SAND2011-5490C

ForTrilinos & Morfeus

Commercial Adoption and Application

Damian Rouson, Karla Morris, Nicole Lemaster
Sandia National Laboratories

Salvatore Filippone
University of Rome Tor Vergata

Brian Smith
ASAP, LLC, Numerica 21, Numerical Algorithms Group

Xiaofeng Xu
City University of New York

Sponsors: DOE, ONR, AFOSR

OUTLINE

- Why open-source?
- Software project objectives
  - Trilinos
  - ForTrilinos
  - Morfeus
- Target problems and platforms.
- Framework and library overview.
- SBIR Potential
  - ASAP, LLC
  - Numerica 21, Inc.
  - Numerical Algorithms Group
- ForTrilnos application prototype.
- Morfeus application prototype.

WHY OPEN-SOURCE?

Uniqueness of the national lab business model:

- Coordinated efforts on a larger scale & longer term than industry or academia:
  - → Massive resources
  - → Stable staffing
- Legal mandate to avoid competing with industry.
- Founding mandate to provide national service.

Open source leverages each of these!

Sandia National Laboratories

OBJECTIVES

Trilinos

- To establish uniform, professional software engineering practices across 55+ scalable solver and services packages totaling ~1M LOC.
- Philosophy: Generic, Parallel, Object-Oriented Design.
- Tools: Git, CMake, CTest, CDash, Bugzilla, Mailman.

ForTrilinos

- To increase the adoption of Trilinos in scientific research communities that principally write Fortran, e.g., energy and climate communities.
- To provide object-oriented Fortran 2003 interfaces to Trilinos C++ packages.

Morfeus

- To distill and disseminate object-oriented software design patterns for multiphysics modeling.
- To demonstrate these patterns on DOD mission problems.

TRILINOS CONTRIBUTORS

Chris Baker
Ross Bartlett
Pavel Bochev
Paul Boggs
Erik Boman
Cedric Chevalier
Todd Coffey
Eric Cyr
David Day
Karen Devine
Clark Dohrmann
Kelly Fermoyle
David Gay
Mike Heroux
Ulrich Hetmaniuk

Robert Hoekstra
Russell Hooper
Vicki Howle
Jonathan Hu
Joe Kotulski
Rich Lehoucq
Nicole Lemaster
Kevin Long
Karla Morris
Kurtis Nusbaum
Roger Pawlowski
Brent Perschbacher
Eric Phipps
Lee Ann Riesen
Damian Rouson

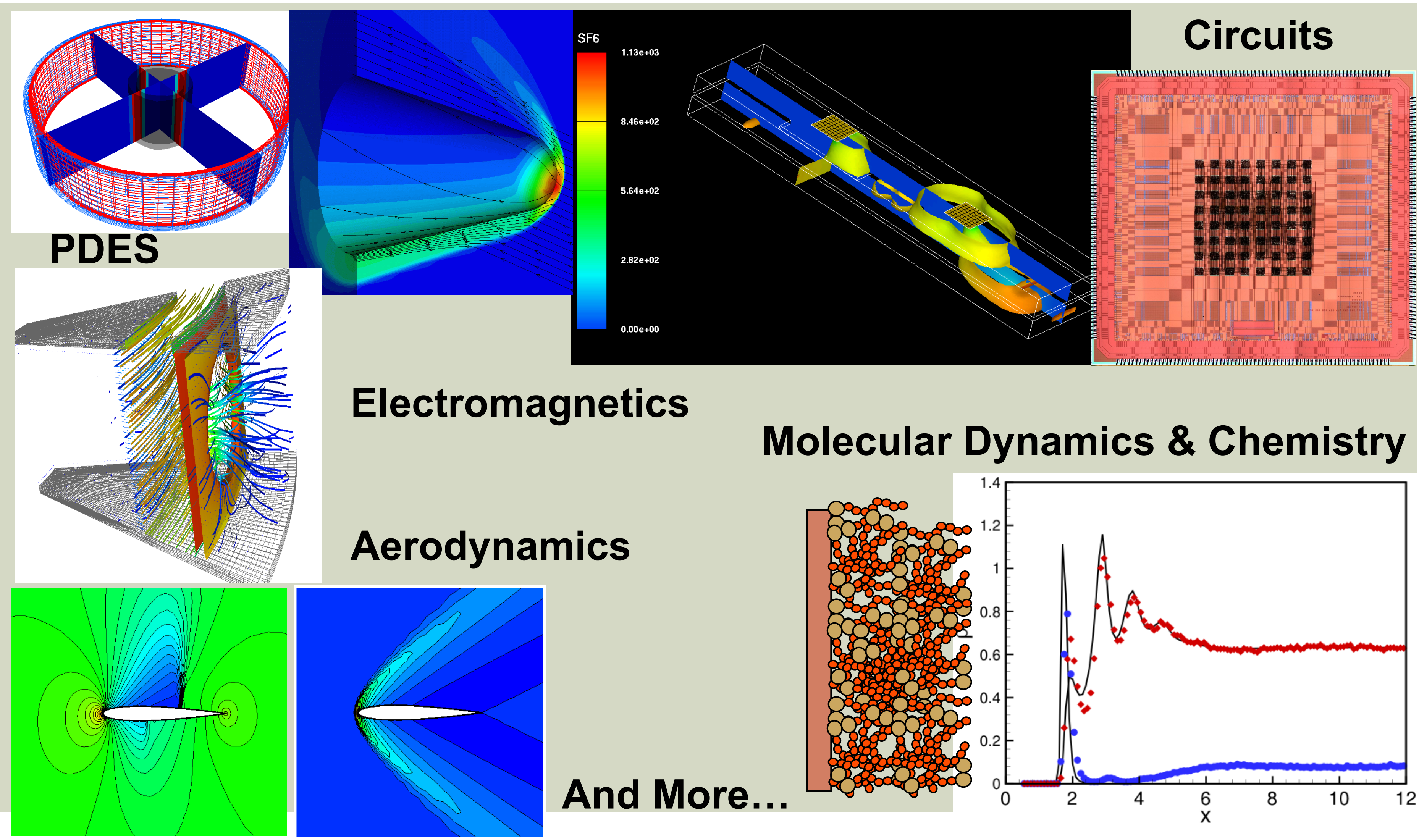
Marzio Sala
Andrew Salinger
Chris Siefert
Bill Spotz
Heidi Thornquist
Ray Tuminaro
Jim Willenbring
Alan Williams

