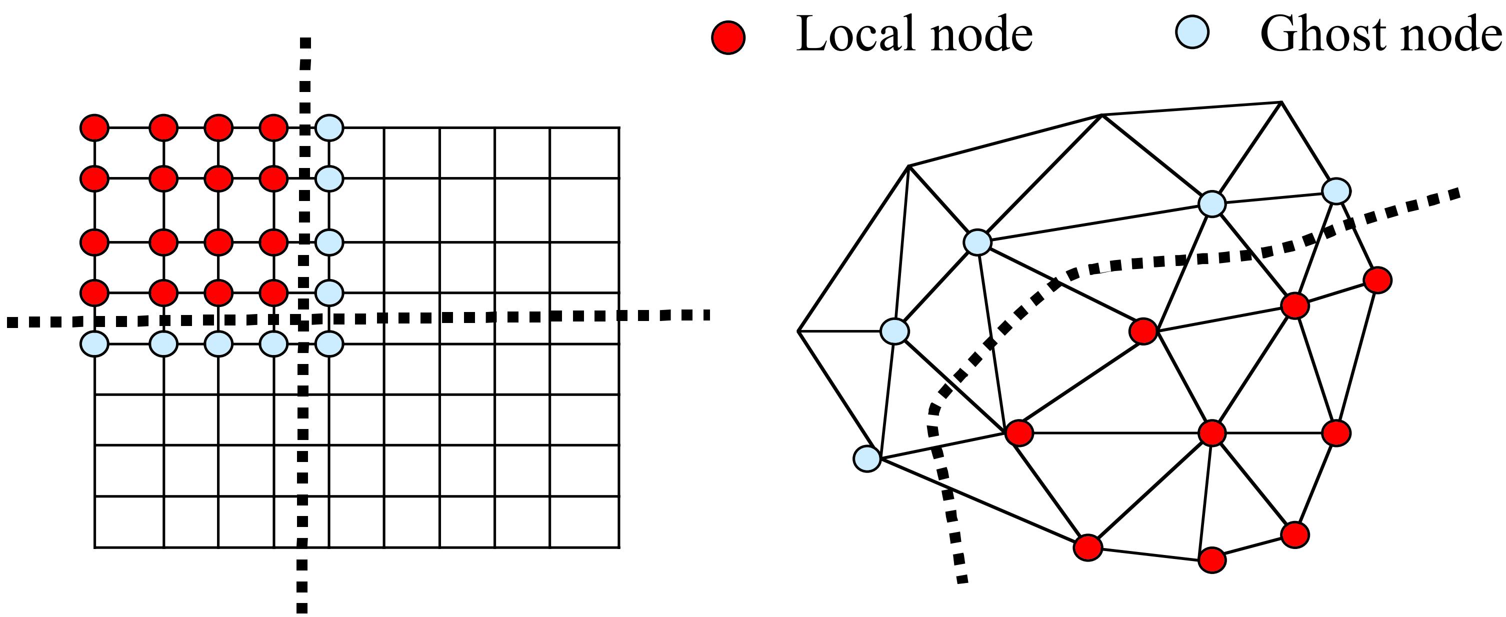
- **Semi-Structured**: In well-defined regions, determine neighbor relationships purely from logical I, J, K coordinates



- No explicit PETSc support
  - OVERTURE-PETSc for composite meshes
  - SAMRAI-PETSc for AMR

PETSc Concepts for managing Structured and Unstructured Meshes

Managing *field data layout* and required *ghost values* is the key to high performance of most PDE-based parallel programs.



Ghost values: To evaluate a local function *f(x)* , each process requires its local portion of the vector *x* as well as its **ghost values** – or bordering portions of *x* that are owned by neighboring processes.

PETSc Concepts for managing Structured and Unstructured Meshes

Communication

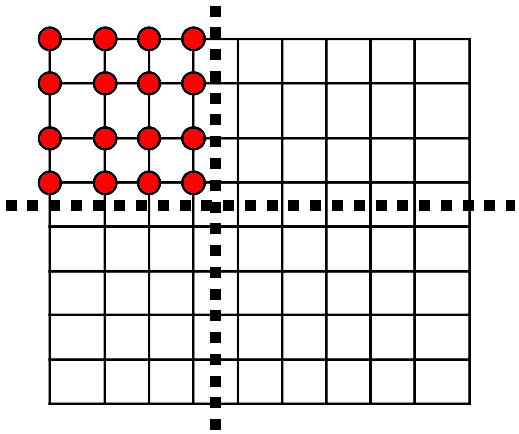
Geometric Data	Data Structure Creation	Ghost Point Data Structures	Ghost Point Updates	**Local Numerical Computation**
stencil [implicit]	DACreate( )	DA AO	DAGlobalToLocal( )	Loops over I,J,K indices
elements edges vertices	VecScatterCreate( )	VecScatter AO	VecScatter( )	Loops over entities

