SAND2006-5858C

Software Strategies for Flexible High-Performance Implicit Numerical Solver Libraries

Roscoe A. Bartlett

Department of Optimization and Uncertainty Estimation

Sandia National Laboratories

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

Sandia National Laboratories

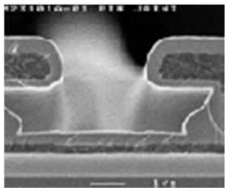
Outline

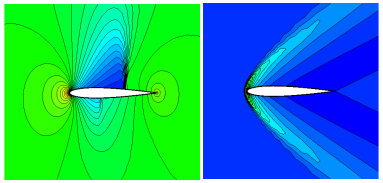
- Overview of Sandia Labs and Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Thyra
- Overview of fundamental Thyra ANA operator/vector interfaces
- Performance of Thyra-based software and algorithms
- Object-Oriented Design Pattern Efficient Abstract “Views”
- Wrapping it up

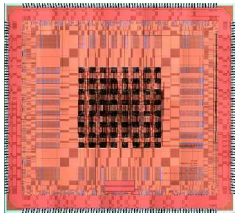
Outline

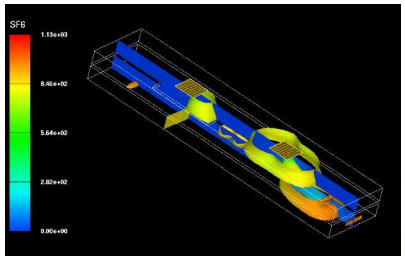
- Overview of Sandia Labs and Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Thyra
- Overview of fundamental Thyra ANA operator/vector interfaces
- Performance of Thyra-based software and algorithms
- Object-Oriented Design Pattern Efficient Abstract “Views”
- Wrapping it up

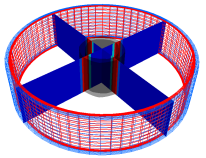
Computational Sciences at Sandia : PDEs and More …

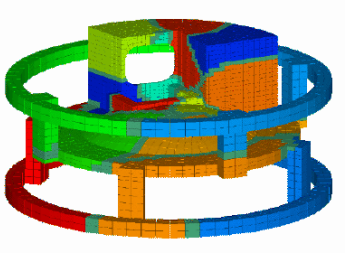




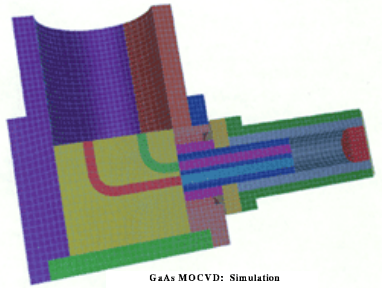


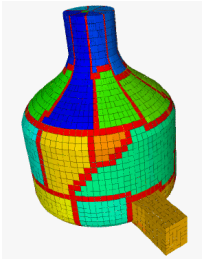


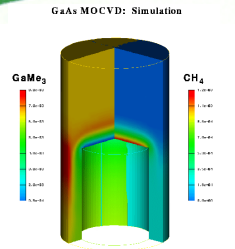


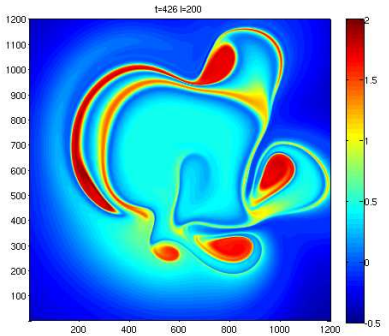


- Chemically reacting flows
- Climate modeling
- Combustion
- Compressible flows
- Computational biology
- Circuit modeling
- Inhomogeneous fluids
- Materials modeling
- MEMS modeling
- Seismic imaging
- Shock and multiphysics
- Structural dynamics
- Heat transfer
- Network modeling

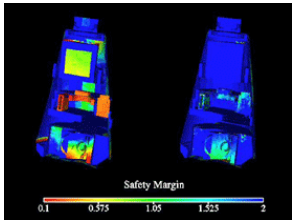


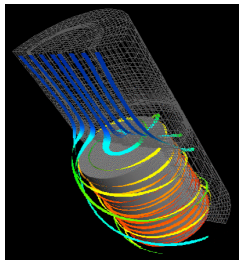


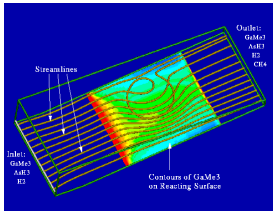


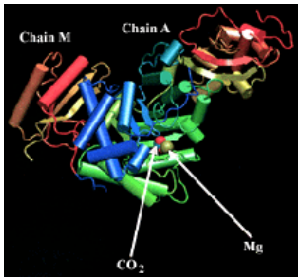


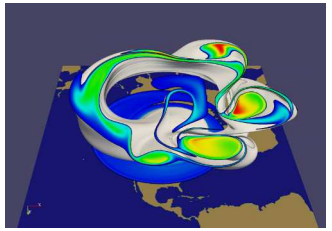
Multi-Physics is a Major Theme!











Sandia National Laboratories

Overview of Trilinos

Trilinos is being developed to:

- Provide a suite of numerical solvers to support massively parallel predictive simulation capabilities for Sandia's customers
- Provide a decoupled and scalable development environment to allow for algorithmic research that can transition to production capabilities

  => "Package"
- Provide support for growing SQA requirements
- Strategic Goals?

At its most basic level Trilinos provides:

- A common source code repository and management system (CVS based)
- A scalable configuration and build support (autoconf/automake based)
- A common infrastructure for SQA
  - Bug reporting and tracking (i.e. Bugzilla)
  - Automated regression testing and reporting (test harness, results emails and webpage)
- Developer and user communication (i.e. Mailman email lists)
- Common integrated documentation system (Trilinos website and Doxygen)

Trilinos website

http://software.sandia.gov/trilinos

Objective	Package(s)	Trilinos Package Summary Trilinos 7.0 September 2006
Linear algebra objects	Epetra, Jpetra, Tpetra	
Krylov solvers	AztecOO, Belos, Komplex	
ILU-type preconditioners	AztecOO, IFPACK	
Multilevel preconditioners	ML, CLAPS	
Eigen problems	Anasazi	
Block preconditioners	Meros	
Direct sparse linear solvers	Amesos	
Direct dense solvers	Epetra, Teuchos, Pliris	
Abstract interfaces	Thyra	
Nonlinear system solvers	NOX, LOCA, CAPO	
Time Integrators/DAEs	Rythmos	
C++ utilities, (some) I/O	Teuchos, EpetraExt, Kokkos	
Trilinos Tutorial	Didasko	
“Skins”	PyTrilinos, WebTrilinos, Star-P, Stratimikos	
Simulation-Constrained Optimization	MOOCHO	
Archetype package	NewPackage	
Other new in 7.0 release	Galeri, Isorropia, Moertel, RTOp	

Trilinos Strategic Goals

- **Scalable Solvers**: As problem size and processor counts increase, the cost of the solver will remain a nearly fixed percentage of the total solution time.
- **Hardened Solvers**: Never fail unless problem essentially unsolvable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.
- **Full Vertical Coverage**: Provide leading edge capabilities from basic linear algebra to transient and optimization solvers.
- ***Grand* Universal Interoperability**: All Trilinos packages will be interoperable, so that any combination of solver packages that makes sense algorithmically will be possible within Trilinos.
- **Universal Solver RAS**: Trilinos will be:
  - Reliable: Leading edge hardened, scalable solutions for each of these applications
  - Available: Integrated into every major application at Sandia
  - Serviceable: Easy to maintain and upgrade within the application environment.

Courtesy of Mike Heroux, Trilinos Project Leader

Trilinos Development Team

Ross Bartlett
Lead Developer of Thyra and MOOCHO
Developer of Rythmos

Paul Boggs
Developer of Thyra

Todd Coffey
Lead Developer of Rythmos

Jason Cross
Developer of Jpetra

David Day
Developer of Komplex

Clark Dohrmann
Developer of CLAPS

Michael Gee
Developer of ML, NOX

Bob Heaphy
Lead developer of Trilinos SQA

Mike Heroux
Trilinos Project Leader
Lead Developer of Epetra, AztecOO,
Kokkos, Komplex, IFPACK, Thyra, Tpetra
Developer of Amesos, Belos, EpetraExt, Jpetra

Ulrich Hetmaniuk
Developer of Anasazi

Robert Hoekstra
Lead Developer of EpetraExt
Developer of Epetra, Thyra, Tpetra

Russell Hooper
Developer of NOX

Vicki Howle
Lead Developer of Meros
Developer of Belos and Thyra

Jonathan Hu
Developer of ML

Sarah Knepper
Developer of Komplex

Tammy Kolda
Lead Developer of NOX

Joe Kotulski
Lead Developer of Pliris

Rich Lehoucq
Developer of Anasazi and Belos

Kevin Long
Lead Developer of Thyra,
Developer of Belos and Teuchos

Roger Pawlowski
Lead Developer of NOX

Michael Phenow
Trilinos Webmaster
Lead Developer of New_Package

Eric Phipps
Developer of LOCA and NOX

Marzio Sala
Lead Developer of Didasko and IFPACK
Developer of ML, Amesos

Andrew Salinger
Lead Developer of LOCA

Paul Sexton
Developer of Epetra and Tpetra

Bill Spotz
Lead Developer of PyTrilinos
Developer of Epetra, New_Package

Ken Stanley
Lead Developer of Amesos and New_Package

Heidi Thornquist
Lead Developer of Anasazi, Belos and Teuchos

Ray Tuminaro
Lead Developer of ML and Meros

Jim Willenbring
Developer of Epetra and New_Package.
Trilinos library manager

Alan Williams
Developer of Epetra, EpetraExt, AztecOO, Tpetra

Outline

- Overview of Sandia Labs and Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Thyra
- Overview of fundamental Thyra ANA operator/vector interfaces
- Performance of Thyra-based software and algorithms
- Object-Oriented Design Pattern Efficient Abstract “Views”
- Wrapping it up

