Trilinos Introduction and Overview

Matthias Mayr[1,2]

HPSF Community Summit | TU Braunschweig | February 25 - 27, 2026

[1] Institute for Mathematics and Computer-Based Simulation, Universität der Bundeswehr München

[2] Data Science & Computing Lab, Universität der Bundeswehr München

I. Introduction to Trilinos

1 Background and motivation

2 Performance Portability through Kokkos

3 Organization

4 Software framework

Disclaimer

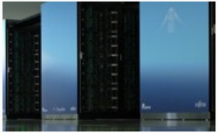
The following slides will give a brief overview over the software package Trilinos. It is far from complete, but on the final slides, some *references to additional introductory material and tutorials will be given*.

What is Trilinos?

TRILINOS

An Open-Source Library of Software for Scientific Computing

Mission statement[1]: *"The* Trilinos *Project is an effort to facilitate the design, development, integration, and ongoing support of mathematical software libraries and enabling technologies within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems on new and emerging high-performance computing (HPC) architectures".*



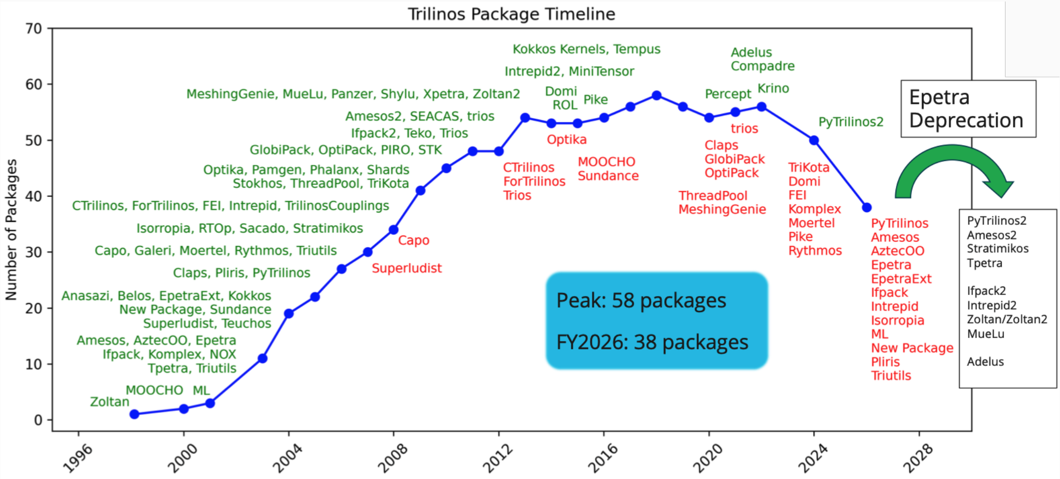
Some history and statistics

Evolution:

- Originally: three packages for distributed memory systems
  - AztecOO
  - Epetra
  - ML
- Peak: 58 packages
- Today: ≈38 packages

Statistics:

- First commit on Fri Dec 14 22:43:40 2001
- Transition to GitHub on Nov 11, 2015
- $\geq 105k$ commits
- $\geq$ 600 contributors


Trilinos Package Timeline

Peak: 58 packages

FY2026: 38 packages

Epetra Deprecation

PyTrilinos2
Amesos2
Stratimikos
Tpetra

Ifpack2
Intrepid2
Zoltan/Zoltan2
MueLu

Adelus

Why using Trilinos?

Wide range of functionality (organized in 5 *product areas*)

Core	Vectors, matrices, graphs and similar data containers, and related operations
Linear Solvers and Preconditioners	For large, distributed systems of equations
Nonlinear solvers and analysis tools	Includes basic nonlinear approaches, continuation methods and similar
Discretization Tools	Tools for the discretization of integral and differential equations
Framework	Tools for building, testing, and integrating Trilinos capabilities

Modern Trilinos: Performance Portability through Kokkos

Performance portability for various parallel programming paradigms

Trilinos targets all major parallel architectures, including

- distributed-memory using the Message Passing Interface (MPI),
- multicore using a variety of common approaches,
- accelerators using common and emerging approaches, and
- vectorization.

Performance portability is achieved through the Kokkos programming model[2].