structured meshes

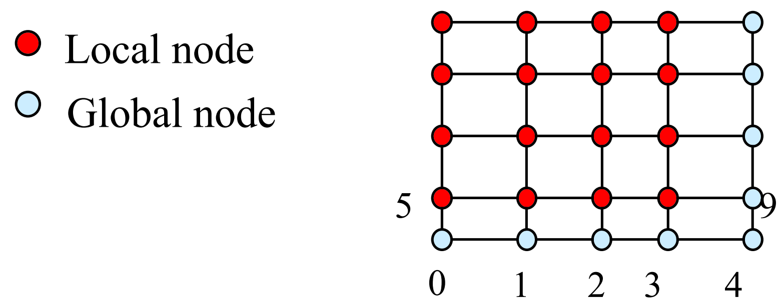
unstructured meshes

ACTS Toolkit

PETSc Concepts for managing Structured and Unstructured Meshes



Global: each process stores a unique local set of vertices (and each vertex is owned by exactly one process)



Local: each process stores a unique local set of vertices *as well as* ghost nodes from neighboring processes

- DA - Distributed Array: object containing information about vector layout across the processes and communication of ghost values
- Form a DA

  DACreateXX(….,DA *)
- Update ghostpoints

  DAGlobalToLocalBegin(DA,…) and DAGlobalToLocalEnd(DA,…)

ACTS Toolkit

PETSc Vectors and DAs

- The DA object contains information about the data layout and ghost values, but **not** the actual field data, which is contained in PETSc vectors
- Global vector: parallel
  - each process stores a unique local portion
  - DACreateGlobalVector(DA da,Vec *gvec);
- Local work vector: sequential
  - each processor stores its local portion plus ghost values
  - DACreateLocalVector(DA da,Vec *lvec);
  - uses "natural" local numbering of indices (0,1,…*nlocal*-1)

PETSc Debugging Support

- -start_in_debugger [gdb,dbx,noxterm]
- -on_error_attach_debugger [gb,dbx,noxterm]
- -on_error_abort
- -debugger_nodes 0,1
- -display machinename:0.0

Debugging with **PETSc**

Breakdown in ILU factorization due to a zero pivot

```
xterm
Buffers Files Tools Edit Search Mule Help
[dreamcast] mpirun -np 1 ex1
--------------------------------------------------------------------------
PETSc Version 2.1.0, Released April 11, 2001
       The PETSc Team     petsc-maint@mcs.anl.gov
 http://www.mcs.anl.gov/petsc/

See docs/copyright.html for copyright information.
See docs/changes.html for recent updates.
See docs/troubleshooting.html for hints about trouble shooting.
See docs/manualpages/index.html for manual pages.
-----------------------------------------------------------------------
ex1 on a linux named dreamcast.mcs.anl.gov by balay Thu Oct  4 15:25:11 2001
Libraries linked from /home/balay/software/petsc-2.1.0/lib/libg/linux
-----------------------------------------------------------------------
[0]PETSC ERROR: MatLUFactorNumeric_SeqAIJ() line 508 in src/mat/impls/aij/seq/aijfact.c
[0]PETSC ERROR:   Detected zero pivot in LU factorization!
[0]PETSC ERROR:   Zero pivot row 0!
[0]PETSC ERROR: MatLUFactorNumeric() line 1575 in src/mat/interface/matrix.c
[0]PETSC ERROR: PCSetUp_ILU() line 646 in src/sles/pc/impls/ilu/ilu.c
[0]PETSC ERROR: PCSetUp() line 783 in src/sles/pc/interface/precon.c
[0]PETSC ERROR: SLESSetUp() line 382 in src/sles/interface/sles.c
[0]PETSC ERROR: SLESSolve() line 483 in src/sles/interface/sles.c
[0]PETSC ERROR: main() line 195 in test/ex1.c
[0] MPI Abort by user Aborting program !
[0] Aborting program!
p0_5469:  p4_error: : 71
--1-:---F1  logs             (Text)--L3-- 2%------------------------------------
```