Categories of Abstract Problems and Abstract Algorithms

Trilinos Packages

- **Linear Problems:** Given linear operator (matrix) $A \in \mathbf{R}^{n \times n}$
  - **Linear equations:** Solve $Ax = b$ for $x \in \mathbf{R}^n$ — Belos
  - **Eigen problems:** Solve $Av = \lambda v$ for (all) $v \in \mathbf{R}^n$ and $\lambda \in \mathbf{R}$ — Anasazi
- **Nonlinear Problems:** Given nonlinear operator $f(x,p) \in \mathbf{R}^{n+m} \rightarrow \mathbf{R}^n$
  - **Nonlinear equations:** Solve $f(x) = 0$ for $x \in \mathbf{R}^n$ — NOX
  - **Stability analysis:** For $f(x,p) = 0$ find space $p \in \mathcal{P}$ such that $\frac{\partial f}{\partial x}$ is singular — LOCA
- **Transient Nonlinear Problems:**
  - **DAEs/ODEs:** Solve $f(\dot{x}(t), x(t), t) = 0, t \in [0,T],\ x(0) = x_0,\ \dot{x}(0) = x'_0$ for $x(t) \in \mathbf{R}^n, t \in [0,T]$ — Rythmos
- **Optimization Problems:**
  - **Unconstrained:** Find $p \in \mathbf{R}^m$ that minimizes $g(p)$ — MOOCHO
  - **Constrained:** Find $x \in \mathbf{R}^n$ and $p \in \mathbf{R}^m$ that:
    minimizes $g(x,p)$
    such that $f(x,p) = 0$

Sandia National Laboratories

Introducing Abstract Numerical Algorithms

What is an abstract numerical algorithm (ANA)?

An ANA is a numerical algorithm that can be expressed abstractly solely in terms of vectors, vector spaces, linear operators, and other abstractions built on top of these without general direct data access or any general assumptions about data locality

Key Points

- ANAs can be very mathematically sophisticated!
- ANAs can be extremely reusable!
- Flexibility needed to achieve high performance!

Example : Linear Conjugate Gradients

Given:

$A \in \mathcal{X} \to \mathcal{X}$: s.p.d. linear operator

$b \in \mathcal{X}$: right hand side vector

Find vector $x \in \mathcal{X}$ that solves $Ax = b$

Linear Conjugate Gradient Algorithm

Compute $r^{(0)} = b - Ax^{(0)}$ for the initial guess $x^{(0)}$.

for $i = 1, 2, \ldots$

$\rho_{i-1} = \left\langle r^{(i-1)}, r^{(i-1)} \right\rangle$

$\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$ $(\beta_0 = 0)$

$p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)}$ $(p^{(1)} = r^{(1)})$

$q^{(i)} = Ap^{(i)}$

$\gamma_i = \left\langle p^{(i)}, q^{(i)} \right\rangle$

$\alpha_i = \rho_{i-1}/\gamma_i$

$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$

$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$

check convergence; continue if necessary

end

Types of operations

- linear operator applications
- vector-vector operations
- Scalar operations
- scalar product <x,y> defined by vector space

Types of objects

Linear Operators
- A

Vectors
- r, x, p, q

Scalars
- ρ, β, γ, α

Vector spaces?
- $\mathcal{X}$

Sandia National Laboratories

Trilinos Strategic Goals

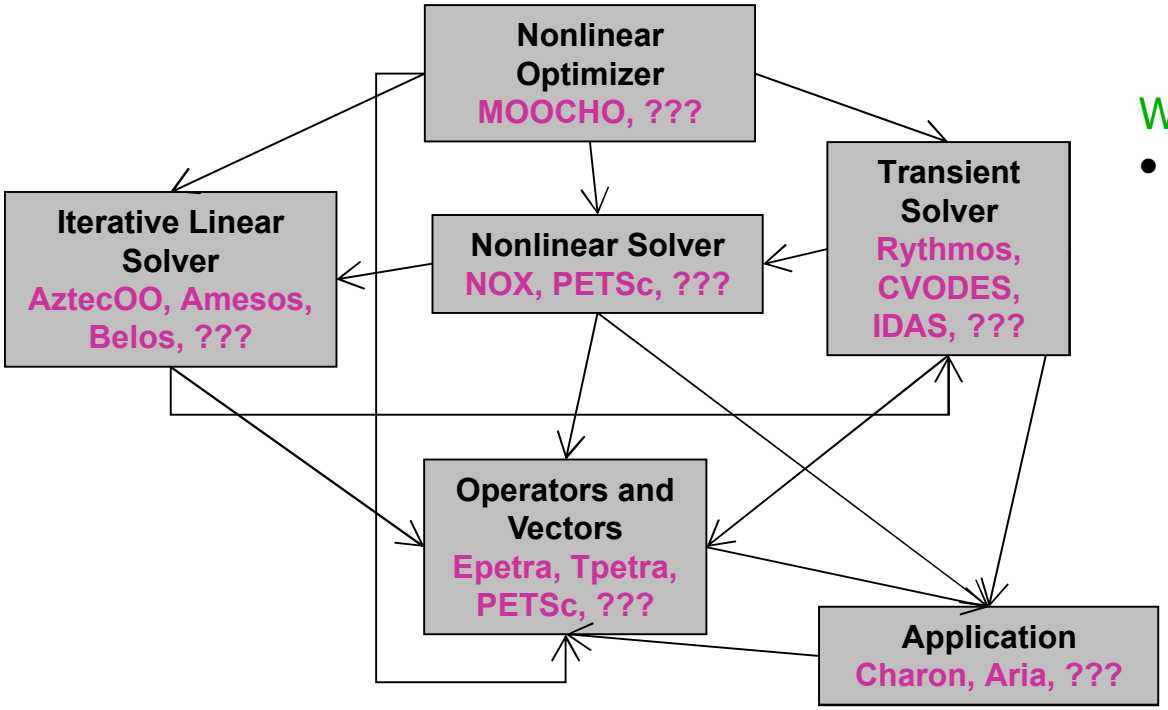
- **Scalable Solvers**: As problem size and processor counts increase, the cost of the solver will remain a nearly fixed percentage of the total solution time.
- **Hardened Solvers**: Never fail unless problem essentially unsolvable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.
- **Full Vertical Coverage**: Provide leading edge capabilities from basic linear algebra to transient and optimization solvers.
- ***Grand* Universal Interoperability**: All Trilinos packages will be interoperable, so that any combination of solver packages that makes sense algorithmically will be possible within Trilinos.
- **Universal Solver RAS**: Trilinos will be:
  - Reliable: Leading edge hardened, scalable solutions for each of these applications
  - Available: Integrated into every major application at Sandia
  - Serviceable: Easy to maintain and upgrade within the application environment.

Thyra is being developed to address this issue

Courtesy of Mike Heroux, Trilinos Project Leader

Interoperability is Especially Important to Optimization

Numerous interactions exist between layered abstract numerical algorithms (ANAs) in a transient optimization problem



What is needed to solve problem?

