ForTrilinos: Bringing Trilinos to Object-Oriented Fortran Parallel Applications

Karla Morris

Sandia National Laboratories

European Trilinos User Group Meeting

EPFL Lausanne, Switzerland

June 5, 2012

SAND 2012-4317C

Sponsor: DOE

Sandia National Laboratories

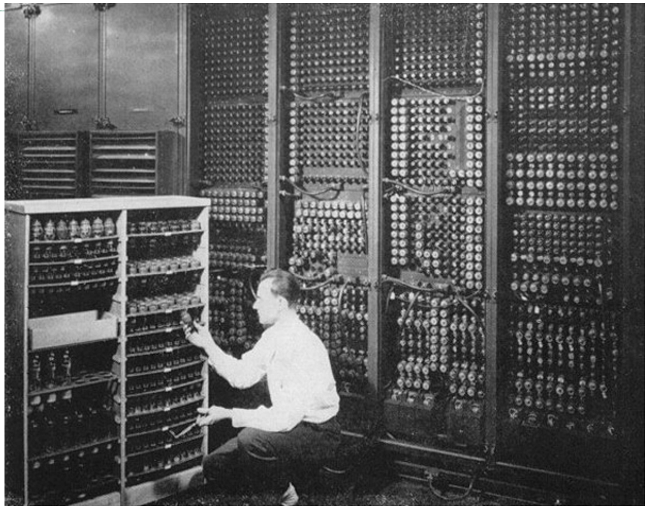
Contributors

- Damian Rouson, Sandia National Laboratories
- Nicole Lemaster, Sandia National Laboratories
- Salvatore Filippone, Università di Roma “Tor Vergata”
- Xioafeng Xu, General Motors
- Jim Xia, IBM
- Brian Smith, Accurate Solutions In Applied Physics

Outline

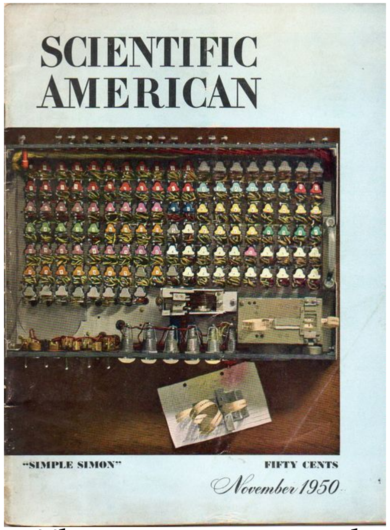
- Introduction
  - Motivation
  - Objectives
- Methodology
  - Interoperability and Portability
  - Software Stack
  - Framework
- Results
  - Code Examples
- Closing remarks & Future Work
  - Support and Documentation

Fortran's Image



http://www.computersciencelab.com/ComputerHistory/HistoryPt4.htm

http://www.computersciencelab.com/ComputerHistory/HistoryPt4.htm



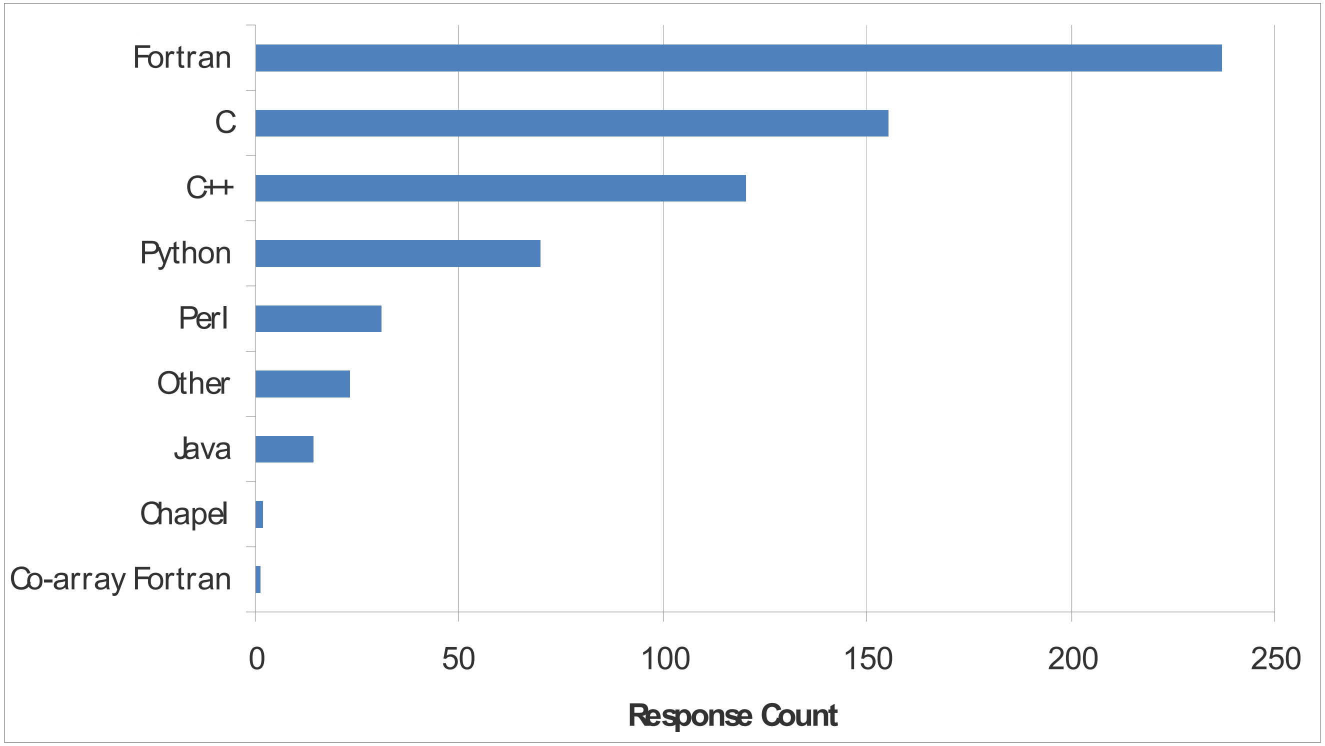
http://longstreet.typepad.com/thesciencebookstore/computer_techhistory/