“ *. . . as long as a given algorithm and problem size contain enough latent parallelism,* ***the same Trilinos source code*** *can be compiled and execution on* ***any reasonable combination of distributed, multicore, accelerator and vectorizing computing devices****.*” — Trilinos Website

Organization of the Trilinos project: governance

Strategic Leadership

- Big picture and vision
- Current Strategic Leaders
  - *Christian Glusa*
  - *Eric Phipps*
  - *Siva Rajamanickam*
  - *Heidi Thornquist*
  - *Jim Willenbring*
  - *Michael Wolf*

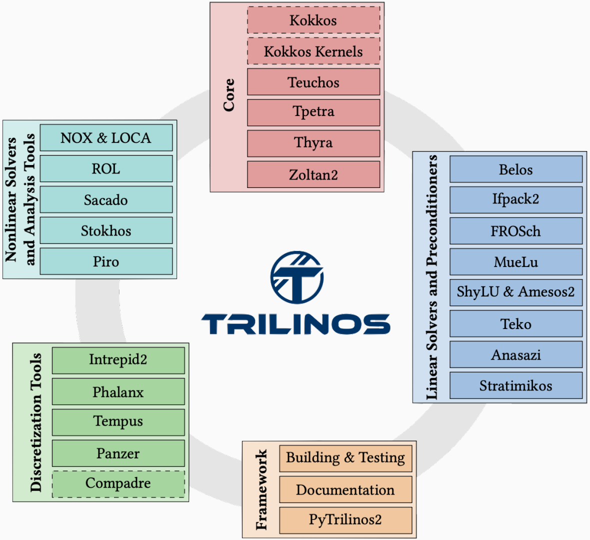
Operational Leadership

- Take care of day-to-day business
- Current Operational Leaders
  - *Sam Browne* – **Framework**
  - *Jonathan Hu* – **Solvers**
  - *Curt Ober* – **Product manager**
  - *Roger Pawlowski* – **Trilinos Core**
  - *Mauro Perego* – **Discretizations and Analysis**

Organization of the Trilinos project: software perspective

- Features and capabilities divided into *packages*
- Each package
  - ... focuses on a set of unique capabilities for a specific task
  - ... is semi-autonomous with clear dependencies and its dedicated development team
- Most packages developed in Trilinos
- Some packages developed externally and snapshotted into Trilinos for user convenience (e.g. Kokkos, Compadre, ...)

Product Areas and their packages



TRILINOS

Core
- Kokkos
- Kokkos Kernels
- Teuchos
- Tpetra
- Thyra
- Zoltan2

Linear Solvers and Preconditioners
- Belos
- Ifpack2
- FROSch
- MueLu
- ShyLU & Amesos2
- Teko
- Anasazi
- Stratimikos

Framework
- Building & Testing
- Documentation
- PyTrilinos2

Discretization Tools
- Intrepid2
- Phalanx
- Tempus
- Panzer
- Compadre

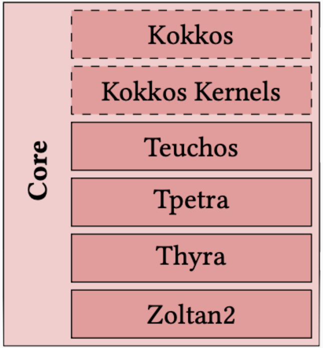
Nonlinear Solvers and Analysis Tools
- NOX & LOCA
- ROL
- Sacado
- Stokhos
- Piro

Product Area: Core

Objective

Provide essential tools for managing and distributing data across processing elements in parallel computing environments

- Kokkos / Kokkos Kernels: node-level performance portability
- Tpetra: MPI-distributed linear algebra
- Teuchos: utilities and parameter lists
- Thyra: abstract interfaces
- Zoltan2: partitioning/load balancing

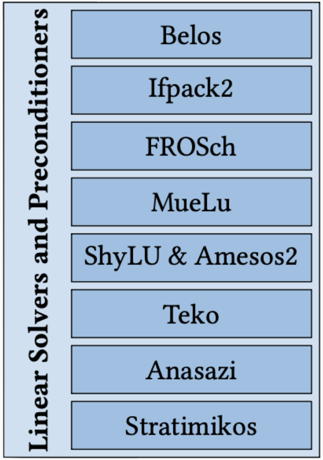


Product Area: Linear Solvers and Preconditioners

Objective

Provide linear solver capabilities for dense and sparse systems

- Belos: Krylov methods
- Amesos2/ShyLU: direct solvers
- Ifpack2: one-level domain decomposition solvers (often used as smoothers)
- FROSch: multi-level domain decomposition solvers
- MueLu: algebraic multigrid methods (AMG)
- Teko: block preconditioning for multiphysics applications
- Anasazi: eigensolvers
- Stratimikos: unified Thyra-driven configuration of linear solvers