ACTS Toolkit

Debugging with **PETSc**

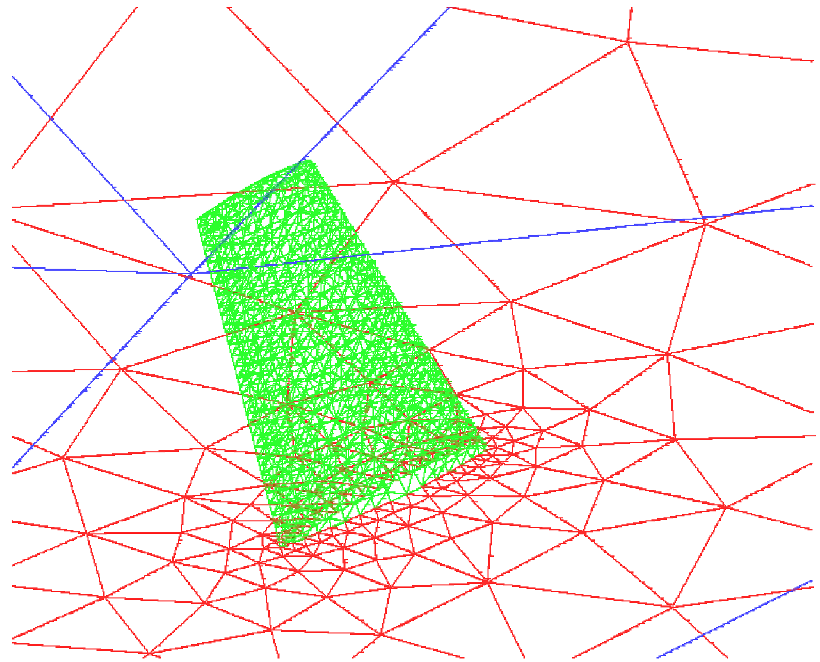
Sample Memory Corruption Error

```
xterm
Buffers Files Tools Edit Search Mule Help
[dreamcast] mpirun -np 1 ex2 -trmalloc_off
[dreamcast] mpirun -np 1 ex2 -trmalloc
---------------------------------------------------------------------------
PETSc Version 2.1.0, Released April 11, 2001
       The PETSc Team    petsc-maint@mcs.anl.gov
 http://www.mcs.anl.gov/petsc/

See docs/copyright.html for copyright information.
See docs/changes.html for recent updates.
See docs/troubleshooting.html for hints about trouble shooting.
See docs/manualpages/index.html for manual pages.
------------------------------------------------------------------------
ex2 on a linux named dreamcast.mcs.anl.gov by balay Thu Oct  4 15:35:29 2001
Libraries linked from /home/balay/software/petsc-2.1.0/lib/libg/linux
------------------------------------------------------------------------
PetscTrFreeDefault called from main() line 51 in test/ex2.c
Block [id=0(14)] at address 0x81152d8 is corrupted (probably write past end)
Block allocated in main() line 49 in test/ex2.c
[0]PETSC ERROR: PetscTrFreeDefault() line 363 in src/sys/src/memory/mtr.c
[0]PETSC ERROR:   Memory corruption!
[0]PETSC ERROR:   Corrupted memory!
[0]PETSC ERROR: main() line 51 in test/ex2.c
[0] MPI Abort by user Aborting program !
[0] Aborting program!
p0_5691:  p4_error: : 78
--1-:---F1  logs            (Text)--L32--27%------------------------------------
```

ACTS Toolkit

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)