http://www.clemson.edu/caah/history/facultypages/PamMack/lec122sts/computers.html

Fortran's Reality:
PRACE HPC User Survey

Question 31: Which programming models and languages do you use for code development? Please select one or more from the following list.

Response rate: 78%



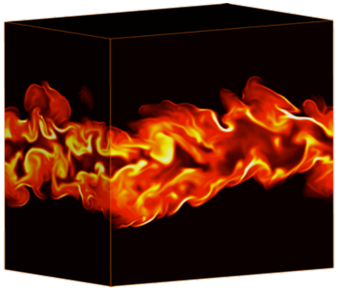
Source: Bull, M., Guo, X., Ioannis Liabotis, I. (Feb. 2011) *Applications and user requirements for Tier-0 systems*, PRACE Consortium Partners (http://www.tinyurl.com/PRACE-survey-2008).

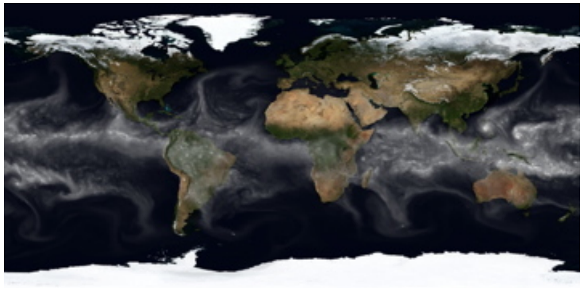
Why Write Fortran Drivers for C++?

- The applied math and computer science infrastructure that aims to support computational science is increasingly written in C++:
  - Trilinos: 92 MB gzipped tar ball of open-source, object-oriented, parallel C++ solvers and services.

http://trilinos.sandia.gov

- Several research communities actively develop important applications in Fortran, e.g., in energy (combustion) and climate (models for the atmosphere, ice sheets, and oceans):





Trilinos Package Summary

	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

Customized Procedural Interface

Advantages

- Gets the job done
- Does not required extensive lines of additional code

Disadvantages

- Requires flattening the data (reduced to intrinsic types & 1D arrays thereof)
- Requires flattening the functions that act on the data (no inheritance)
- Requires hardwiring assumptions into receiving code
- Typically involves passing raw data (no encapsulation on the Fortran side)
- Not extensible (the work involved in wrapping one package cannot be leveraged in wrapping others).

ForTrilinos Objectives

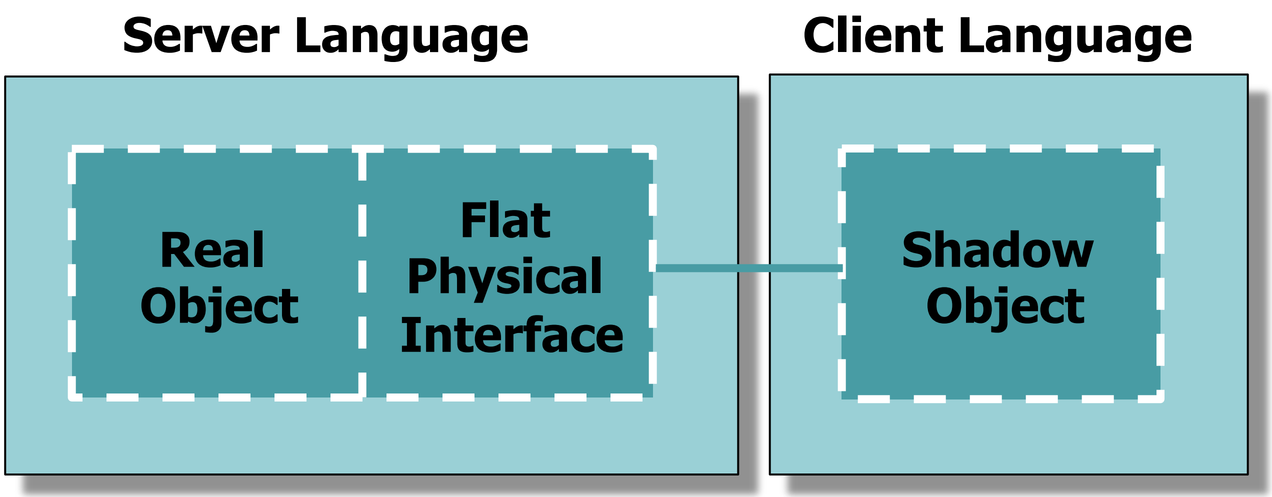
- To expand the use of Trilinos into communities that predominantly write Fortran.
- To maintain the object-oriented design philosophy of Trilinos while providing interfaces that feel natural to Fortran programmers.
- To develop and demonstrate new idioms for writing object-oriented Fortran.