Linear Solvers and Preconditioners

Belos

Ifpack2

FROSch

MueLu

ShyLU & Amesos2

Teko

Anasazi

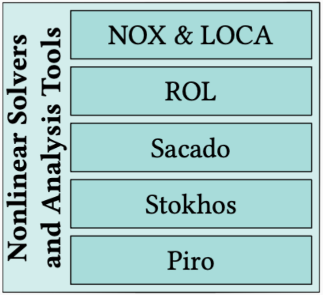
Stratimikos

Product Area: Nonlinear Solvers and Analysis Tools

Objective

Provide top-level algorithms for computational simulations and design studies

- NOX & LOCA: nonlinear solvers & continuation/bifurcation analysis
- ROL: optimization
- Stokhos: Uncertainty Quantification
- Sacado: Automatic Differentiation
- Piro: drivers to compose nonlinear analysis workflows



Nonlinear Solvers and Analysis Tools

NOX & LOCA

ROL

Sacado

Stokhos

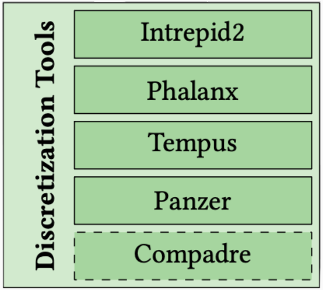
Piro

Product Area: Discretization Tools

Objective

Provide tools for spatial and temporal discretization of integro-differential equations

- Intrepid2: element-level finite element building blocks
- Phalanx: field evaluation based on a directed acyclic graph (DAG)
- Panzer: global FE assembly + solver integration
- Tempus: time integration
- Compadre: meshless operators/remap

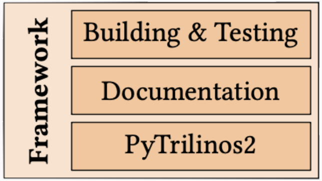


Product Area: Framework

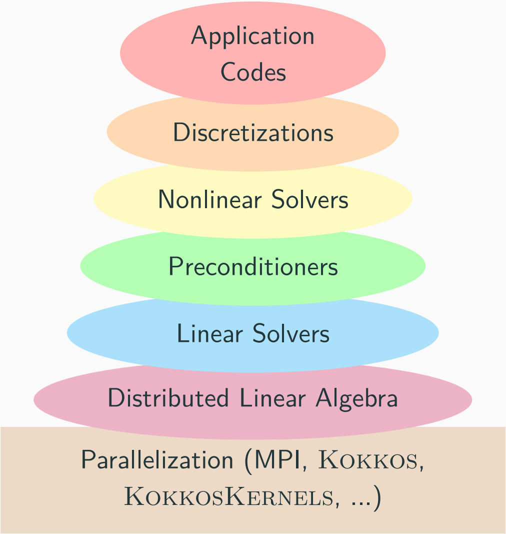
Objective

Provide and maintaine supporting infrastructure for Trilinos users and developers

- TriBITS: consistent configure/build/test of package dependency subgraphs
- Documentation: source code documentation and tutorials
- PyTrilinos2: expose selected capabilities to Python



Layers of a Trilinos-based application



Application Codes

Discretizations

Nonlinear Solvers

Preconditioners

Linear Solvers

Distributed Linear Algebra

Parallelization (MPI, Kokkos, KokkosKernels, ...)

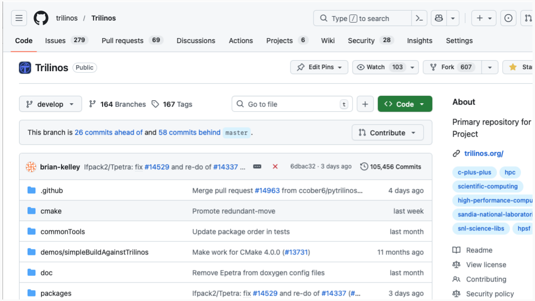
Trilinos resources

Source code repository

- GitHub: https://github.com/trilinos/Trilinos
- Default branch: master
- Development branch: develop

Website

- Link: https://trilinos.github.io
- Provides general information
- Details on all packages
- Links to Doxygen source code documentation

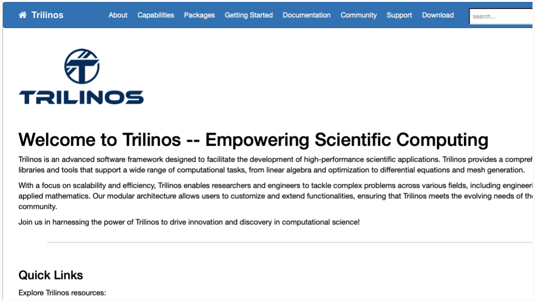


This branch is 26 commits ahead of and 58 commits behind `master`.