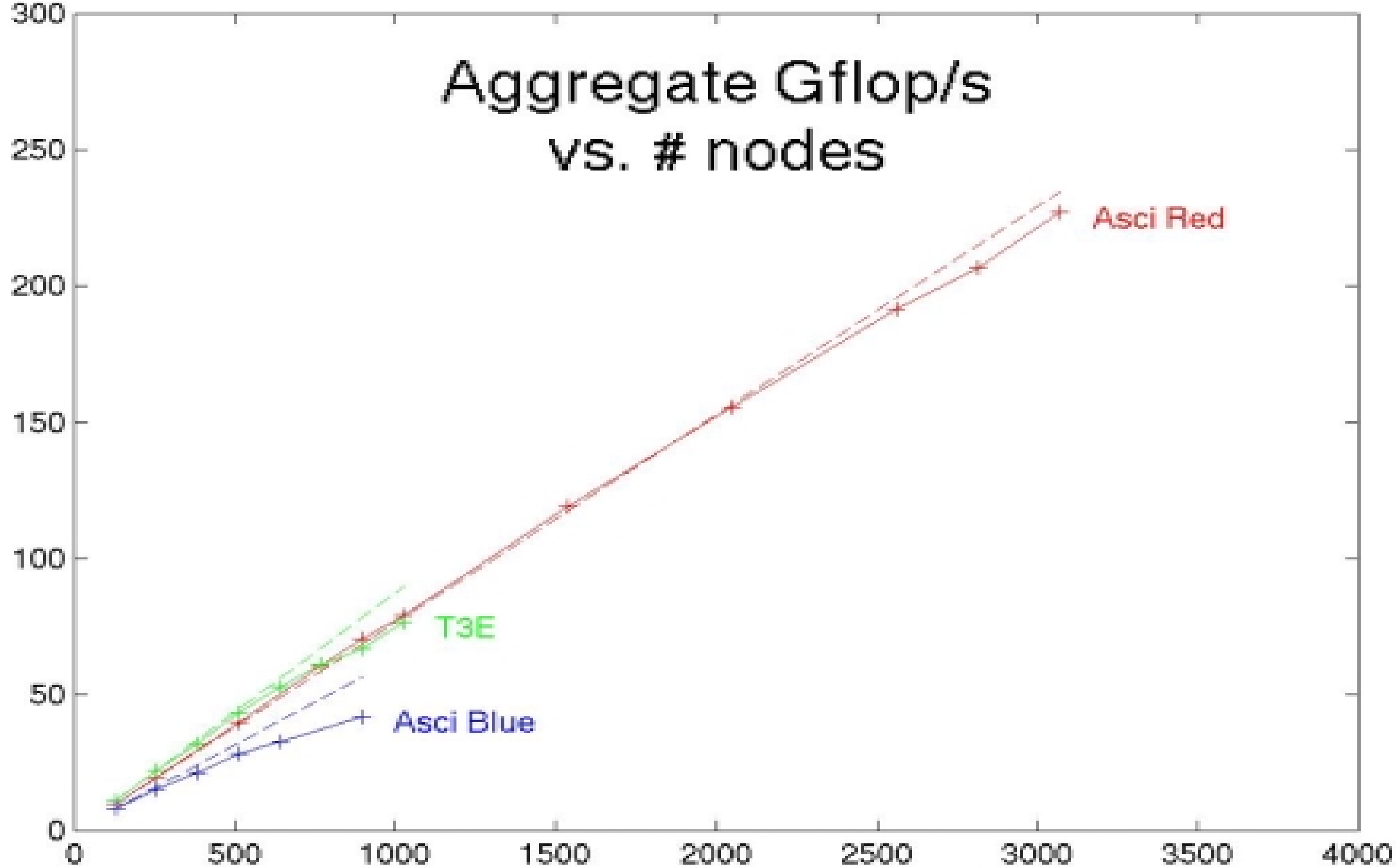


Results courtesy of Dinesh Kaushik and David Keyes, Old Dominion Univ., partially funded by NSF and ASCI level 2 grant