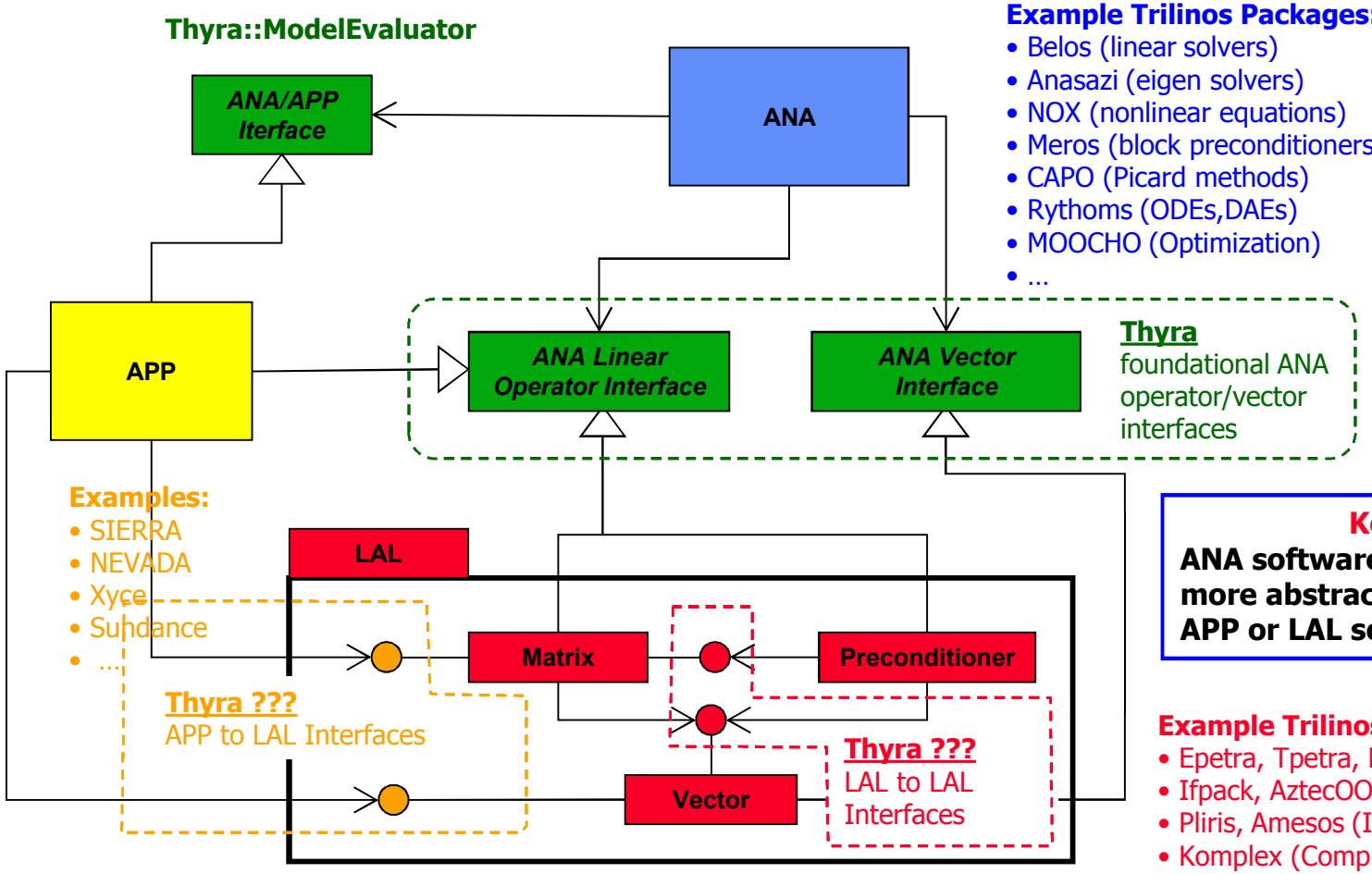
- Standard interfaces to break O(N^2) 1-to-1 couplings
  - Operators/vectors
  - Linear Solvers
  - Nonlinear solvers
  - Transient solvers
  - etc.

Thyra is being developed to address interoperability of ANAs

Key Points

- Higher level algorithms, like optimization, require a lot of interoperability
- Interoperability and layering must be "easy" or these configurations will not be achieved in practice

Software Componentization and Trilinos Interfaces



Example Trilinos Packages:
- Belos (linear solvers)
- Anasazi (eigen solvers)
- NOX (nonlinear equations)
- Meros (block preconditioners)
- CAPO (Picard methods)
- Rythoms (ODEs,DAEs)
- MOOCHO (Optimization)
- ...

Key Point
ANA software allows for a much more abstract interface than APP or LAL software!

Example Trilinos Packages:
- Epetra, Tpetra, Kokkos (Mat,Vec)
- Ifpack, AztecOO, ML (Preconditioners)
- Pliris, Amesos (Interface to direct solvers)
- Komplex (Complex/Real forms)
- ...

Three Different Types of Software Components

1) ANA : Abstract Numerical Algorithm (e.g. linear solvers, eigen solvers, nonlinear solvers, stability analysis, uncertainty quantification, transient solvers, optimization etc.)

2) LAL : Linear Algebra Library (e.g. vectors, sparse matrices, sparse factorizations, preconditioners)

3) APP : Application (the model: physics, discretization method etc.)

Requirements for Abstract Numerical Algorithms and Thyra

An important consideration ⇒ Scientific computing is computationally expensive!

Abstract Numerical Algorithms Implemented through Thyra must:

- Be portable to all ASC (advanced scientific computing) computer platforms
- Provide for stable and accurate numerics
- Result in algorithms with near-optimal storage and runtime performance
  - Scientific computing is expensive!

An important ideal ⇒ In an object-oriented interface only specify what needs to happen and not how it happens!

An important ideal ⇒ Do not constrain the possible implementation options through a bad interface design or specification!

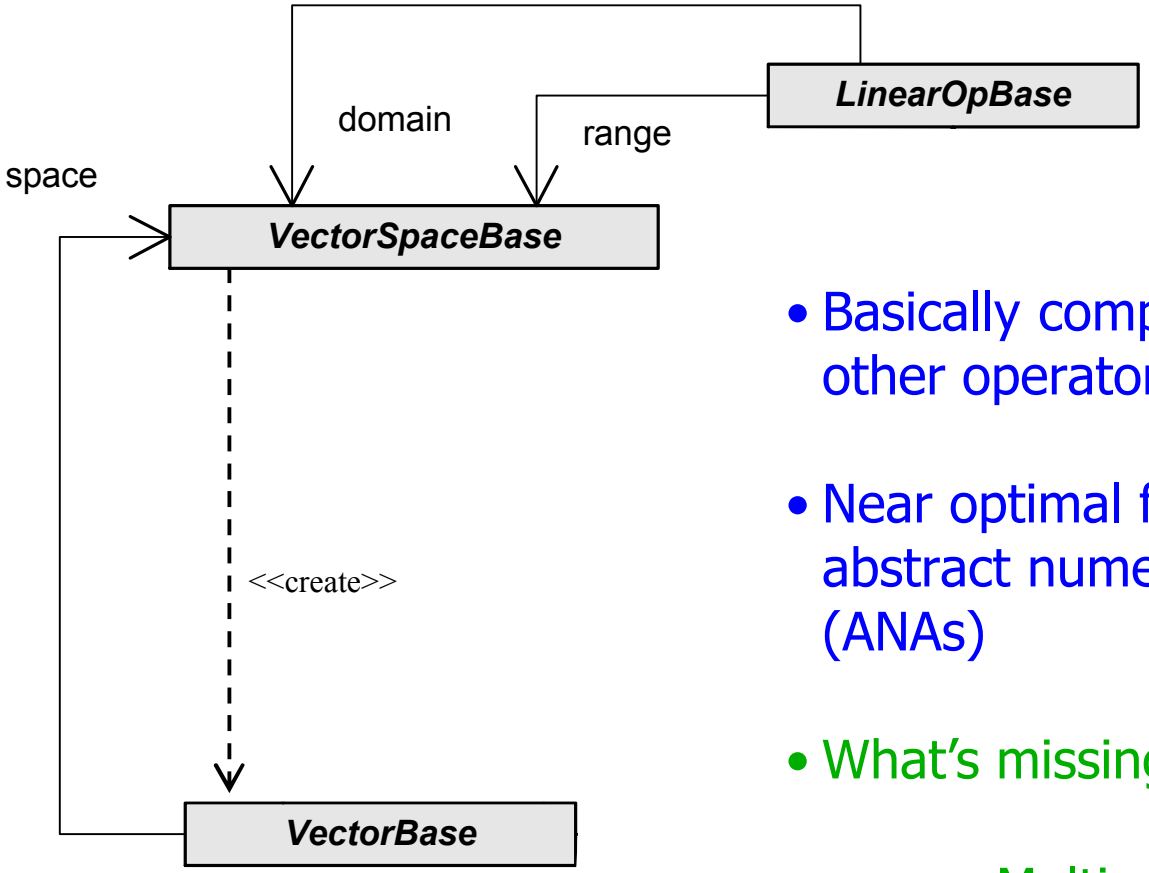
An important ideal ⇒ Object-oriented "overhead" should be constant and not increase as the problem size increases!

An important ideal ⇒ A customized hand-code algorithm in Fortran 77 should not provide *significant* improvements in storage requirements, speed, or numerical stability!

Outline

- Overview of Sandia Labs and Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Thyra
- Overview of fundamental Thyra ANA operator/vector interfaces
- Performance of Thyra-based software and algorithms
- Object-Oriented Design Pattern Efficient Abstract “Views”
- Wrapping it up

Fundamental Thyra ANA Operator/Vector Interfaces



- Basically compatible with many other operator/vector interfaces
- Near optimal for many but not all abstract numerical algorithms (ANAs)
- What's missing?

  => Multi-vectors!

UML Class Diagram

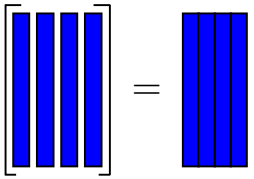
Introducing Multi-Vectors

What is a multi-vector?

- An *m* multi-vector *V* is a tall thin dense matrix composed of *m* column vectors v_j

$$V = \left[\begin{array}{cccc} v_1 & v_2 & \dots & v_m \end{array} \right] \in \mathcal{S} \times \mathbf{R}^m$$

Example: m = 4 columns

$V =$ 

What ANAs can exploit multi-vectors?

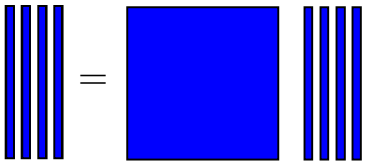
- Compact limited-memory quasi-Newton (e.g. MOOCHO)
- Tensor methods for nonlinear equations (e.g. NOX)
- Block linear solvers (e.g. Belos)
- Block eigen solvers (e.g. Anasazi)

Why are multi-vectors important?