Outline

- Introduction
  - Motivation
  - Objectives
- Methodology
  - Interoperability and Portability
  - Software Stack
  - Framework
- Results
  - Code Examples
- Closing remarks & Future Work
  - Support and Documentation

Manual Interoperability: C++/Fortran 95

- Gray et al. (1999) "Shadow-object interface between Fortran 95 and C++," *Computing in Science & Engineering* **11**:2, pp. 64—70:
  - Interface OO C++ and object based Fortran 95
  - Flat interface exports real object behavior
  - Shadow object is a logical interface, can be treated as a native object



A flat interface exported to a shadow object in language B by a real object in language A.

ForTrilinos and CTrilinos

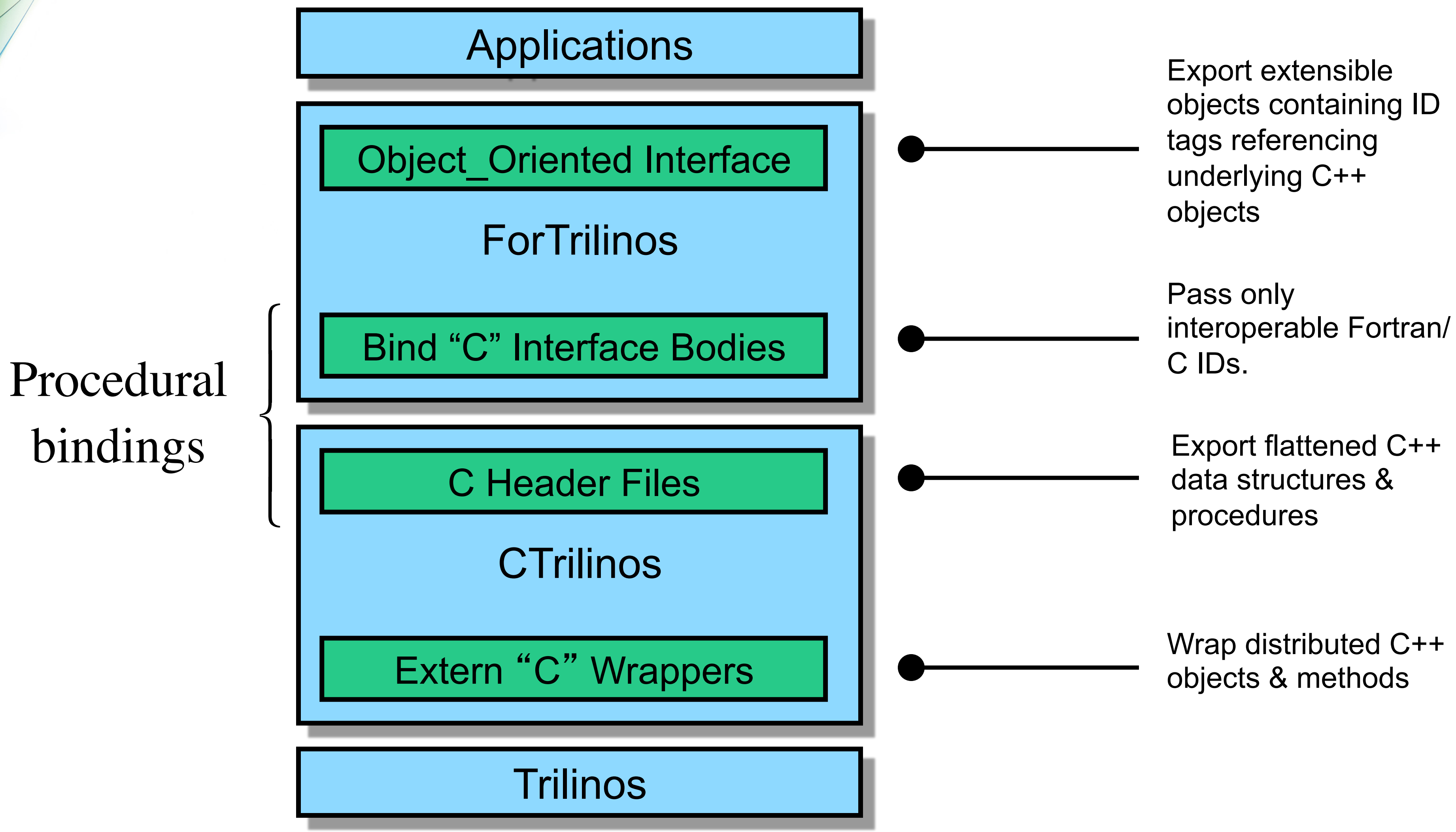
- ForTrilinos

  Open-source software package that provides Fortran interfaces to C++ packages published by the Trilinos project (http://trilinos.sandia.gov)

- CTrilinos (exists only as a service to ForTrilinos → no user interface)

  Open-source software package that exports C interfaces to Trilinos packages and serves as an intermediary between ForTrilinos and other Trilinos packages.