Past Contributors

Jason Cross
Michael Gee
Esteban Guillen
Bob Heaphy
Kris Kampshoff
Ian Karlin
Sarah Knepper
Tammy Kolda
Joe Outzen
Mike Phenow
Paul Sexton
Bob Shuttleworth
Ken Stanley

Sandia National Laboratories

TARGET PROBLEMS



TARGET PLATFORMS

Any and all.

- Desktop: Development and more…
- Leadership-class machines:
  - Redstorm (XT3), JaguarPF (XT5), Clusters
  - Roadrunner (Cell-based).
  - Multicore nodes.
- Parallel software environments:
  - MPI.
  - Threads, vectors, CUDA OpenCL, …
  - Combinations of the above.
- User “skins”:
  - C++
  - Fortran
  - C
  - Python
  - Web, CCA.

Sandia National Laboratories

BEYOND SOLVERS

physics

$L(u)=f$
Math. model

$L_h(u_h)=f_h$
Numerical model

$u_h=L_h^{-1}\cdot f_h$
Algorithms

computation

- Natural expansion of capabilities to satisfy application and research needs

Numerical math
Convert to models that can be solved on digital computers

Algorithms
Find faster and more efficient ways to solve numerical models

Trilinos

discretizations
- Time domain
- Space domain

methods
- Automatic diff.
- Domain dec.
- Mortar methods

solvers
- Linear
- Nonlinear
- Eigenvalues
- Optimization

core
- Petra
- Utilities
- Interfaces
- Load Balancing

- Discretization methods, AD, Mortar methods, …

Sandia National Laboratories

SOME TRILINOS PACKAGES

(http://trilinos.sandia.gov)

	Objective	Package(s)
Discretizations	Meshing & Spatial Discretizations	phdMesh, Intrepid, Pamgen, Sundance, ITAPS
	Time Integration	Rythmos
Methods	Automatic Differentiation	Sacado
	Mortar Methods	Moertel
Services	Linear algebra objects	Epetra, Jpetra, Tpetra, Kokkos
	Interfaces	Thyra, Stratimikos, RTOp, FEI, Shards
	Load Balancing	Zoltan, Isorropia
	"Skins"	PyTrilinos, WebTrilinos, ForTrilinos, Ctrilinos, Optika
	C++ utilities, I/O, thread API	Teuchos, EpetraExt, Kokkos, Triutils, ThreadPool, Phalanx
Solvers	Iterative (Krylov) linear solvers	AztecOO, Belos, Komplex
	Direct sparse linear solvers	Amesos
	Direct dense linear solvers	Epetra, Teuchos, Pliris
	Iterative eigenvalue solvers	Anasazi, Rbgen
	ILU-type preconditioners	AztecOO, IFPACK, Tifpack
	Multilevel preconditioners	ML, CLAPS
	Block preconditioners	Meros
	Nonlinear system solvers	NOX, LOCA
	Optimization (SAND)	MOOCHO, Aristos, TriKota, Globipack, Optipack
	Stochastic PDEs	Stokhos