brian-kelley Ifpack2/Tpetra: fix #14529 and re-do of #14337 ... ✕ 6dbac32 · 3 days ago 105,456 Commits

.github	Merge pull request #14963 from ccober6/pytrilinos...	4 days ago
cmake	Promote redundant-move	last week
commonTools	Update package order in tests	last month
demos/simpleBuildAgainstTrilinos	Make work for CMake 4.0.0 (#13731)	11 months ago
doc	Remove Epetra from doxygen config files	last month
packages	Ifpack2/Tpetra: fix #14529 and re-do of #14337 (#...	3 days ago

Primary repository for Project



Welcome to Trilinos -- Empowering Scientific Computing

Trilinos is an advanced software framework designed to facilitate the development of high-performance scientific applications. Trilinos provides a compreh libraries and tools that support a wide range of computational tasks, from linear algebra and optimization to differential equations and mesh generation.

With a focus on scalability and efficiency, Trilinos enables researchers and engineers to tackle complex problems across various fields, including engineeri applied mathematics. Our modular architecture allows users to customize and extend functionalities, ensuring that Trilinos meets the evolving needs of th community.

Join us in harnessing the power of Trilinos to drive innovation and discovery in computational science!

Quick Links

Explore Trilinos resources:

II. Trilinos Community

5 Contributing to Trilinos

6 Platforms for exchange among users or developers

7 Trilinos and HPSF

Contributing to Trilinos

Contributing to Trilinos

Just open a pull request on GitHub.

- Contributions through pull requests on GitHub
- Automated testing
  - ... of all downstream dependencies of every change
  - ... on different hardware and software platforms
- (Partial) enforcement of code formatting style
- Signed commits to enforce *Developer Certificate of Origin (DCO)*
- Useful resource: https://github.com/trilinos/Trilinos/wiki

Platforms for exchange among users or developers

Platforms for exchange among users or developers

Online resources:

- GitHub issues and discussion
- `#trilinos` channel in the Kokkos slack workspace

Events:

- TRILINOS User Group (TUG) meeting at HPSF Conference
- TRILINOS session at HPSF Community Summit (formerly known as EuroTUG)

Recent work – project support
• Project logo and website refreshes
AMReX
TRILINOS
VISKORES

HPSF Conference 2025

Trilinos and HPSF

Trilinos and HPSF

Commitment to open-source software

TRILINOS is an early member of HPSF to underline its commitment to open-source software for high-performance computing.

- GitHub for hosting and development of code repository
- Open to external contributions
- Current activities:
  - Transformation towards multi-institutional governance
  - Migrate CI/CD testing to non-Sandia hardware

HPSF
ESTABLISHED PROJECT

III. Recent Developments & Future Trends in Trilinos

8 Removal of the Epetra stack

9 Selected updates from packages

Removal of the Epetra stack

The Petra object model

Objective

Facilitate the construction and manipulation of distributed memory objects in parallel computing environments

Its core concept: Map

- **Distribution of global objects:** define partitioning of objects across the local memories of different MPI processes
- **Global-to-Local Index Translation:** translation between global indices (representing the entire dataset) and local indices (specific to a processes' subset), facilitating efficient data access and manipulation.

Types of maps:

- Row map: defines ownership
- Columns map: specifies required data access
- Range map: defines output of linear algebra operation
- Domain map: specifies input of linear algebra operation

Implementations of the Petra object model

Epetra:

- "Essential Petra"
- Since the beginning of TRILINOS
- Hard-coded to
  - 32 Bits, so limited to 2B elements per map
  - `double` and some `int`
  - No support for KOKKOS

Tpetra:

- "Templated Petra"
- Templated to
  - `Scalar` type
  - `LocalOrdinal` type
  - `GlobalOrdinal` type
  - Kokkos `Node` type

Xpetra:

- "Cross-over Petra"
- Hybrid between Epetra & Tpetra

Linear algebra stacks

Epetra

- Amesos
- AztecOO
- Epetra
- EpetraExt
- Ifpack
- Intrepid
- Isorropia
- ML
- New Package
- Pliris
- PyTrilinos
- ThyraEpetraAdapters
- ThyraEpetraExtAdapters
- Triutils
- ShyLU_DDCore

Tpetra

- Amesos2
- Anasazi
- Belos
- Galeri
- MueLu
- NOX
- PanzerDiscFE
- PanzerDofMgr
- Piro
- ROL
- ShyLU_DDFROSch,
- Teko
- TpetraCore
- TrilinosCouplings
- Stokhos
- Stratimikos
- Xpetra
- Zoltan2Core