ACTS Toolkit

CFD Application using **PETSc**

Dimension=11,047,096

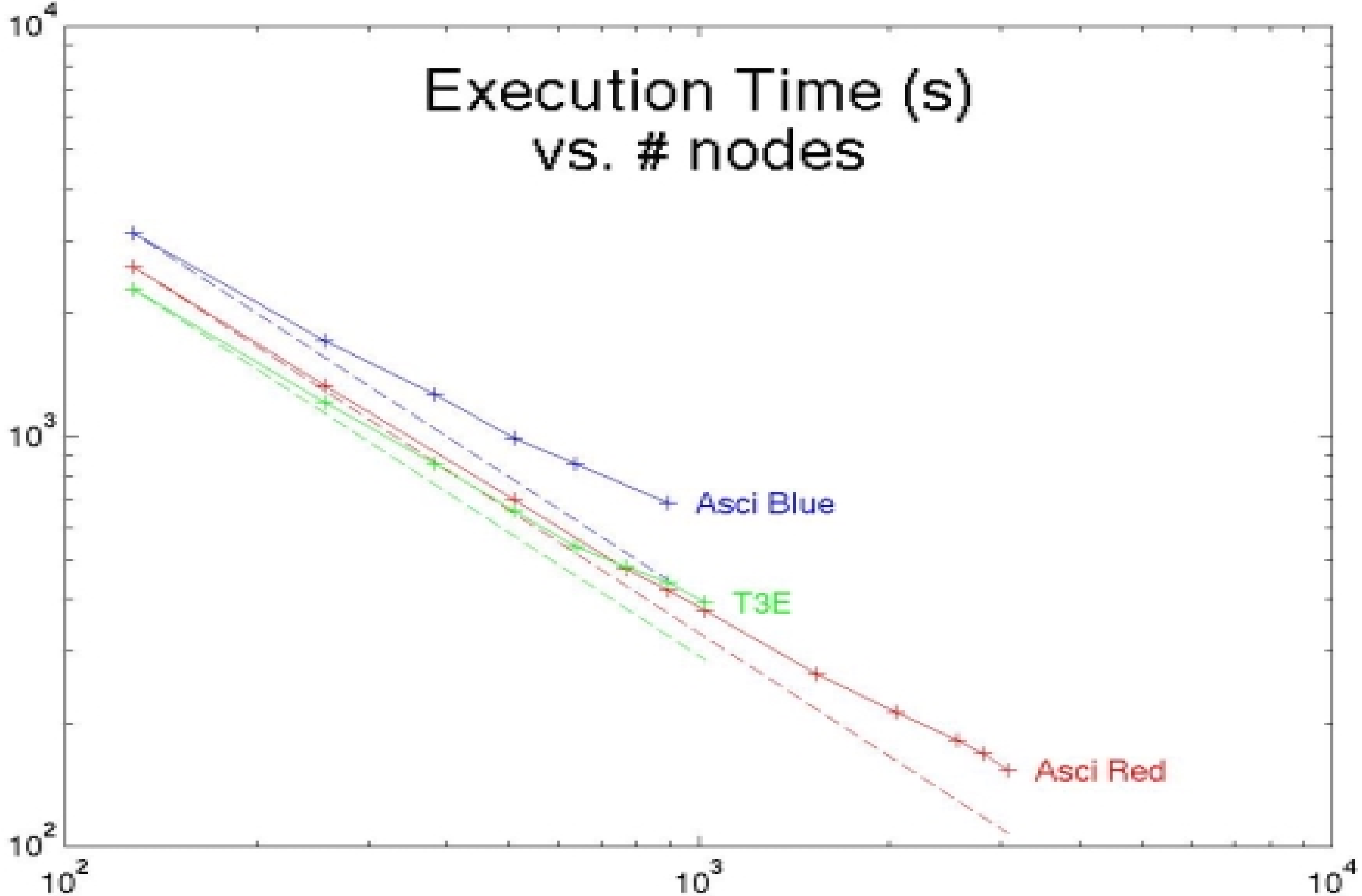


Fixed-size Parallel Scaling Results (GFlop/s)

ACTS Toolkit

CFD Application using **PETSc**

(Time in seconds)