Sandia National Laboratories

SBIR OPPORTUNITIES

- Parallel algorithms (Trilinos):

  Sandia ←→ ASAP, LLC

  ASAP is contributing code to ForTrilinos in order to access dense linear solvers in Pliris for use in a Boundary Element Method electromagnetics solver for AFOSR.

- Modern Fortran (ForTrilinos):

  Numerica 21, Inc.← → Sandia

  Fortran test harness

  NAG ←→ Sandia

  Standard conformance check for ForTrilinos

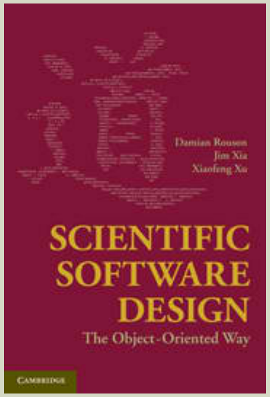
  Pre-release testing for the NAG Fortran compiler

  XML documentation for ForTrilinos

- User training (Morfeus):

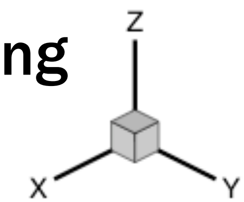
  Sandia ←→ Numerical Algorithms Group (NAG)

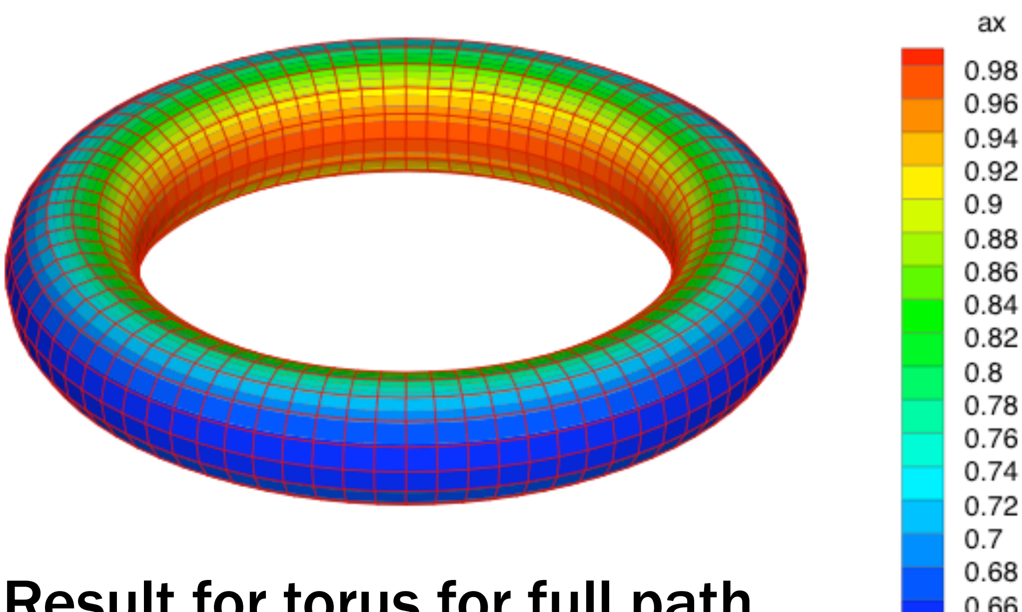
  Course development for UK's HECToR supercomputer center



ASAP, LLC
(AFOSR SBIR PHASE I/II)

3D Magnetic Field Modeling via Boundary Element Methods





Result for torus for full path integration

Wave-diffusion equation for magnetic field vector potential:

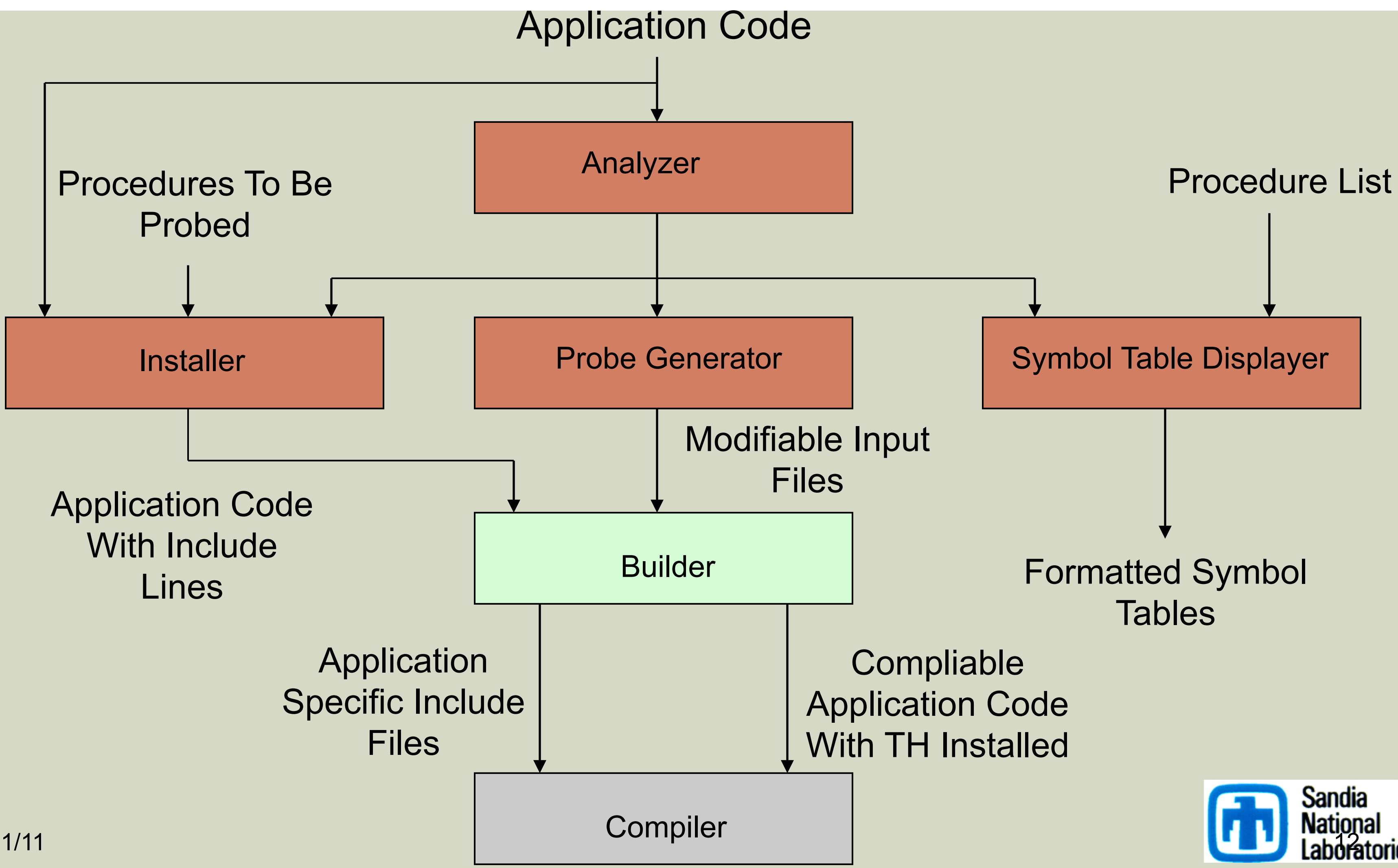
$$\nabla^2 \vec{A} = \mu\sigma \frac{\partial \vec{A}}{\partial t} + \mu\varepsilon \frac{\partial^2 \vec{A}}{\partial t^2}$$

Leveraging ForTrilinos:

- Support for parellel and serial OOP.
- Protects investments in Fortran and C++ implementations.