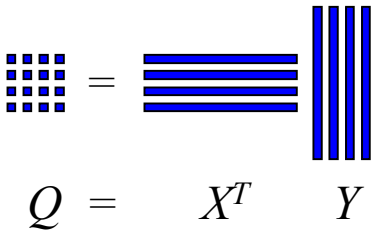
- Cache performance (level-1 to level-3 BLAS)
- Amortization of global communication

Examples of multi-vector operations

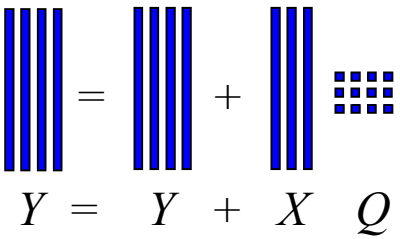
- Operator applications (i.e. mat-vecs)



- Block dot products (m^2)

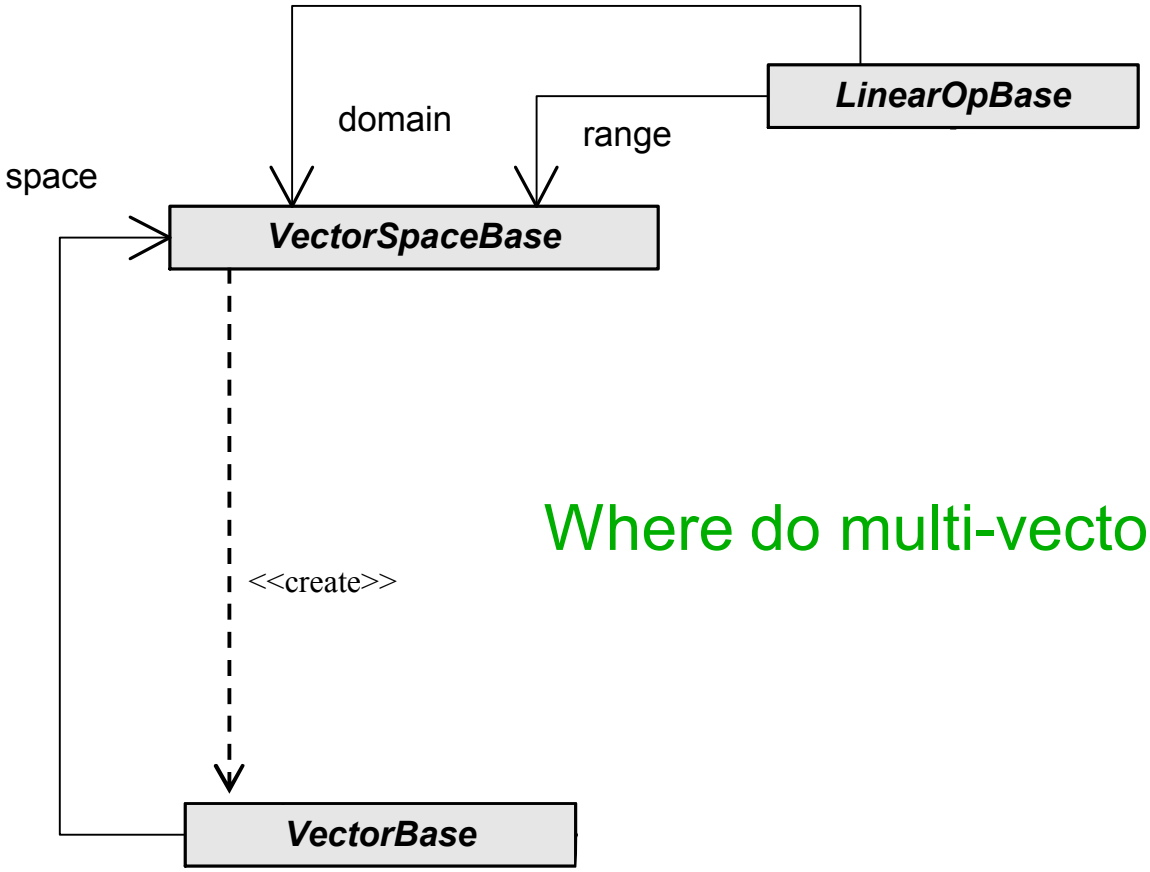


- Block update



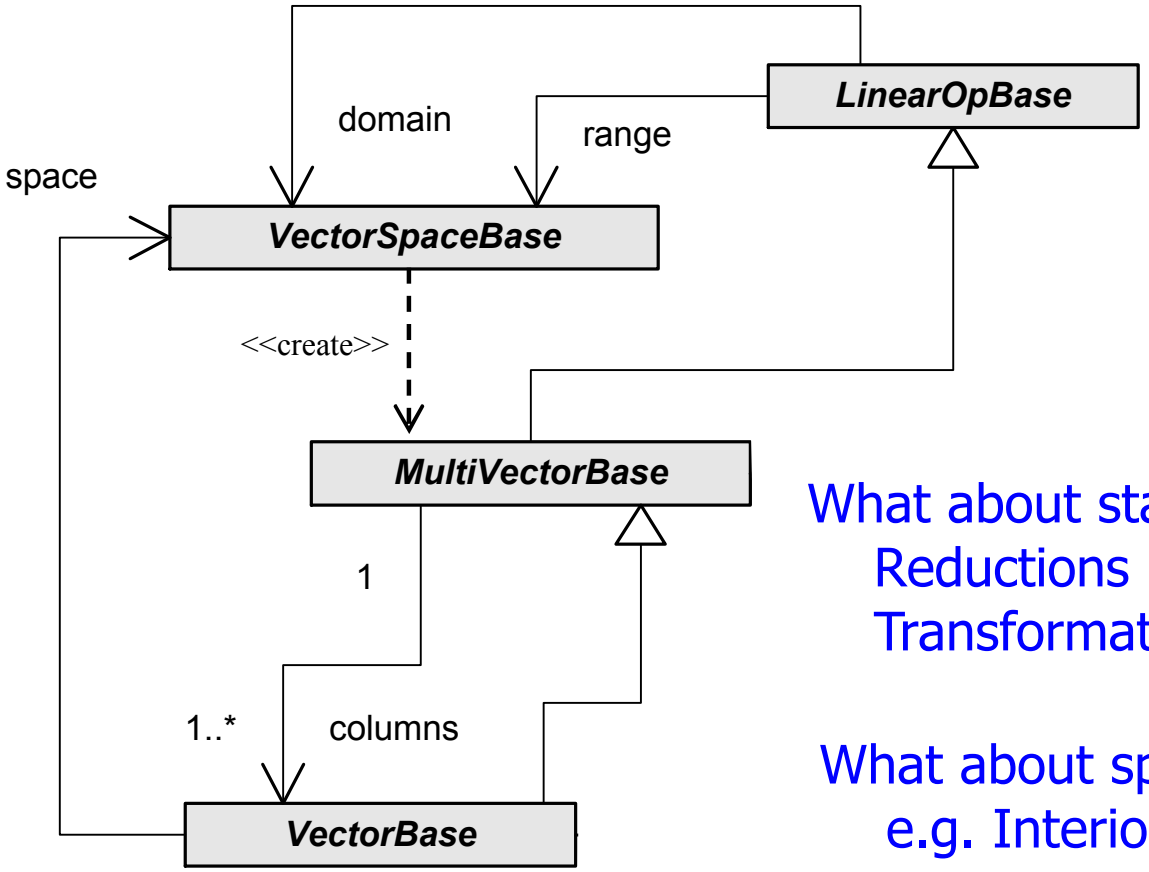
Sandia National Laboratories

Fundamental Thyra ANA Operator/Vector Interfaces



Where do multi-vectors fit in?