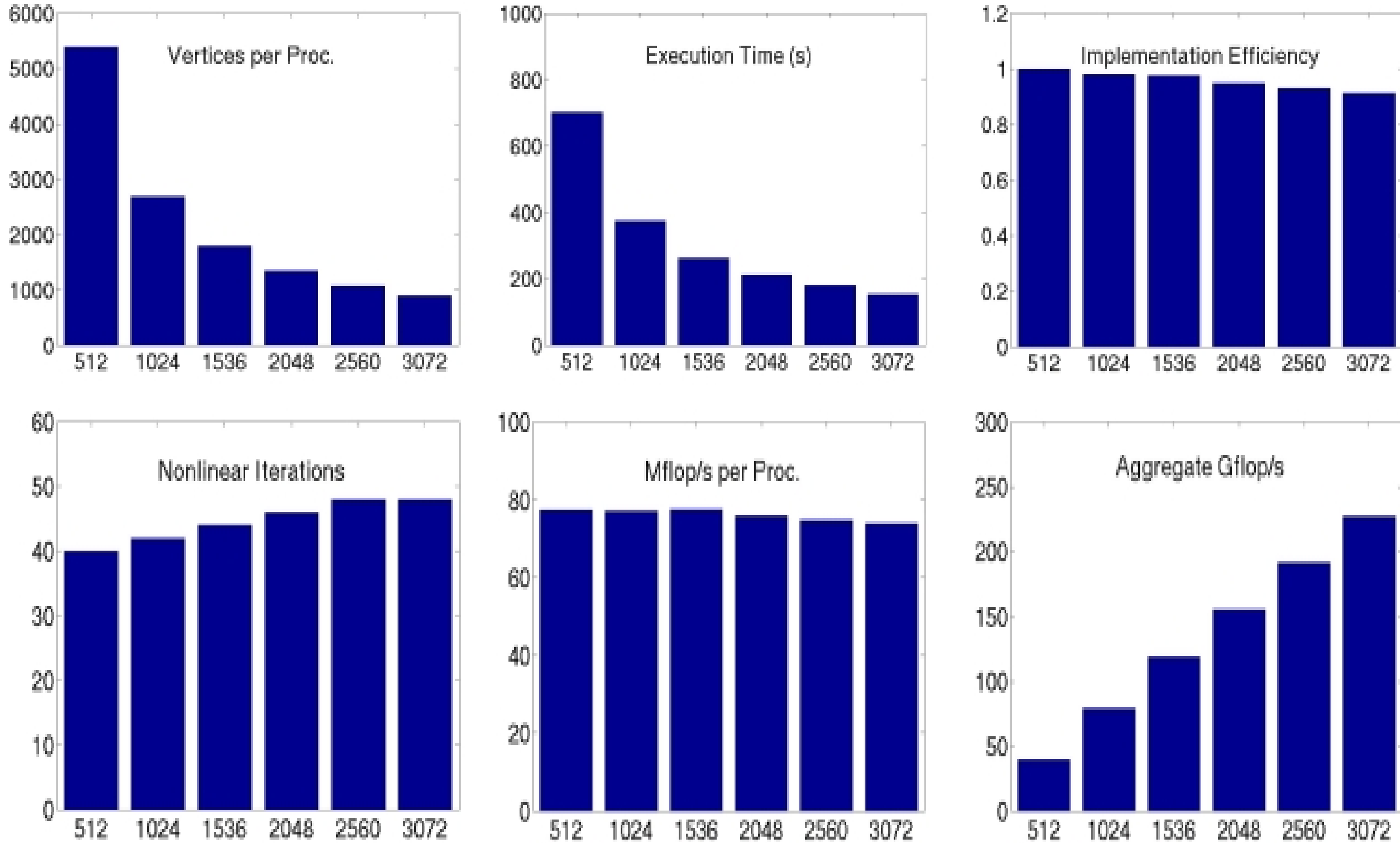


Fixed-size Parallel Scaling Results

ACTS Toolkit

CFD Application using **PETSc**
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)



Using PETSc with Other Packages

- **Linear solvers**
  - AMG http://www.mgnet.org/mgnet-codes-gmd.html
  - BlockSolve95 http://www.mcs.anl.gov/BlockSolve95
  - ILUTP http://www.cs.umn.edu/~saad/
  - LUSOL http://www.sbsi-sol-optimize.com
  - SPAI http://www.sam.math.ethz.ch/~grote/spai
  - SuperLU http://www.nersc.gov/~xiaoye/SuperLU
- **Optimization software**
  - TAO http://www.mcs.anl.gov/tao
  - Veltisto http://www.cs.nyu.edu/~biros/veltisto

Using PETSc with Other Packages

- **Mesh and discretization tools**
  - Overture http://www.llnl.gov/CASC/Overture
  - SAMRAI http://www.llnl.gov/CASC/SAMRAI
  - SUMAA3d http://www.mcs.anl.gov/sumaa3d
- **ODE solvers**
  - PVODE http://www.llnl.gov/CASC/PVODE
- **Others**
  - Matlab http://www.mathworks.com
  - ParMETIS
    http://www.cs.umn.edu/~karypis/metis/parmetis

ACTS Toolkit

Some references to PETSc material

- Documentation: http://www.mcs.anl.gov/petsc/docs
  - PETSc Users manual
  - Manual pages
  - Many hyperlinked examples
  - FAQ, Troubleshooting info, installation info, etc.
- ACTS Toolkit: http://acts.nersc.gov/petsc/main.html
- Publications: http://www.mcs.anl.gov/petsc/publications
  - Research and publications that make use PETSc
- MPI Information: http://www.mpi-forum.org
- ***Domain Decomposition***, by Smith, Bjorstad, and Gropp

ACTS Toolkit