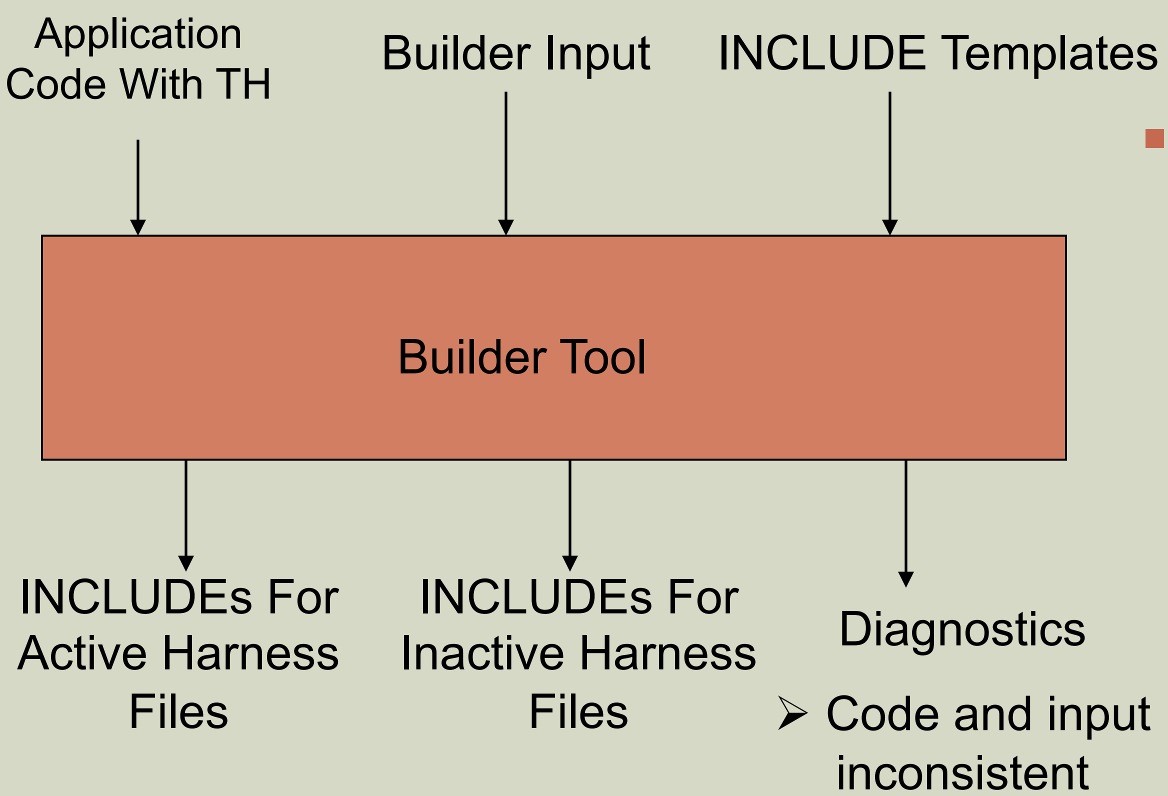
Sandia National Laboratories

NUMERICA 21 TEST HARNESS



NUMERICA 21 USER INPUT

Building an application with the test harness installed:



- Builder Input (user-supplied)
  - Variables To Monitor
    - Subprogram
    - Name
    - Declaration
    - Identity/closeness check strategy
    - Frequency
    - Debug Levels

NUMERICAL ALGORITHMS GROUP

HECToR Training Courses

Date	Course	Location
4/6/11	Transitioning to the Cray XE6	NAG Manchester
4/7/11	Co-Array Fortran	NAG Manchester
4/13/11	Transitioning to the Cray XE6	NAG Oxford
4/14/11	Co-Array Fortran	NAG Oxford
5/10-11/11	OpenMP (FULL)	Univ. College London
5/23-24/11	Intro. to CUDA Programming (FULL)	NAG Oxford
5/25-26/11	Intro. to OpenCL Programming	NAG Oxford
5/25-26/11	OpenMP	University of Bath
6/14-16/11	Parallel Programming with MPI (FULL)	Univ. College London
?	OOP in Fortran 2003/2008	?

ForTrilinos APPLICATION PROTOTYPE

Blackboard abstractions (Burgers equation):

$$u_t = \nu u_{xx} - u u_x$$

$u = u(x,t)$: velocity field

ν: diffusion coefficient

Software abstractions:

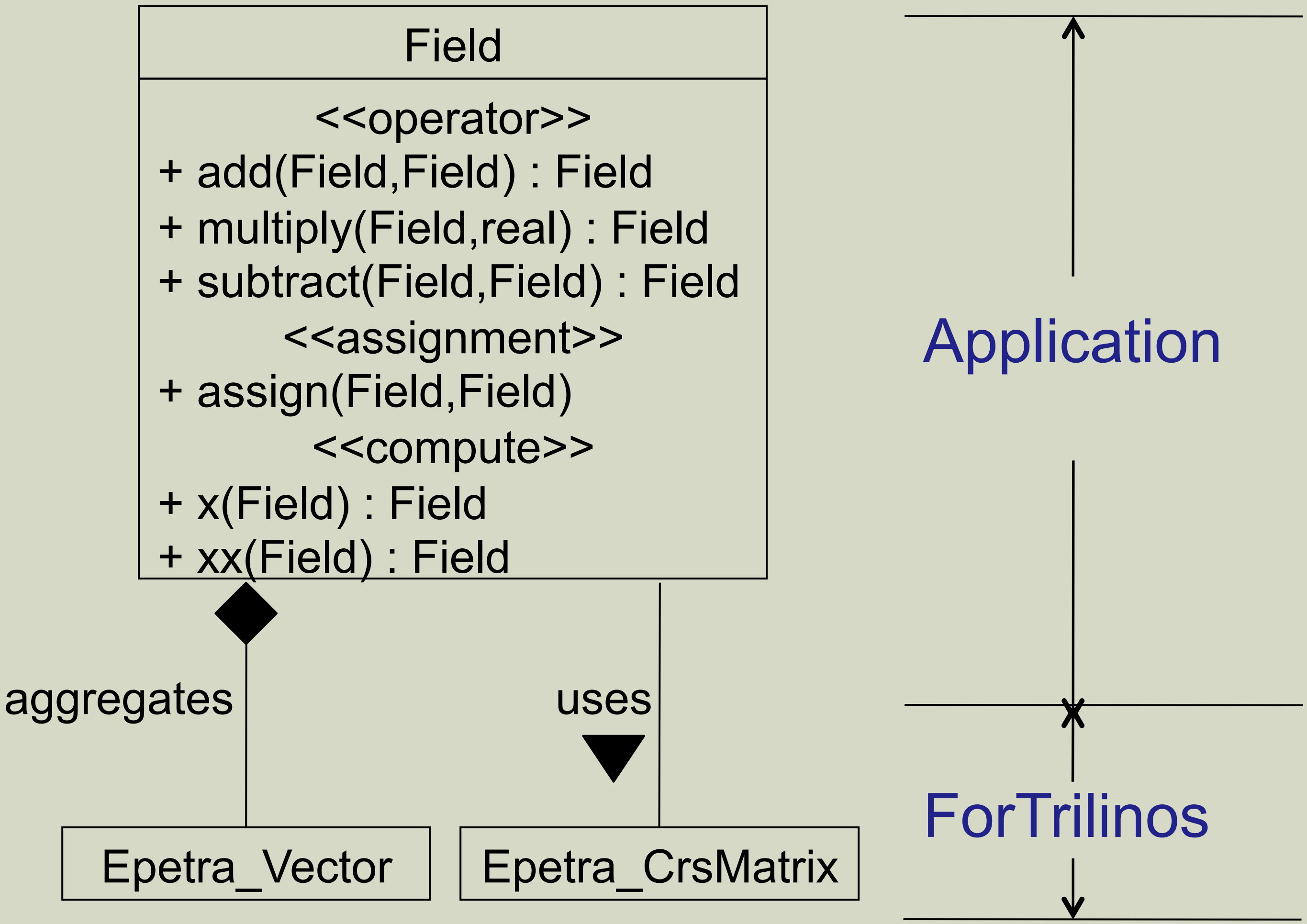
```
type(field) :: u=field(), u_t
u_t =  nu*u%xx() - u*u%x()
```

Synchronization

Asynchronous, purely functional operators and methods.

Sandia National Laboratories

ARCHITECTURE



IMPLEMENTATION

```
program main
!-------------- Dependencies -------------------------
#include "ForTrilinos_config.h”
#ifdef HAVE_MPI
  use mpi
  use FEpetra_MpiComm,    only : Epetra_MpiComm
#else
  use FEpetra_SerialComm, only : Epetra_SerialComm
#endif
  use ForTrilinos_utils,  only : valid_kind_parameters
  use iso_c_binding,      only : c_int,c_double
  use field_module,       only : field,initial_field
  use initializer,        only : u_initial
  implicit none
```

IMPLEMENTATION (CONT.)

```
!---------------- Declarations ----------------------
#ifdef HAVE_MPI
  type(Epetra_MpiComm)     :: comm
#else
  type(Epetra_SerialComm) :: comm
#endif
  type(field)              :: u,u_t
  procedure(initial_field), pointer :: initial
!---------------- MPI Start-up ----------------------
#ifdef HAVE_MPI
  call MPI_INIT(ierr)
  comm = Epetra_MpiComm(MPI_COMM_WORLD)
#else
  comm = Epetra_SerialComm()
#endif
```