Fundamental Thyra ANA Operator/Vector Interfaces



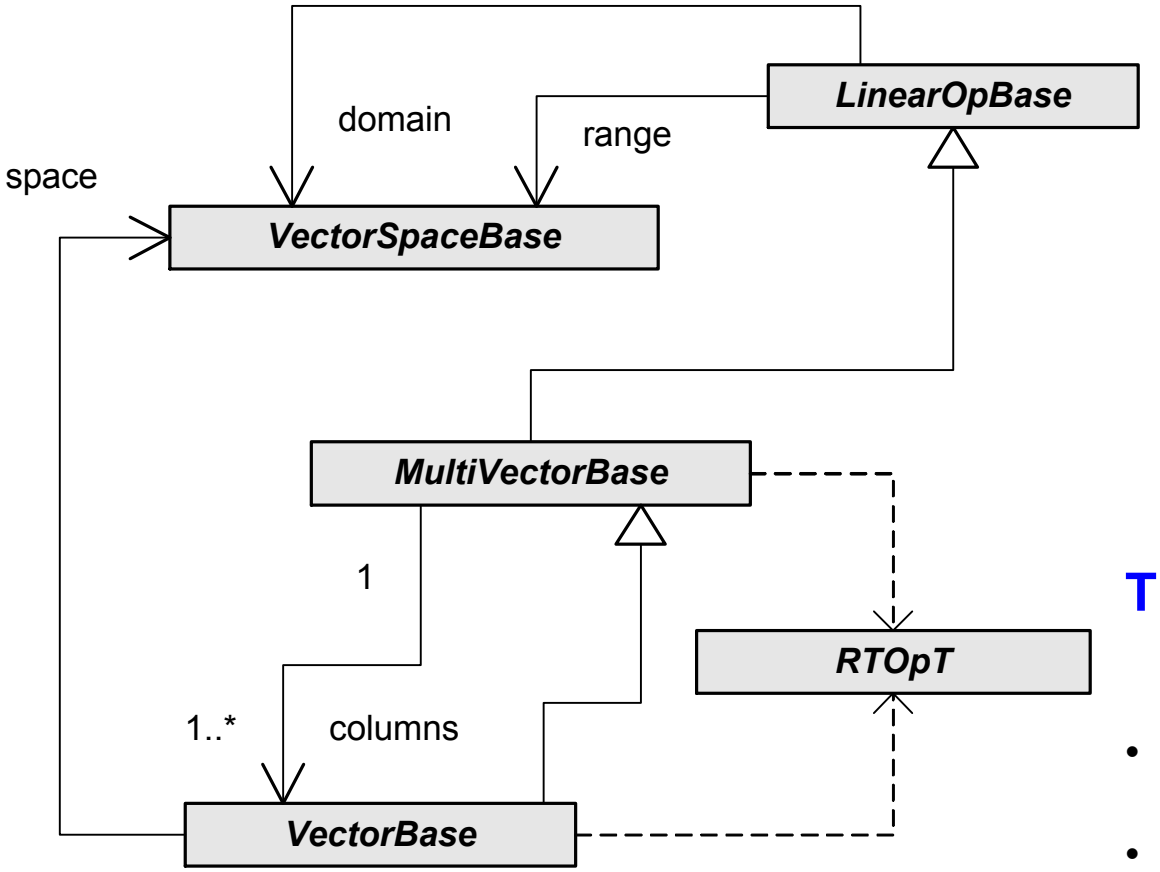
What about standard vector ops?
Reductions (norms, dot etc.)?
Transformations (axpy, scaling etc.)?

What about specialized vector ops?
e.g. Interior point methods for opt

Key Point

It is easy to come up with a list of 100 or more vector/array operations from a simple literature search into active-set, interior-point, and other algorithms!

Fundamental Thyra ANA Operator/Vector Interfaces



A Few Quick Facts about Thyra Interfaces

- All interfaces are expressed as abstract C++ base classes (i.e. object-oriented)
- All interfaces are templated on a `Scalar` data (i.e. generic)

The Key to success!

Reduction/Transformation Operators

- Supports all needed element-wise vector operations
- Data/parallel independence
- Optimal performance

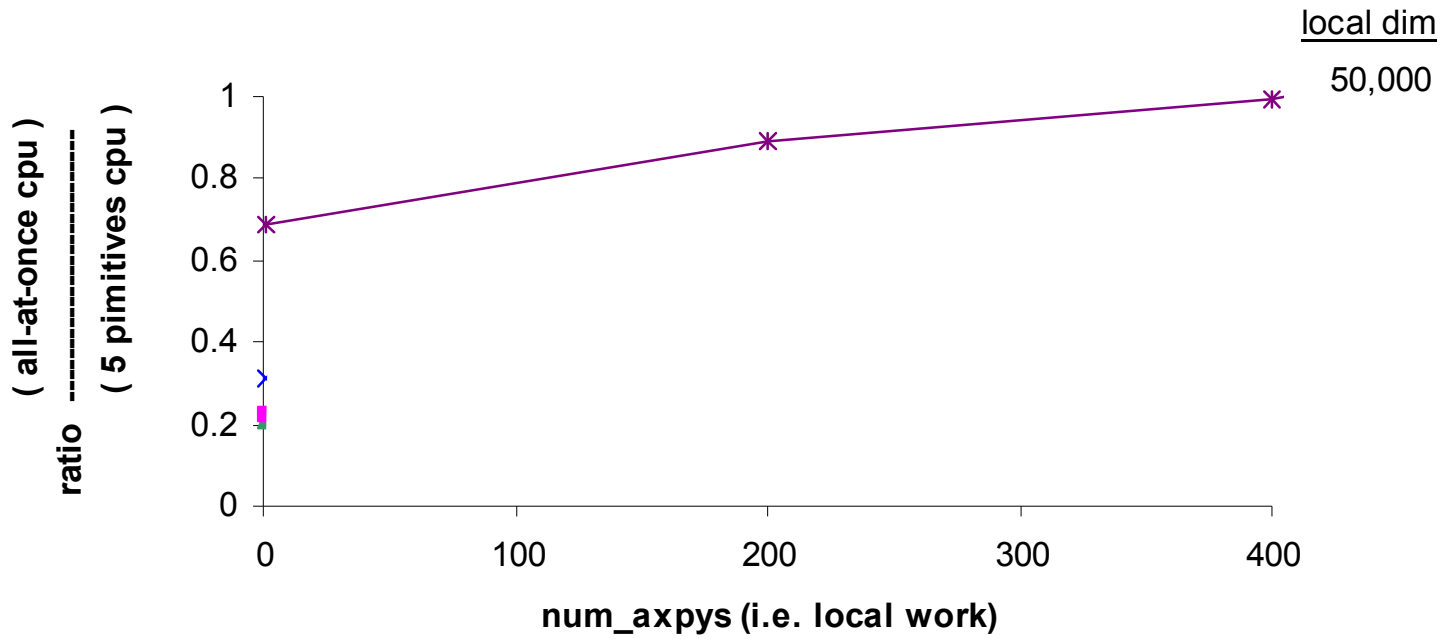
R. A. Bartlett, B. G. van Bloemen Waanders and M. A. Heroux. *Vector Reduction/Transformation Operators*, ACM TOMS, March 2004

Sandia National Laboratories

Outline

- Overview of Sandia Labs and Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Thyra
- Overview of fundamental Thyra ANA operator/vector interfaces
- Performance of Thyra-based software and algorithms
  - Level-1 Reduction/Transformation Operations
  - Matrix-[Multi]Vector Multiplication
  - Multi-Vector Views
- Object-Oriented Design Pattern Efficient Abstract “Views”
- Wrapping it up

RTOp vs. Primitives : Distributed Process Communication



* 128 processors on CPlant®

- **Compare**
  - **RTOp (all-at-once reduction (i.e. ISIS++ QMR solver))**

    $\{ \alpha, \gamma, \xi, \rho, \varepsilon \} \leftarrow \{ (x^T x)^{1/2}, (v^T v)^{1/2}, (w^T w)^{1/2}, w^T v, v^T t \}$
  - **Primitives (5 separate reductions)**

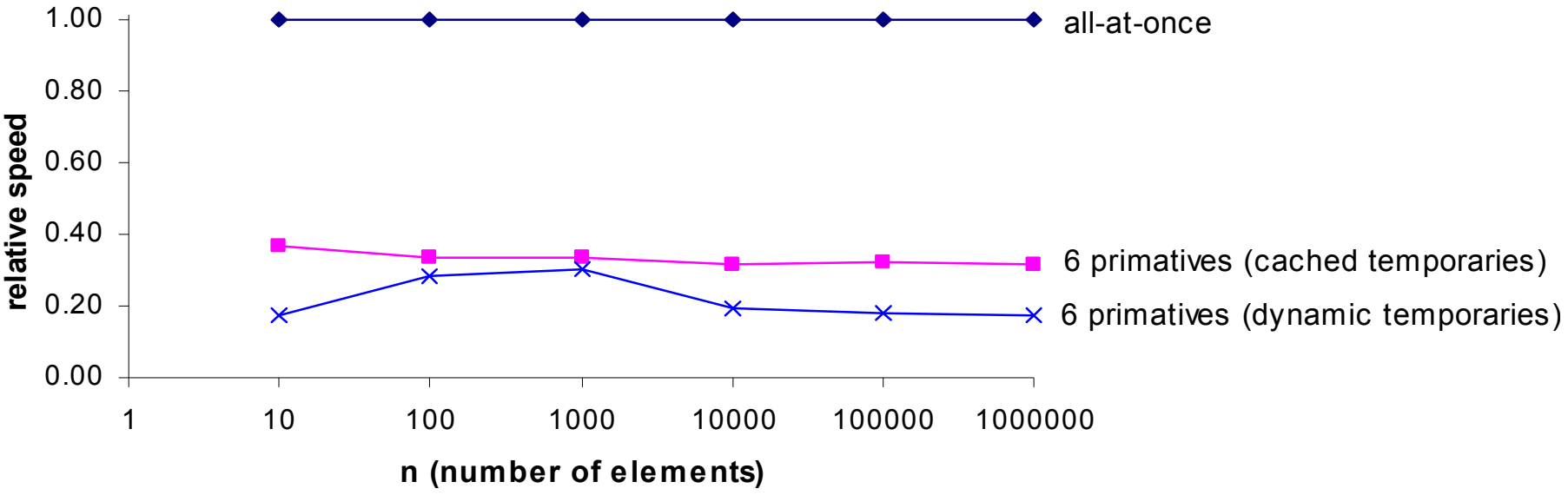
    $\alpha \leftarrow (x^T x)^{1/2}, \gamma \leftarrow (v^T v)^{1/2}, \xi \leftarrow (w^T w)^{1/2}, \rho \leftarrow w^T v, \varepsilon \leftarrow v^T$

Key Point

RTOp allows for amortization of global communication

Sandia National Laboratories

RTOp vs. Primitives : Local Multiple Ops and Temporaries



* 1 processor (gcc 3.1 under Linux)