Software Stack



C interoperability in Fortran 2003: Compatible Types/Kinds

Fortran types	Fortran 2003 kind parameter	C type
integer	c_int	int
integer	c_long	long
real	c_double	double
real	c_float	float
character	c_char	char

Fortran 2003 kind parameters and the corresponding interoperable C types

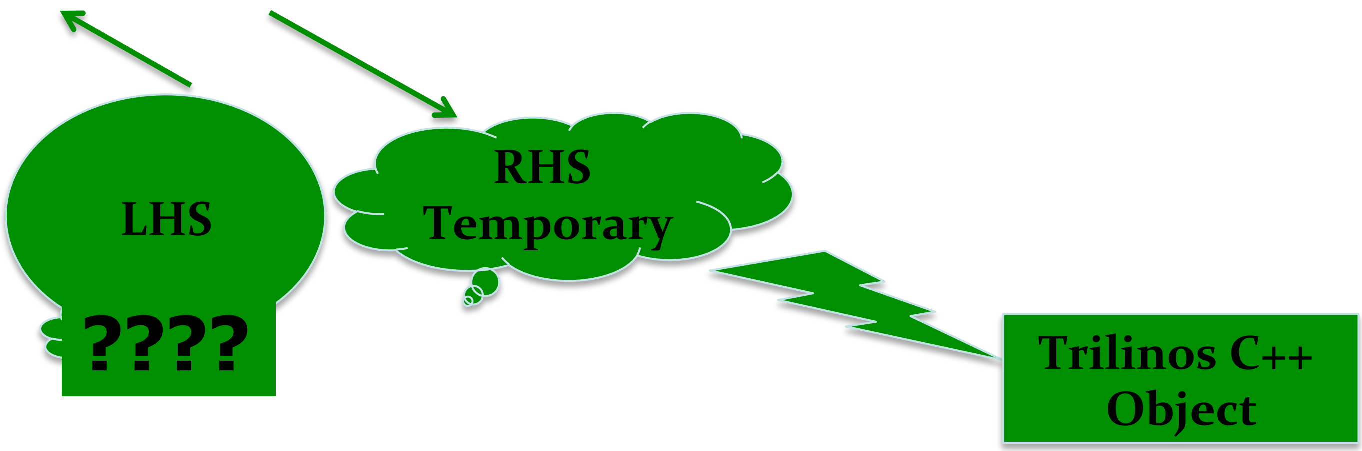
Sample Main Program

```
program main
  use iso_c_binding, only : c_double
  use Fepetra_MpiComm, only : Epetra_MpiComm
  use FEpetra_Map, only : Epetra_Map
  use FEpetra_Vector, only : Epetra_Vector
  use mpi
  implicit none
  type(Epetra_MpiComm) :: comm
  type(Epetra_Map) :: map
  type(Epetra_Vector) :: b
  integer :: ierr
  call mpi_init(ierr) ! MPI startup
  comm = Epetra_MpiComm(MPI_COMM_WORLD)
  map = Epetra_Map(numGlobalElements=64,IndexBase=1,comm=comm)
  b = Epetra_Vector(map)
  call b%PutScalar(2.0_c_double)
  print *, "L2 norm of b = ",b%Norm2()
  call mpi_finalize(ierr) ! MPI shutdown
end program
```

ForTrilinos

Object construction/destruction in Fortran 2003:

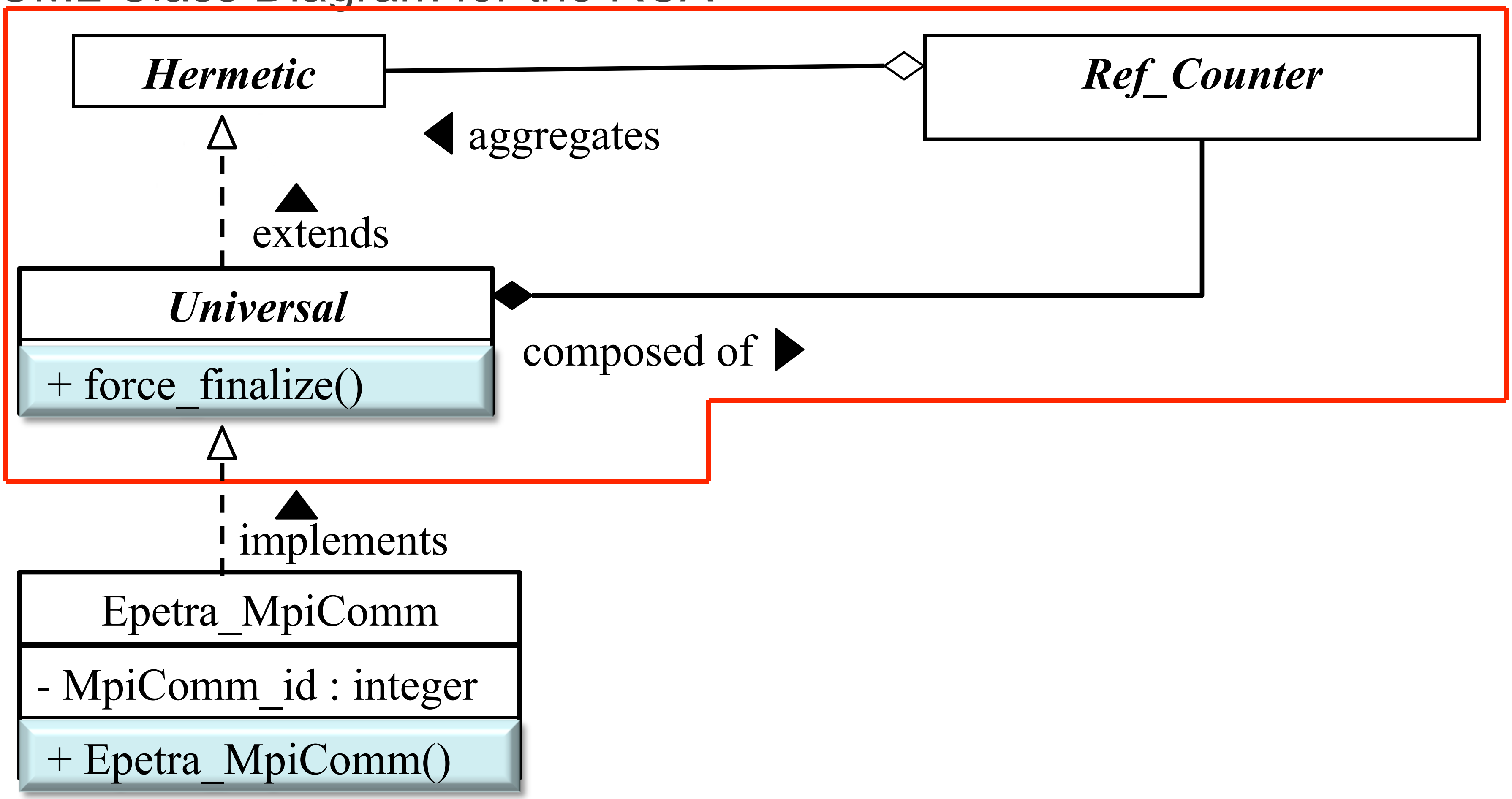
```
program main
 use mpi
 use FEpetra_MpiComm ,only : Epetra_MpiComm
 implicit none
 type(Epetra_MpiComm) :: comm
 integer ierr
 call mpi_init(ierr)
  comm = Epetra_MpiComm(mpi_comm_world)
```



Rouson, D. W. I., Morris, K., and Xia, J., “Managing C++ objects with modern Fortran in the driver's seat: This is not your father's Fortran,” Computing in Science and Engineering

Reference Counting Architecture (RCA): Class Structure

UML Class Diagram for the RCA



Morris, K., Rouson, D. W. I., and Xia, J., "On the object-oriented design of reference-counted shadow objects in Fortran 2003," Fourth International Workshop on SECSE, 2011

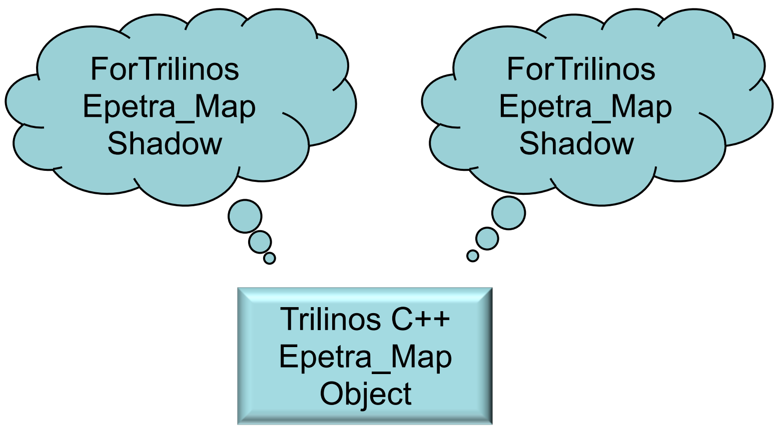
ForTrilinos Objects: Construction/Destruction