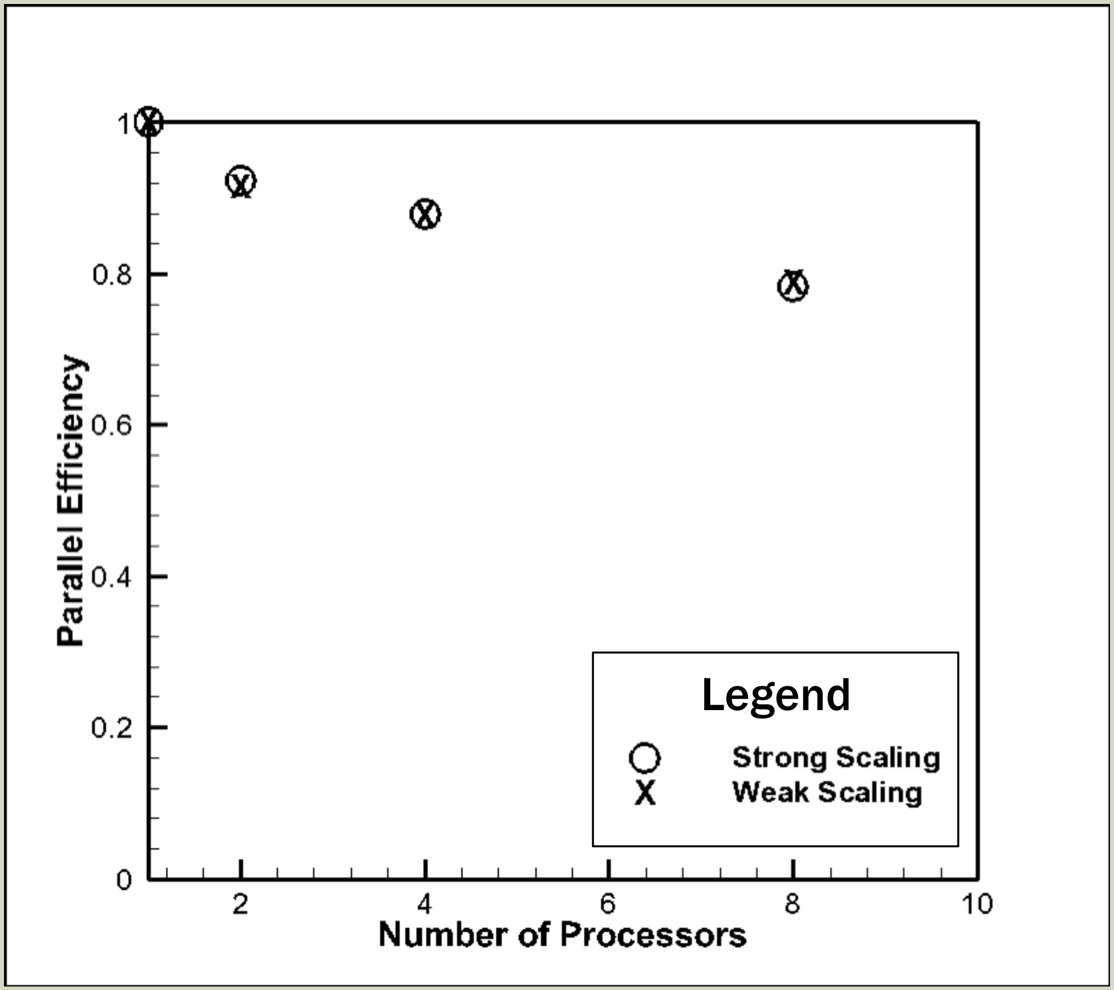
IMPLEMENTATION (CONT.)

```
!---------------- Object initialization ----------------
  initial => u_initial
  u = field(initial,grid_resolution,comm)
!------------- Forward Euler time integration ----------
  do tstep=1,1000
   dt = u%euler_step(nu ,grid_resolution)
   u_t = u%xx()*nu – u*u%x()
   u  = u + u_t*dt
   t = t + dt
  end do
```

IMPLEMENTATION (CONT.)

```
!-------------------- Memory clean-up ------------------
  call u%force_finalize
  call u_t%force_finalize
  call comm%force_finalize
!-------------------- MPI shutdown ---------------------
#ifdef HAVE_MPI
  call MPI_FINALIZE(rc)
#endif
end program
```