Removal of Epetra stack

Motivation

- Epetra/Tpetra has been around for 20+ years
  - Many applications have been dependent on Epetra
- Driver for deprecation
  - Reduce costs of duplicative code
  - Reduce complexity of Trilinos DAG

Concrete steps

- Epetra stack has been removed on Nov 26, 2026
- Release v16.2: last release with Epetra stack
- Release v17.0: same state, but without Epetra stack

Impact on application codes

- Need to migrate to Tpetra, though this might be tricky and a lengthy process
- See other talks later today for some user stories

Selected updates from packages

Framework: software development

- Use pre-commit hooks to run clang-format and cmake-format
- New compiler support:
  - gcc@14
  - gcc@10.4.0+openmpi@4.1.6
  - cuda@12.4.1+gcc@10.4.0+openmpi@4.1.6
  - oneapi@2024.2+gcc@11.4
  - clang@19.1+openmpi@4.1.6
- Removed support for some older compilers:
  - gcc@8.3.0
  - gcc@10.3.0
  - clang@11.0.1
  - oneapi@2024.1

Framework: software delivery & deployment

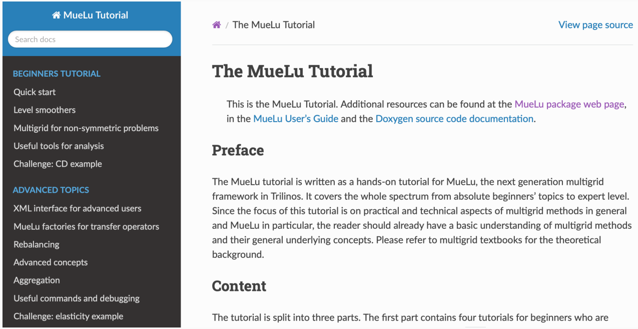
- Continue CMake & TriBITS builds (more for developers/power users)
- Provide/maintain a Spack recipe that others can use
- Support both delivery (Trilinos GitHub) and deployment (Spack)
- Create Spack recipes for each Trilinos package
  - Deduce package dependencies from CMake
  - Simplify configuration by the user (instead of common practice of writing their own Trilinos recipe)
  - Enable package developers to update their package-specific Spack recipe
- Maintain a Spack PR build (meaning a Spack build must pass to commit)

Ifpack2

- Improve performance of Chebyshev on GPU
- Added heuristic to Chebyshev to select performant kernel based on architecture

MueLu

- Refactoring and updates to CoalescDropFactory
- Overhaul of the MueLu tutorial
- MueLu scaling driver can rebalance initial user matrix



MueLu Tutorial
Search docs
BEGINNERS TUTORIAL
Quick start
Level smoothers
Multigrid for non-symmetric problems
Useful tools for analysis
Challenge: CD example
ADVANCED TOPICS
XML interface for advanced users
MueLu factories for transfer operators
Rebalancing
Advanced concepts
Aggregation
Useful commands and debugging
Challenge: elasticity example
/ The MueLu Tutorial
View page source

The MueLu Tutorial

This is the MueLu Tutorial. Additional resources can be found at the MueLu package web page, in the MueLu User's Guide and the Doxygen source code documentation.

Preface

The MueLu tutorial is written as a hands-on tutorial for MueLu, the next generation multigrid framework in Trilinos. It covers the whole spectrum from absolute beginners' topics to expert level. Since the focus of this tutorial is on practical and technical aspects of multigrid methods in general and MueLu in particular, the reader should already have a basic understanding of multigrid methods and their general underlying concepts. Please refer to multigrid textbooks for the theoretical background.

Content

The tutorial is split into three parts. The first part contains four tutorials for beginners who are

NOX

- Nox wrappers for Tpetra::MultiVector

Teko

- Improved conversion between Teko blocked operators and Tpetra matrices

Tpetra

- Improve many-to-few communication
- Properly enable communication routines from MPI Advance
- Update assembly examples to demonstrate current Kokkos practices
- Added several Epetra matrix transforms to Tpetra
  - SingletonFiltering: condenses out singleton rows and columns.
  - SolverMap: checks for missing indices wrt the local rows.
  - Reindex: reindex based on row map.
  - Rebalance: rebalance the distribution of linear problem contents (matrix, lhs, and rhs) among MPI nodes.

The end

Useful links:

- https://github.com/trilinos/Trilinos
- https://trilinos.github.io

Preprint: Mayr, M. *et al.* **Trilinos: Enabling Scientific Computing across Diverse Hardware Architectures at Scale.** *submitted for publication.* https://arxiv.org/abs/2503.08126 (2025)

Thank you.