- **Compare**
  - **RTOp (all-at-once reduction)**

    $\{ \max \alpha : x + \alpha d \geq \beta \} = \min\{ \max((\beta - x_i)/d_i, 0), \text{for } i = 1 \ldots n \} \rightarrow \alpha$
  - **Primitives (5 temporaries, 6 vector operations)**

    $-x_i \rightarrow u_i, \quad x_i + \beta \rightarrow v_i, \quad v_i / d_i \rightarrow w_i, \quad 0 \rightarrow y_i, \quad \max\{w_i, y_i\} \rightarrow z_i, \quad \min\{z_i, i=1 \ldots n\} \rightarrow \alpha$

Key Point

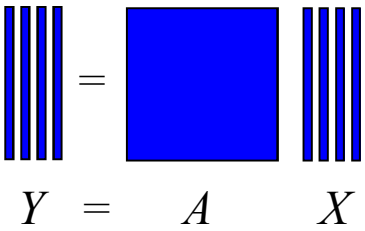
RTOp provides for better cache performance and avoids multiple read/writes

Outline

- Overview of Sandia Labs and Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Thyra
- Overview of fundamental Thyra ANA operator/vector interfaces
- Performance of Thyra-based software and algorithms
  - Level-1 Reduction/Transformation Operations
  - Matrix-[Multi]Vector Multiplication
  - Multi-Vector Views
- Object-Oriented Design Pattern Efficient Abstract “Views”
- Wrapping it up

Speeding Up Matrix-[Multi]Vector Multiplication

Matrix-Multivector Multiplication



Amortization of Global communication
= > Epetra

Local sparse matrix-multivector multiplcation
=> Kokkos (F77)

Key Point
More than a 3X speedup for 5 RHSs!

Kokkos Results (Local Effects Only)

Pentium M 1.6GHz Cygwin/GCC 3.2 (WinXP Laptop)

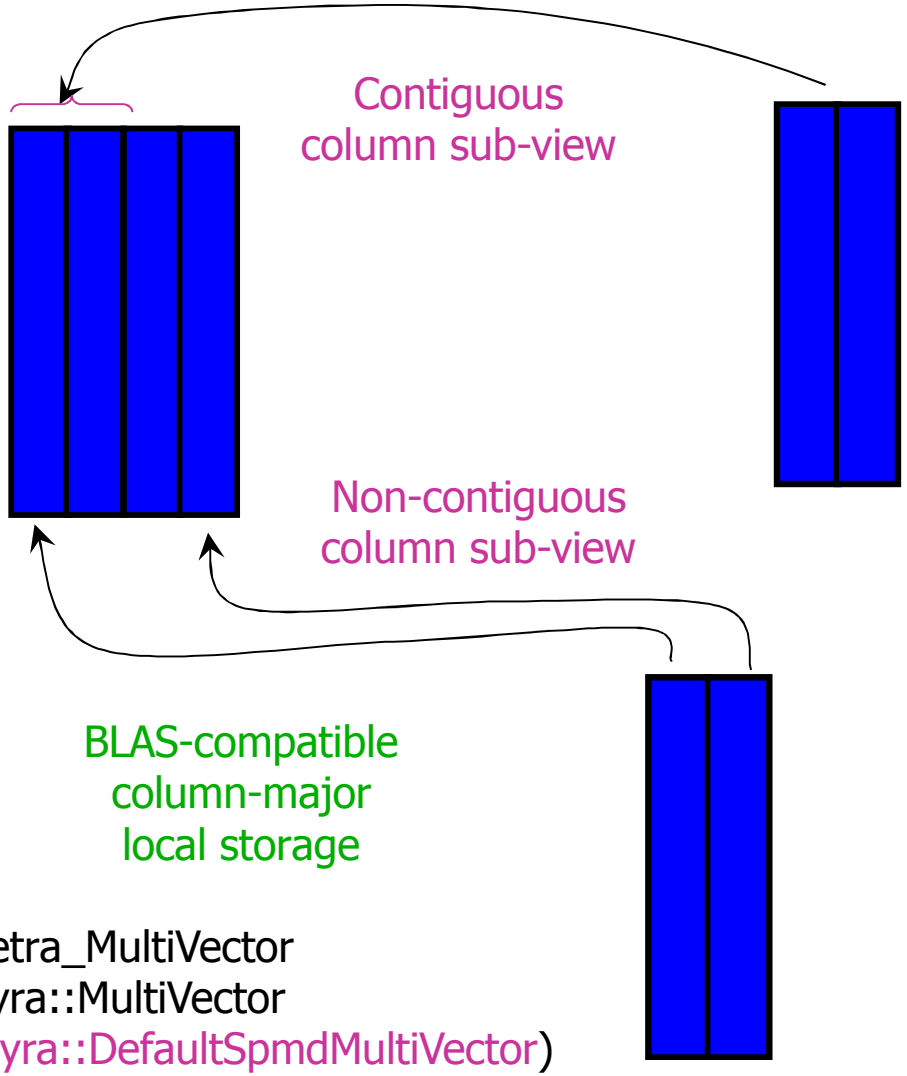
Data Set	Dimension	Nonzeros	# RHS	MFLOPS
DIE3D	9873	1733371	1	247.62
			2	418.94
			3	577.79
			4	691.62
			5	787.90
dday01	21180	923006	1	230.75
			2	407.96
			3	553.80
			4	668.24
			5	738.40
FIDAP035	19716	218308	1	171.22
			2	349.29
			3	374.24
			4	498.99
			5	545.77

Courtesy of Mike Heroux

Outline

- Overview of Sandia Labs and Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Thyra
- Overview of fundamental Thyra ANA operator/vector interfaces
- Performance of Thyra-based software and algorithms
  - Level-1 Reduction/Transformation Operations
  - Matrix-[Multi]Vector Multiplication
  - Multi-Vector Views
- Object-Oriented Design Pattern Efficient Abstract “Views”
- Wrapping it up

Multi-Vectors and Multi-Vector Views



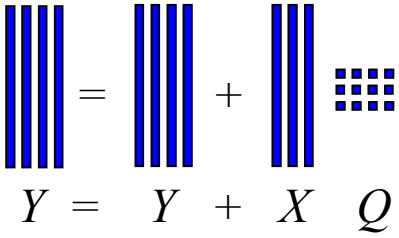
- Epetra_MultiVector
- Thyra::MultiVector
  (Thyra::DefaultSpmdMultiVector)

Use cases for Multi-vector views

- Single-RHS GMRES
- Block Krylov methods
- Compact limited memory quasi-Newton methods
- …

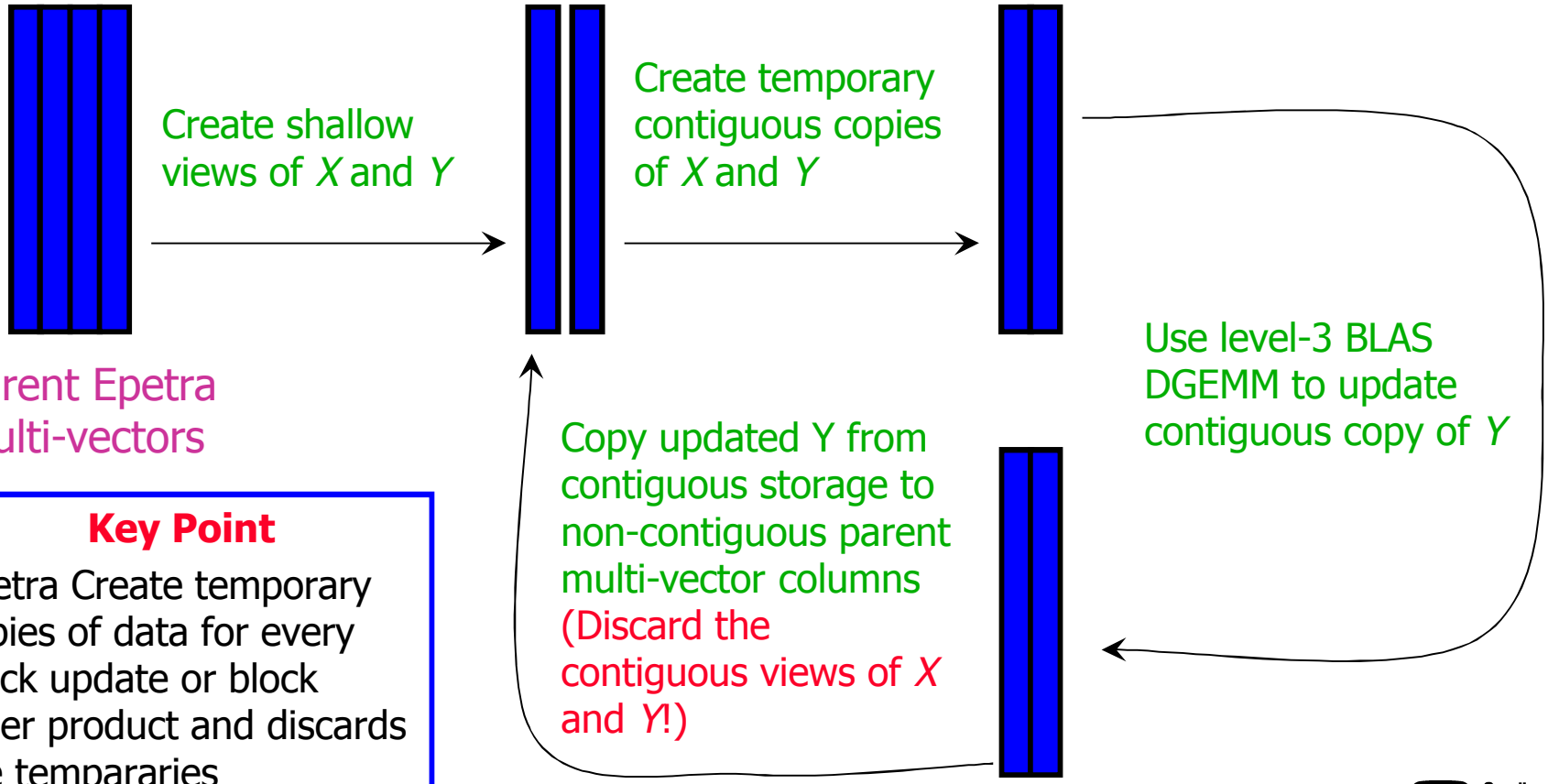
Epetra Multi-Vector Non-Contiguous Column Views

- Block update



Key Point

Epetra does **not** create temporary contiguous storage for matrix-multivector multiplication!