PERFORMANCE



Sandia National Laboratories

MORFEUS APPLICATION PROTOTYPE

Burgers Equation: $$u_t + uu_x = \nu u_{xx}$$

$u = u(x,t)$: velocity field

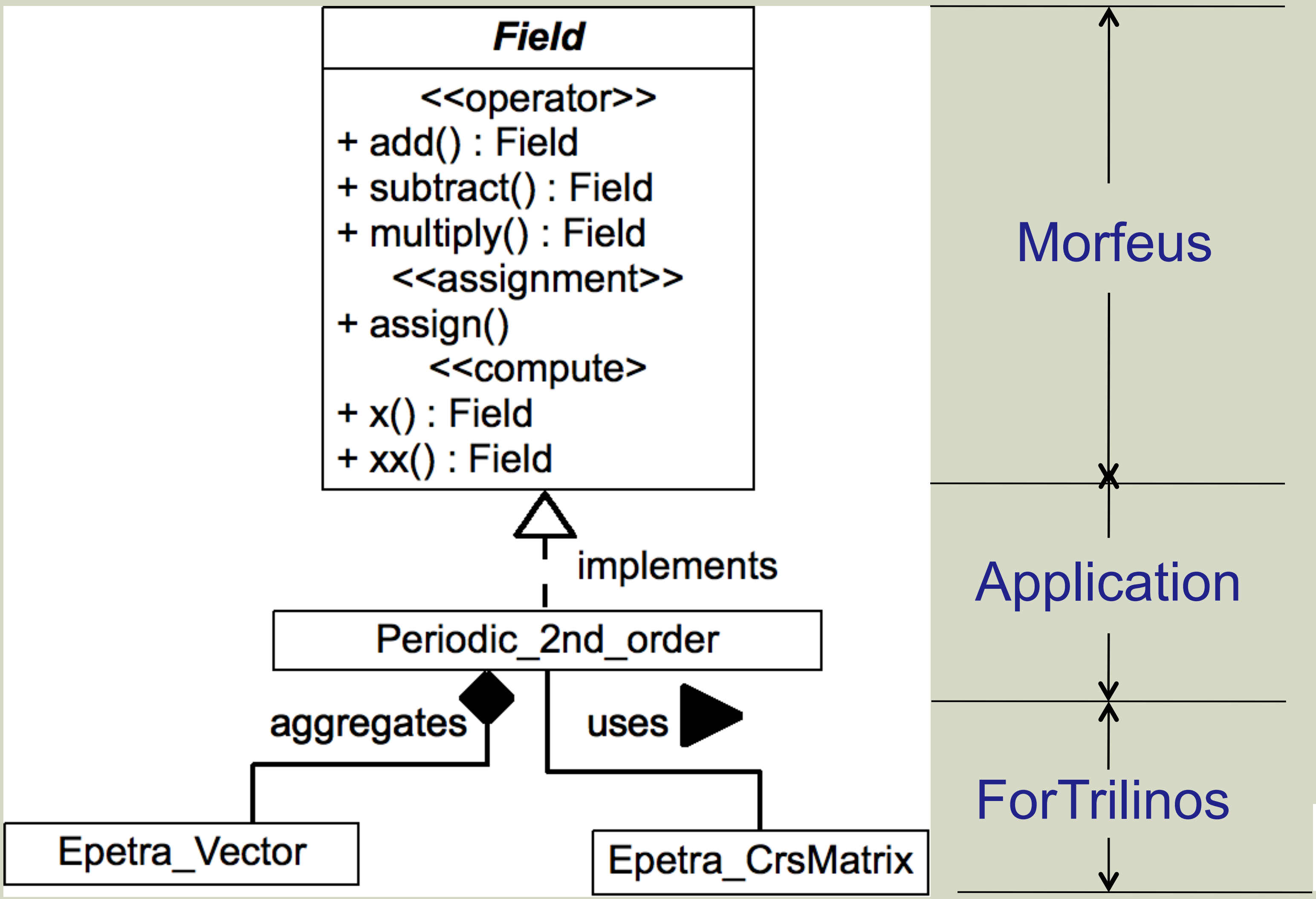
ν : diffusion coefficient

Abstract Calculus Pattern:

```
class(field),pointer :: u,u_t
u_t =  nu*u%xx() - u*u%x()
```

"Design to an interface, not an implementation."
Gamma et al.

ARCHITECTURE



CONTACT INFORMATION

- Brian T. Smith, N21 Inc.
    - 22 Crystal Mountain Rd
    - Angel Fire, NM 87710-1668
    - Email: carbess@swcp.com
- Ken Kubat, NAG
  - E-mail: KKubat@nag.com
- Robert Meyer, NAG
    - E-mail: RMeyer@nag.com