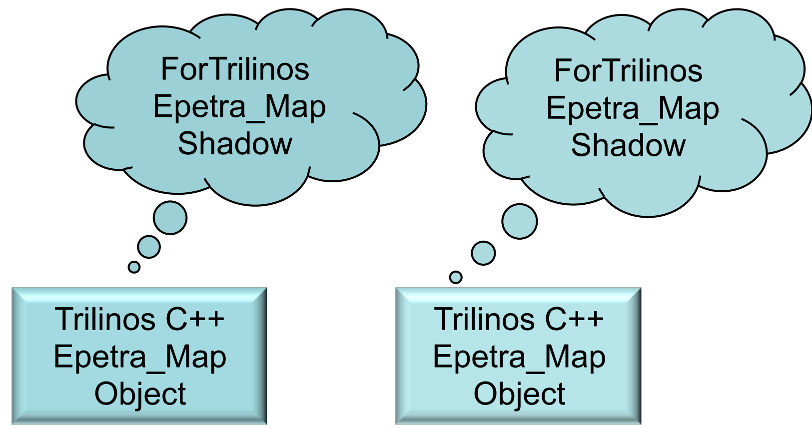
```
program main
 use mpi
 use FEpetra_MpiComm ,only : Epetra_MpiComm
 use FEpetra_Map ,only : Epetra_Map
 type(Epetra_MpiComm) :: comm
 type(Epetra_Map)     :: map,map_copy
 integer :: ierr
 call mpi_init(ierr)     ! MPI startup
 ! Object construction
 comm = Epetra_MpiComm(mpi_comm_world)
 map=Epetra_Map(Num_GlobalElements=512_c_int,IndexBase=1_c_int,comm=comm)
 map_copy = Epetra_Map(map)
 ! Object destruction
 call map_copy%force_finalize ; call map%force_finalize
 call comm%force_finalize
 call mpi_finalize(ierr) ! MPI finalize
end
```

ForTrilinos Objects: Construction/Destruction

```
type(Epetra_Map):: map,map_copy
map = Epetra_Map(64, 1, comm)
map_copy = map
```

```
type(Epetra_Map):: map,map_copy
map = Epetra_Map(64, 1, comm)
map_copy = Epetra_Map(map)
```





Outline

- Introduction
  - Motivation
  - Objectives
- Methodology
  - Interoperability and Portability
  - Software Stack
  - Framework
- Results
  - Code Examples
- Closing remarks & Future Work
  - Support and Documentation

Matrix-Vector Multiplication
2nd order Finite Difference Scheme

$$Af = f''$$

$$\frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{bmatrix}\begin{bmatrix} f_1 \\ f_2 \\ f_i \\ f_{n-1} \\ f_n \end{bmatrix} = \begin{bmatrix} f_1'' \\ f_2'' \\ f_i'' \\ f_{n-1}'' \\ f_n'' \end{bmatrix}$$

$$f(x) = \sin(x) \quad f''(x) = -f(x)$$

Matrix Vector Multiplication: Source Code

```
program main
#include "ForTrilinos_config.h"  ! Configuration Flags
  ! Module dependencies
#ifdef HAVE_MPI
  use mpi
  use FEpetra_MpiComm ,only : Epetra_MpiComm
#else
  use FEpetra_SerialComm ,only : Epetra_SerialComm
#endif
  use FEpetra_Map ,only : Epetra_Map
  use FEpetra_Vector ,only : Epetra_Vector
  use FEpetra_CrsMatrix ,only : Epetra_CrsMatrix
  use ForTrilinos_enum_wrappers ,only: FT_Epetra_DataAccess_E_Copy
  use kind_parameters ,only : rkind,ikind
  use math_constants ,only : pi
! Continues next slide
```

Matrix Vector Multiplication: Source Code

```
 implicit none
  ! Local variables
#ifdef HAVE_MPI
  type(Epetra_MpiComm) :: comm
#else
  type(Epetra_SerialComm) :: comm
#endif
  type(Epetra_Map)    :: map
  type(Epetra_Vector) :: f, f_pp
  type(Epetra_CrsMatrix) :: A
  integer(ikind) :: NumMyElements,i,ierr,indices(3)
  real(rkind) :: values(3),dx
  real(rkind) ,dimension(:) ,allocatable :: x_node
  integer(ikind) ,dimension(:) ,allocatable :: NumNz
  integer(ikind) ,dimension(:) ,allocatable :: MyGlobalElements
! Continues next slide
```