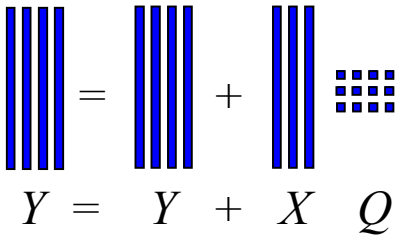
Key Point

Epetra Create temporary copies of data for every block update or block inner product and discards the tempararies

Sandia National Laboratories

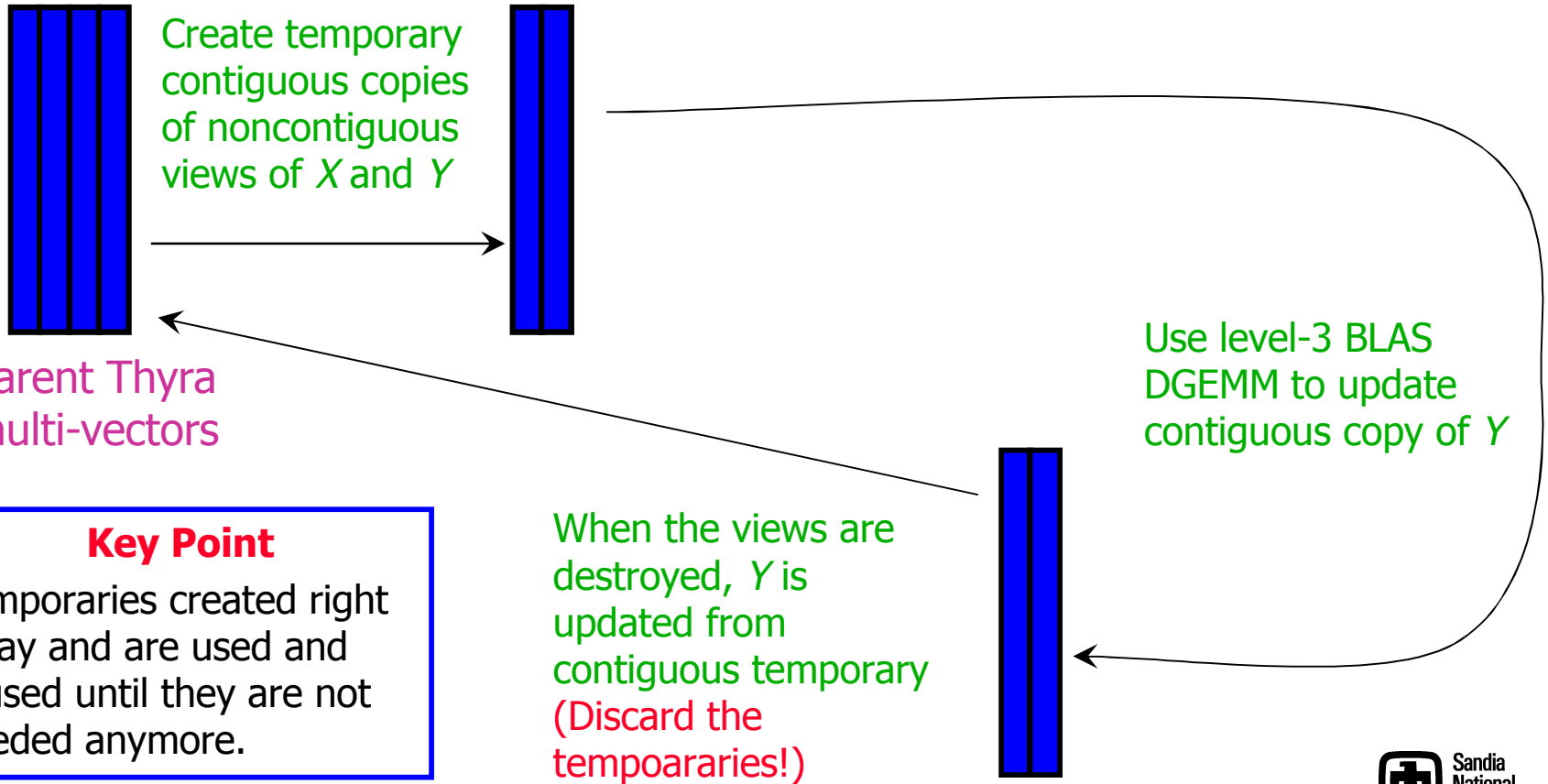
Thyra SPMD Multi-Vector Non-Contiguous Column Views

- Block update



Key Point

Contiguous copies used for **all** operations including block inner product and matrix-multivector multiplication.



Key Point

Temporaries created right away and are used and reused until they are not needed anymore.

Sandia National Laboratories

Performance of Pure Epetra vs. Epetra/Thyra Adatpers

Test program put together by Chris Baker for Anasazi

1) Apply Operator:

```
V = A * X

where: X and V
       are numel x n
```

2) Block multivector update (MvTimeMatAddMv)

```
Z = alpha*Z + X * T

where: Z is numel x n, X is numel x m,
       and T is m x n
```

3) Block inner (i.e. dot) product (MvTimesMv)

```
T = X' * Y

where: X is numel x m,
       Y is numel x n,
       and T is m x n
```

In these tests numel=50000, m=100, and n=5.

CPU Times for Epetra vs. Default Thyra SPMD Implementation with Epetra/Thyra Adapters

*** Pentium 1.8GHz Linux/g++ 3.4.3**

		epetra	thyra/epetra
FULL:	Apply Operator	0.153514	0.14886
FULL:	MvTimeMatAddMv	0.278658	0.278105
FULL:	MvTransMv	0.185737	0.185522
CONT-VIEW:	Creation	1.3e-05	3.2e-05
CONT-VIEW:	Apply Operator	0.151971	0.14631
CONT-VIEW:	MvTimeMatAddMv	0.278541	0.27983
CONT-VIEW:	MvTransMv	0.1857	0.1864
VIEW:	Creation	1.5e-05	0.116881
VIEW:	Apply Operator	0.328524	0.146495
VIEW:	MvTimeMatAddMv	0.38062	0.279953
VIEW:	MvTransMv	0.28367	0.185714

Similar performance for whole multi-vectors and contiguous multi-vector views

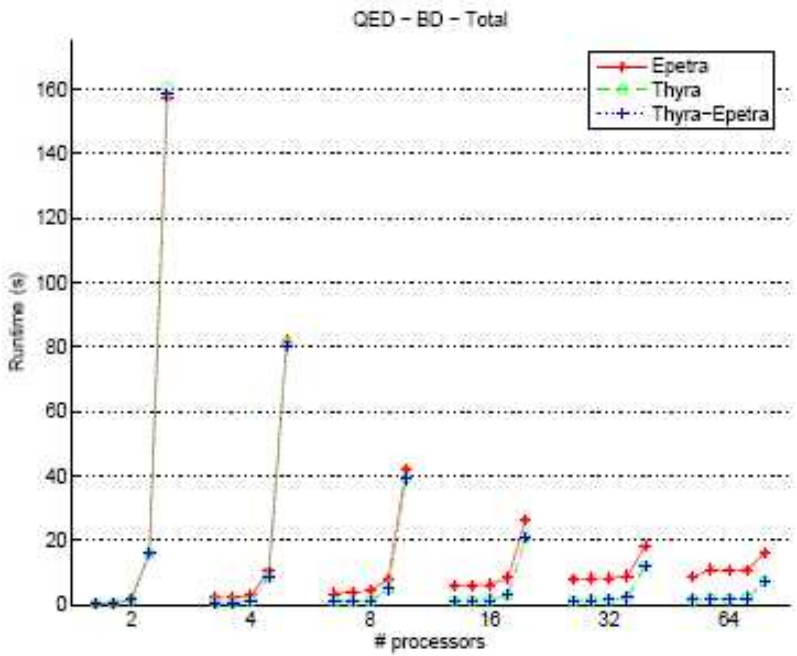
Overall, better performance for Epetra/Thyra adapters than for pure Epetra for noncontiguous views!

