ORNL/TM-2022/2622

Five years of ForTrilinos ECP

Seth R Johnson
Andrey Prokopenko

August 2022

OAK RIDGE National Laboratory

ORNL IS MANAGED BY UT–BATTELLE, LLC, FOR THE U.S. DEPARTMENT OF ENERGY

ORNL/TM-2022/2622

Computational Sciences and Engineering Division

FIVE YEARS OF FORTRILINOS ECP

Seth R Johnson
Andrey Prokopenko

Date Published: August 2022

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-Battelle, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

ABSTRACT

The ForTrilinos subproject of the Exascale Computing Project (ECP) [1] was initiated to bring the capabilities and scalability of the Trilinos numerical solver collection [2] to Fortran scientific application codes. A novel Fortran extension to the Simplified Wrapper and Interface Generator (SWIG) tool, which automatically generates Fortran bindings from existing C/C++ library code, has been applied to key Trilinos solver libraries to generate the new ForTrilinos libraries. SWIG-Fortran has additionally been used to generate new Fortran compatibility layers for additional scientific libraries and applications. This report summarizes the products and impact of the ForTrilinos subproject.

1. INTRODUCTION

The ForTrilinos subproject of the ECP [1] was initiated to bring the capabilities and scalability of the Trilinos numerical solver collection [2] to Fortran scientific application codes. It was envisioned as a replacement to an extant but dated set of ForTrilinos [3] bindings to the first-generation numeric libraries in Trilinos, including Epetra and AztecOO. The new ForTrilinos provides wrappers to the *second*-generation numeric solver libraries Tpetra, Belos, and NOX.

Several existing exascale-targeted scientific application codes such as Energy Exascale Earth System Model (E3SM) [4] and SCALE [5] used hand-rolled Fortran interface layers that required error-prone and tedious translation of data types and functions between C++ and Fortran. The ForTrilinos project developed a Fortran-targeted extension to the SWIG tool [6] to generate these interfaces automatically. Applied to Trilinos, SWIG generates object-oriented, standards-compliant Fortran 2003 interface code that mirrors the native Trilinos C++ interface.

Since its inception, the new SWIG Fortran extension has been applied to several other ECP codes, replacing manual Fortran interface layers with automated ones that require less maintenance by C++ library developers. Additionally, SWIG-Fortran has been used to provide additional bindings to library functions via the Flibcpp library [7], Formetis [8], and FlibHPC [9] libraries. Although SWIG-Fortran's capabilities have not been merged into the main SWIG repository, several minor improvements and fixes have been incorporated upstream, and the new Fortran capabilities are in an open "pull request," which when completed will improve the long-term maintainability and uptake of the code.

Finally, ForTrilinos and Fortran in general face challenges in adoption on exascale systems due to the difficulty of integration with heterogeneous architectures. In ECP, the number of codes using Fortran has decreased by half since the program's inception, with many transitioning to using C++ [10]. However, this transitional period has been an ideal time for the use of SWIG to couple new C++ components with the older Fortran components and provide a pathway for modernizing application codes.

2. PRODUCTS

The ForTrilinos ECP subproject has resulted in the release of several polished, well-documented, maintainable software artefacts. All of the products are maintained on GitHub and distributed under permissible open source licences.

2.1 FORTRILINOS VERSION 2

The primary goal of ForTrilinos has been accomplished by releasing the second version, which uses SWIG-Fortran to wrap the second-generation solver libraries of Trilinos. The following modules in ForTrilinos wrap corresponding Trilinos packages:

forerror provides exception handling support that defines an error integer and native string error messages.

forteuchos wraps Teuchos, which defines helper classes for processing XML files, managing memory, and communicators. It seamlessly translates "views" of data between C++ and native Fortran array pointers, and also transforms MPI communicators between Fortran and Trilinos.

fortpetra wraps Tpetra, the scalable linear algebra library. It includes support for sparse matrices and generic operators.

forbelos is for interfacing to the Belos solvers. Currently ForTrilinos only processes Belos type definitions for including in the high level module.

fortrilinos_hl is a high-level set of wrapper classes that provide access to parallel sparse eigensolvers and nonlinear solvers. They define Fortran class interfaces that allow inversion of control: callbacks *to* Fortran *from* Trilinos internal solver functions in C++.

The capabilities of ForTrilinos 2 are described in more detail in [11].

Because of C++ interface changes to Trilinos at each version, and smaller Fortran interface changes to SWIG at each version, there are many slightly different versions that are built to work with slightly different versions of SWIG and ForTrilinos. The version scheme is based on "semantic versioning" [12]:

- Major version numbers with Trilinos and minor versions of SWIG-Fortran (since it's still not officially upstreamed) can result in major version number changes for ForTrilinos.
- New features in Trilinos, and new support by ForTrilinos, can result in minor version number changes. Features removed or deprecated by a minor version change in Trilinos may also result in a minor version change.
- Minor changes to the SWIG-Fortran implementation (which don't affect the interface in the .F90 files) result in a patch version.