Matrix Vector Multiplication: Source Code

```
! Local parameters
  real(rkind)      ,parameter :: tolerance=1.0E-4
  integer(ikind) ,parameter :: Index_Base=1_ikind
  logical          ,parameter :: zero_initial=.true.
  ! Total number of grid points
  integer(ikind) ,parameter :: NumGlobalElements=256_ikind

 ! Create a comm to handle communication
#ifdef HAVE_MPI
  call MPI_INIT(ierr)
  comm = Epetra_MpiComm(MPI_COMM_WORLD)
#else
  comm = Epetra_SerialComm()
#endif
! Continues next slide
```

Matrix Vector Multiplication: Source Code

```
 ! Create a map
 map = Epetra_Map(NumGlobalElements,Index_Base,comm)
 ! Get update list and number of local equations from given Map
 MyGlobalElements = map%MyGlobalElements()
 NumMyElements = map%NumMyElements()

! Create integer vector NumNz.
! NumNz(i) is number of non-zero elements for the ith global equation
! on this processor
 allocate(NumNz(NumMyElements))

! Tridiagonal matrix where each row has (1/dx2 -2/dx2  1/dx2)
! Interior grid points have 2 off-diagonal terms
 NumNz = 3
! Continues next slide
```

Matrix Vector Multiplication: Source Code

```
! Create a Epetra_Matrix
  A = Epetra_CrsMatrix(FT_Epetra_DataAccess_E_Copy,map,NumNz)

! Grid resolution for a domain of length 2*pi
  dx=2.*pi/real(NumGlobalElements,rkind)

! Add rows one at a time
! Need some vectors to help
! off diagonal values will always be 1/dx2 and 1/dx2
  values(1) = 1.0/(dx*dx)
  values(2) =-2.0/(dx*dx)
  values(3) = 1.0/(dx*dx)
! Continues next slide
```

Matrix Vector Multiplication: Source Code

```
do i=1,NumMyElements
  if (MyGlobalElements(i)==1) then
    indices(1) = NumGlobalElements; indices(2) = 1; indices(3) = 2
  else if(MyGlobalElements(i)==NumGlobalElements) then
    indices(1) = NumGlobalElements-1; indices(2) = NumGlobalElements
    indices(3) = 1
  else
    indices(1) = MyGlobalElements(i)-1
    indices(2) = MyGlobalElements(i)
    indices(3) = MyGlobalElements(i)+1
  end if
  call A%InsertGlobalValues(MyGlobalElements(i),values,indices)
end do
  !Finish up
  call A%FillComplete(.true.)
 ! Continues next slide
```

Matrix Vector Multiplication: Source Code

```fortran
! Create vectors
  call f%Epetra_Vector_(A%RowMap(), zero_initial)
  call f%Epetra_Vector_(f_pp)

 ! Insert values f(i)=sin(2*(i-1)*pi/N) in to vector x
 allocate(x_node(NumMyElements))
 do i=1,NumMyElements
   x_node(i) = sin(dx*(real(MyGlobalElements(i),rkind)-1))
 enddo
 call f%ReplaceGlobalValues( &
              values=x_node,indices=MyGlobalElements)
 ! Matrix Vector Multiplication
 call A%Multiply_Vector(.false.,f,f_pp)

! Continues next slide
```

Matrix Vector Multiplication: Source Code

```
! Clean up before exiting main
  call f_pp%force_finalize()
  call f%force_finalize()
  call A%force_finalize()
  call map%force_finalize()
  call comm%force_finalize()

 ! Finalize mpi
#ifdef HAVE_MPI
  call MPI_FINALIZE(ierr)
#endif

end
```

Linear System Solve
Higher Order Finite Difference Scheme

$$Af'' = Bf$$

$$\beta f''_{i-2} + \alpha f''_{i-1} + f''_i + \alpha f''_{i+1} + \beta f''_{i+2} =$$

$$c\frac{f_{i+3} - 2f_i + f_{i-3}}{9h^2} + b\frac{f_{i+2} - 2f_i + f_{i-2}}{4h^2} +$$

$$a\frac{f_{i+1} - 2f_i + f_{i-1}}{h^2}$$

$$f(x) = \sin(x)$$

$$f''(x) = -f(x)$$

Lele, S. K. (1992). Compact finite difference schemes with spectral-like resolution. J. Comp. Phys. **103**,16-42.