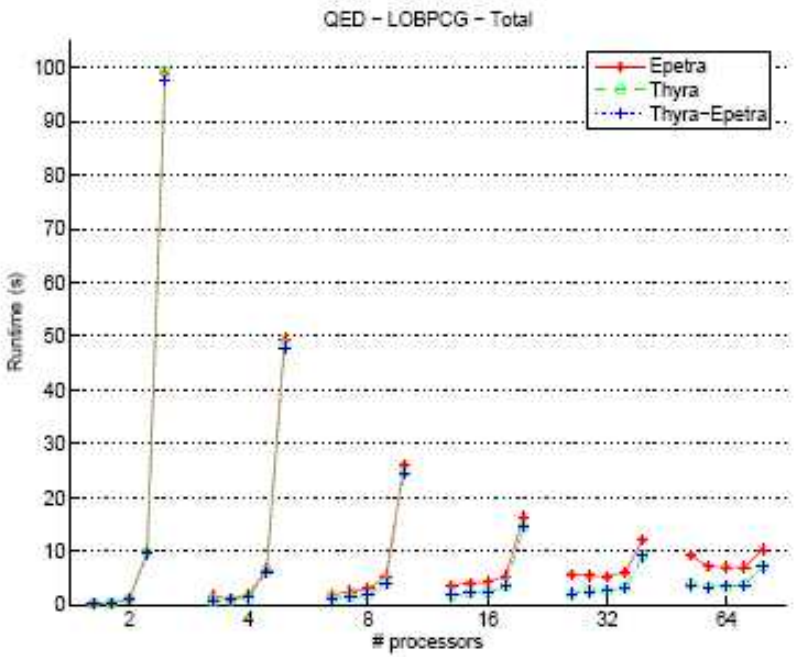
Key Point

Abstraction and implementation flexibility is key to achieving high performance!

Sandia National Laboratories

Overall Performance of Pure Epetra vs. Epetra/Thyra Adatpers





Performance of block Block Davidson and LOBPCG algorithms in Anasazi:

- Block size = 5
- Global sizes n = 10^3, 10^4 , 10^5 , 10^6
- QED (32 node Linux cluster, 2.8GHz dual processors)
- See Trilinos/packages/anasazi/doc/ThyraPerf/thyra_overhead.pdf

Key Point

Thyra-based implementation performs as well or out-performs Epetra-based implementation

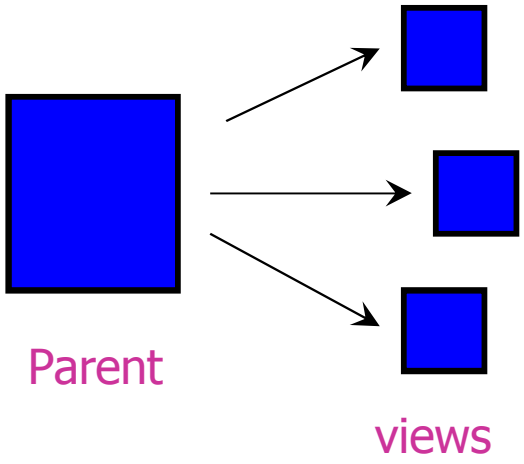
Courtesy of Chris Baker and Heidi Thornquist (Lead Anasazi Developer)

Sandia National Laboratories

Outline

- Overview of Sandia Labs and Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Thyra
- Overview of fundamental Thyra ANA operator/vector interfaces
- Performance of Thyra-based software and algorithms
- Object-Oriented Design Pattern Efficient Abstract “Views”
- Wrapping it up

Object-Oriented Design Pattern Efficient Abstract "Views"



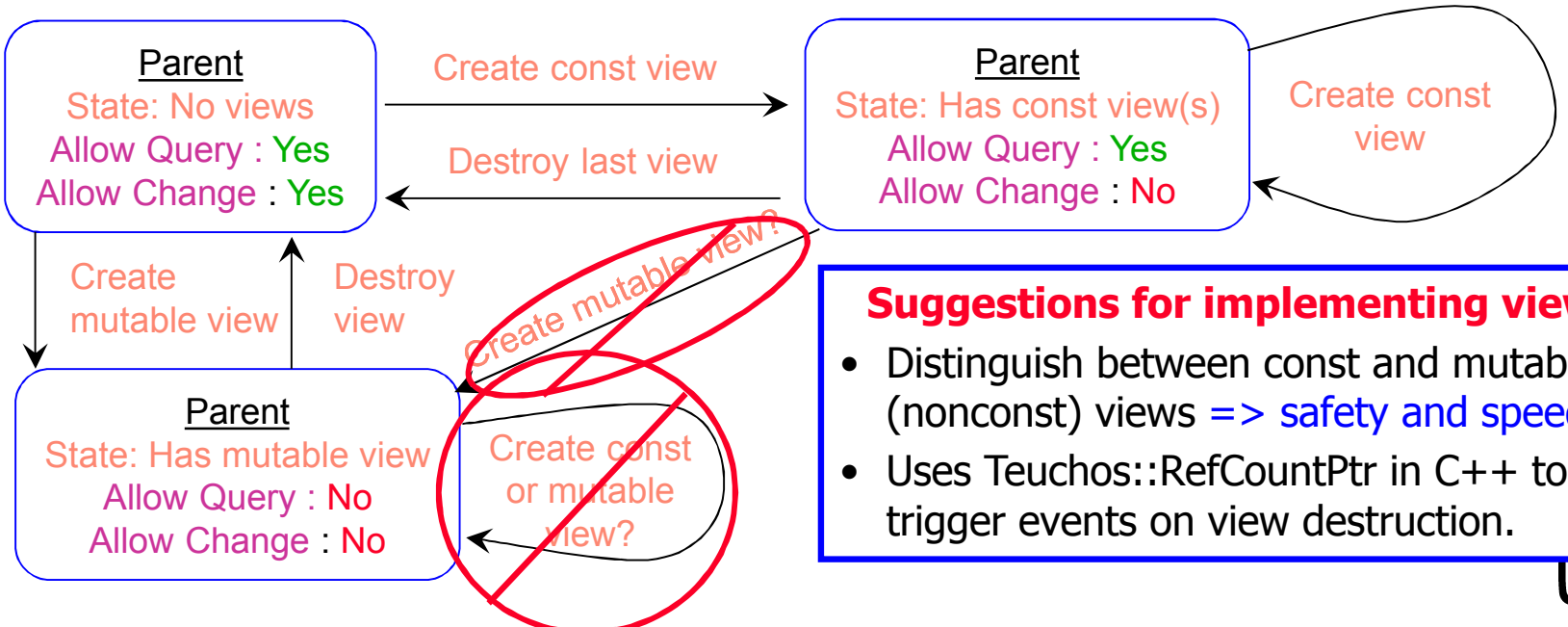
Examples of Views

- Abstract multi-vector column views
- Explicit element access
- Sub-matrix views
- …

Interface design issues

- Don't constrain freedom to achieve high performance (i.e. allow for "detached" views)
- Don't make it too difficult to provide correct concrete implementations

Suggested Behavior of Parent and Views (UML State Diagram)



Suggestions for implementing views

- Distinguish between const and mutable (nonconst) views => safety and speed!
- Uses Teuchos::RefCountPtr in C++ to trigger events on view destruction.

Outline

- Overview of Sandia Labs and Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Thyra
- Overview of fundamental Thyra ANA operator/vector interfaces
- Performance of Thyra-based software and algorithms
- Object-Oriented Design Pattern Efficient Abstract “Views”
- Wrapping it up

Summary

- Trilinos:
  - provides a suite of numerical solvers to support massively parallel predictive simulation capabilities for Sandia's customers
  - provides a decoupled and scalable development environment based on packages to allow for algorithmic research that can transition to production capabilities
  - provides support for growing SQA requirements
- Abstract Numerical Algorithms(ANAs):
  - are mathematically sophisticated but are independent of the implementation characteristics of vectors, vector spaces, linear operators or abstractions built on top of these
  - can be extremely reusable if implemented in the right way
  - allow great freedom in the implementation of operators/vectors etc. to achieve high performance
- Thyra:
  - defines basic interoperability interfaces between ANAs and concrete implementations of operators/vectors etc.
  - allows for unmatched flexibility in the implementation and performance of ANA objects
  - allows ANAs that are stable and near optimal in speed and storage

Acknowledgements and the TUG

- Paul Boggs (ptboggs@sandia.gov) : Split/O3D lead developer, Thyra developer
- Todd Coffey (tscoffe@sandia.gov) : Rythmos lead developer, Thyra developer
- Michael Heroux (maherou@sandia.gov): Trilinos leader, Epetra and AztecOO lead developer, Thyra developer
- Rob Hoekstra (rjhoeks@sandia.gov) : EpetraExt lead developer
- Victoria Howle (vehowle@sandia.gov) : Meros lead developer, Thyra developer
- Kevin Long (krlong@sandia.gov) : Sundance lead developer, Thyra developer
- Roger Pawlowski (rppawlo@sandia.gov) : NOX lead developer
- Eric Phipps (etphipp@sandia.gov) : LOCA lead developer, Thyra developer
- Bill Spotz (wfspotz@sanida.gov) : PyTrilinos Lead developer, Thyra/Python adapter developer
- Heidi Thornquist (hkthorn@sandia.gov): Belos and Anasazi lead developer, Thyra developer
- Allan Williams (william@sandia.gov) : FEI lead developer, Thyra developer