SWIG was designed for C++98 functionality and has difficulty parsing newer C++ constructs. Furthermore, because any language translation layer library requires preselected types, the increasingly templated nature of Trilinos and its new Kokkos backend are in conflict with the static nature of Fortran. The only way to dynamically instantiate C++ types from Fortran code would be to invent a meta-language that preprocesses the Fortran code to generate the appropriate C++wrapper layer at compile time. This is far outside the scope of our project.

Another approach to combating the increased dynamicism of Trilinos would be to expand the high-level library based on targeted application needs and typical instantiations of the Trilinos library.

ForTrilinos	Trilinos	SWIG
2.1.0	13.2	4.1.0-dev1+fortran
2.0.1	13.0:13.1	4.1.0-dev1+fortran
2.0.0	13.0:13.1	4.0.2+fortran
2.0.0-dev3	12.18.1	4.0.2+fortran
2.0.0-dev2	12.18.1	4.0.0+fortran+15e6ed59
2.0.0-dev1	12.17+8a82b322	4.0.0+fortran+15e6ed59
1.0	12.8.1	—

Table 1. ForTrilinos version compatibility. The +fortran suffix for SWIG indicates the SWIG-Fortran fork. A +sha extension refers to a specific Git commit that comes after the given version.

2.2 SWIG-FORTRAN

The new Fortran capability added to SWIG is described extensively in [6]. Since the publication of that article, additional enhancements and capabilities have been added to the SWIG-Fortran fork of SWIG, including support for complex numbers.

Over the past year, many ancillary code features used by SWIG-Fortran have been merged into the upstream SWIG repository. The addition of the Fortran module is still under review.

The standards-compliant code generated by SWIG-Fortran has been tested on most of the compilers in use by Oak Ridge Leadership Computing Facility (OLCF) machines, including GNU, IBM XL, CRAY, and PGI. The bindings have also been used in production code built with an Intel/MSVC toolchain on Windows.

2.3 ADDITIONAL LIBRARIES

Flibcpp is a set of Fortran bindings to the standard C++ library enabling well-tested and algorithmically optimal operations on potentially large arrays of native Fortran data. These can take the place of custom sorting and mapping that may work on test problems but be nonperformant at exascale. The capabilities and use of this new code are described in [7].

Flibhpc is a small library that demonstrates the use of HPC-targeted library features, including adapters for MPI communicators, non-copying conversion between CUDA Thrust data and Fortran OpenACC pointers, and adapters between CUDA-Fortran and CUDA.

Finally, Formetis is a demonstration library that generates thin wrappers to the METIS and ParMETIS [**parmetis**] C libraries using SWIG-Fortran.

3. IMPACT AND UPTAKE

ForTrilinos, SWIG, Flibcpp, and Formetis have defined packages in the Spack package manager [13] for easy distribution and installation. The E4S collection [14], which is based on Spack, includes ForTrilinos and the Fortran fork of SWIG.

Although Spack does not collect analytics data about software installations, we can gauge interest in the different ForTrilinos products with GitHub's star feature. (TODO: add star plots for the various fortrilinos libs, compare to spack/trilinos?)

3.1 FORTRILINOS

At the outset of this endeavor, several projects expressed interest in replacing their custom Fortran wrappers with official Trilinos wrappers. Unfortunately, only one project, E3SM, ended up testing ForTrilinos integration. The E3SM project is testing the replacement of a custom Fortran-based CG solver and preconditioners with the Belos Trilinos CG solver and a variety of Trilinos-based preconditioners [15]. Unfortunately, as of this writing, the Trilinos solvers have scalability problems and solver performance degradation compared to the in-house Fortran preconditioner and CG solver. Profiling has demonstrated that this is due to characteristics of the solvers themselves rather than any of the Fortran-C++ translation layer.

3.2 SWIG-FORTRAN

Despite the limited uptake of ForTrilinos, the SWIG-Fortran product has been integrated into numerous libraries both inside and outside ECP. It is currently in use by SUNDIALS [16], STRUMPACK [17], TASMANIAN [18], and DTK [19], either as a supplement to their C/C++-only library or replacing previous handwritten code, lowering the maintenance burden on library developers. SWIG-Fortran is also being used by the SCALE project [5], which shares code with the ExaSMR ECP application, to help transition away from Fortran development by integrating new C++ code into older Fortran code. Additional applications outside of ECP have made inquiries to integrate SWIG-Fortran for coupling its Fortran and C++ components.