Linear System Solve: Source Code

```
program main
  ! Configuration Flags
#include "ForTrilinos_config.h"
  ! Module dependencies
#ifdef HAVE_MPI
  use mpi
  use FEpetra_MpiComm ,only : Epetra_MpiComm
#else
  use FEpetra_SerialComm ,only : Epetra_SerialComm
#endif
  use FEpetra_Map ,only : Epetra_Map
  use FEpetra_Vector ,only : Epetra_Vector
  use FEpetra_CrsMatrix ,only : Epetra_CrsMatrix
  use FAztecOO ,only : AztecOO
  use kind_parameters ,only : rkind,ikind
  use math_constants ,only : pi
  implicit none
```

Linear System Solve: Source Code

```
  ! Local variables
#ifdef HAVE_MPI
  type(Epetra_MpiComm) :: comm
#else
  type(Epetra_SerialComm) :: comm
#endif
  type(Epetra_Map)     :: map
  type(Epetra_Vector) :: f, f_pp
  type(Epetra_CrsMatrix) :: A
  type(AztecOO) :: Solver
  integer(ikind) :: NumMyElements,i,ierr
  real(rkind) :: dx
  real(rkind) ,dimension(:) ,allocatable :: x_node
  integer(ikind) ,dimension(:) ,allocatable :: MyGlobalElements
```

Linear System Solve: Source Code

```
! Local parameters
  real(rkind)    ,parameter :: tolerance=1.0E-10  integer
   (ikind) ,parameter :: Index_Base=1_ikind,  & MaximumIter=100_
  logical        ,parameter :: zero_initial=.true.
  ! Total number of grid points
  integer(ikind) ,parameter :: NumGlobalElements=256_ikind
  ! Coefficients according to Lele J. Comp. Phys. 103 (1992)
  ! lhs_coef=(/beta, alpha, 1, alpha, beta/)
  ! rhs_coef=(/c/9, b/4, a, (-2c/9-2b/4-2a), a, b/4, c/9/)
  ! Classic Pade finite difference scheme
  real(rkind) ,dimension(5) ,parameter :: &
    lhs_coef=(/ 0.0, 1.0/10.0, 1.0, 1.0/10.0, 0.0 /)
  real(rkind) ,dimension(5) ,parameter :: &
    rhs_coef=(/ 0.0, 6.0/5.0, -12.0/5.0, 6.0/5.0, 0.0 /)
```

Linear System Solve: Source Code

```
 ! Create a comm to handle communication
#ifdef HAVE_MPI
  call MPI_INIT(ierr)
  comm = Epetra_MpiComm(MPI_COMM_WORLD)
#else
  comm = Epetra_SerialComm()
#endif
  ! Create a map
  map = Epetra_Map(NumGlobalElements,Index_Base,comm)
  ! Get update list and number of local equations from given Map
  MyGlobalElements = map%MyGlobalElements()
  NumMyElements = map%NumMyElements()

  ! Grid resolution for a domain of length 2*pi
  dx=2.*pi/real(NumGlobalElements,rkind)
```

Linear System Solve: Source Code

```
! Create RHS Matrix
call SparseMatrix(A,map,rhs_coef)
call A%Scale(real((1./(dx*dx)),rkind))

! Create vectors
f=Epetra_Vector(A%RowMap(), zero_initial)
f_pp = Epetra_Vector(f)

! Insert values f(i)=sin(2*(i-1)*pi/N) into vector f
allocate(x_node(NumMyElements))
do i=1,NumMyElements
  x_node(i) = sin(dx*(real(MyGlobalElements(i),rkind)-1))
enddo
call f%ReplaceGlobalValues( &
 values=x_node,indices=MyGlobalElements)
```

Linear System Solve: Source Code

```
! Creates RHS by Matrix Vector Multiplication
call A%Multiply_Vector(.false.,f,f)

! Create LHS Matrix
call SparseMatrix(A,map,lhs_coef)

! Linear System Solve for f_pp
f_pp = Epetra_Vector(f)
Solver = AztecOO(A,f_pp,f)
call Solver%iterate(A,f_pp,f,MaximumIter,tolerance)
```

Linear System Solve: Source Code

```
  ! Clean up before exiting main
  call f%force_finalize()
  call f_pp%force_finalize()
  call A%force_finalize()
  call Solver%force_finalize()
  call map%force_finalize()
  call comm%force_finalize()

#ifdef HAVE_MPI
  call MPI_FINALIZE(ierr)
#endif

end
```

Outline

- Introduction
  - Motivation
  - Objectives
- Methodology
  - Interoperability and Portability
  - Software Stack
  - Framework
- Results
  - Code Examples
- Closing remarks & Future Work
  - Support and Documentation

Compiler Support for ForTrilinos

- Successful, complete builds (no test failures):
  - IBM XL Fortran/C++
  - Numerical Algorithms Group (NAG) Fortran atop GCC (g++) 4.2
- Complete builds (with test failures):
  - Cray 8.0.5.106
- Complete builds (with 2 workarounds):
  - GCC 4.7 (missing deferred-length characters and final subroutines).
- Nominal support for all Fortran 2003 constructs in ForTrilinos:
  - Portland Group
  - Intel

Documentation and Software Downloads

- Trilinos download
  - http://trilinos.sandia.gov/
- ForTrilinos doxygen documentation
  - http://trilinos.sandia.gov/packages/docs/r10.10/packages/ForTrilinos/doc/html/index.html
- MacPorts
  - http://www.macports.org/
- GCC 4.7
  - http://gcc.gnu.org/wiki/GFortranBinaries

Questions?

What ?

How ?

Where ?

Why ?

When ?

Who ?