4. CONCLUSIONS

The exascale application landscape has changed substantially over the course of the ECP. ForTrilinos finds itself among a shrinking pool of Fortran applications, none of which have decided to capitalize on its advanced capabilities. However, the SWIG-Fortran technology underpinning ForTrilinos has proven itself useful many times over to relieve the maintenance costs of manual Fortran binding for scientific libraries. It also has great potential for helping application codes in Fortran transition to more maintainable and exascale-friendly C++.

5. BIBLIOGRAPHY AND REFERENCES CITED

[1] *The Exascale Computing Project, Addressing a national imperative: Application Development Update*, Oak Ridge, TN, Sep. 2019. [Online]. Available: www.exascaleproject.org.

[2] M. A. Heroux and J. M. Willenbring, "A new overview of the trilinos project," *Scientific Programming*, vol. 20, no. 2, pp. 83–88, 2012.

[3] K. Morris, D. W. Rouson, M. N. Lemaster, and S. Filippone, "Exploring Capabilities within ForTrilinos by Solving the 3D Burgers Equation," *Scientific Programming*, vol. 20, no. 3, pp. 275–292, 2012. DOI: 10.1155/2012/378791. [Online]. Available: http://www.hindawi.com/journals/sp/2012/378791/ (visited on 02/04/2019).

[4] J.-C. Golaz, P. M. Caldwell, L. P. Van Roekel, M. R. Petersen, Q. Tang, J. D. Wolfe, *et al.*, "The doe e3sm coupled model version 1: Overview and evaluation at standard resolution," *Journal of Advances in Modeling Earth Systems*, vol. 11, no. 7, pp. 2089–2129, 2019.

[5] W. A. Wieselquist, R. A. Lefebvre, and M. A. Jessee, "Scale code system," Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), Tech. Rep. ORNL/TM-2005/39, 2020.

[6] S. R. Johnson, A. Prokopenko, and K. J. Evans, "Automated Fortran-C++ Bindings for Large-Scale Scientific Applications," *Computing in Science & Engineering*, vol. 22, no. 5, pp. 84–94, Oct. 2020. DOI: 10.1109/MCSE.2019.2924204. [Online]. Available: https://ieeexplore.ieee.org/document/8745480/ (visited on 08/20/2019).

[7] S. R. Johnson, "Flibcpp user manual," Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), Tech. Rep. ORNL/TM-2021/2041, 2021.

[8] ——, *Formetis*, 2021. [Online]. Available: https://github.com/swig-fortran/formetis.

[9] ——, *FlibHPC*, 2021. [Online]. Available: https://github.com/swig-fortran/flibhpc.

[10] T. M. Evans, A. Siegel, E. W. Draeger, J. Deslippe, M. M. Francois, T. C. Germann, *et al.*, "A survey of software implementations used by application codes in the exascale computing project," *The International Journal of High Performance Computing Applications*, vol. 36, no. 1, pp. 5–12, 2022.

[11] A. V. Prokopenko, S. R. Johnson, and K. J. Evans, "ForTrilinos 2: Idiomatic Fortran bindings to Trilinos," Oak Ridge National Laboratory, Tech. Rep. ORNL/TM-2021/2042, 2021, (pending).

[12] T. Preston-Werner, *Semantic Versioning 2.0.0*, 2021. [Online]. Available: https://semver.org.

[13] T. Gamblin, M. LeGendre, M. R. Collette, G. L. Lee, A. Moody, B. R. de Supinski, *et al.*, "The Spack Package Manager: Bringing Order to HPC Software Chaos," ser. Supercomputing 2015 (SC15), LLNL-CONF-669890, Austin, Texas, USA, Nov. 2015. DOI: 10.1145/2807591.2807623. [Online]. Available: https://github.com/spack/spack.

[14] L. C. McInnes, M. A. Heroux, E. W. Draeger, A. Siegel, S. Coghlan, and K. Antypas, "How community software ecosystems can unlock the potential of exascale computing," *Nature Computational Science*, vol. 1, no. 2, pp. 92–94, Feb. 2021. DOI: 10.1038/s43588-021-00033-y. [Online]. Available: http://www.nature.com/articles/s43588-021-00033-y (visited on 05/22/2021).

[15] H.-G. Kang, K. J. Evans, S. R. Johnson, A. Prokopenko, A. G. Salinger, and R. S. Tuminaro, "An implicit barotropic mode solver for mpas-ocean using a modern fortran solver interface."

[16] D. J. Gardner, D. R. Reynolds, C. S. Woodward, and C. J. Balos, "Enabling new flexibility in the sundials suite of nonlinear and differential/algebraic equation solvers," *ACM Transactions on Mathematical Software (TOMS)*, 2020.

[17] P. Ghysels, X. S. Li, C. Gorman, and F.-H. Rouet, "Strumpack: Scalable preconditioning using low-rank approximations and random sampling," ser. Supercomputing 2016 (SC16).

[18] M. Stoyanov, "User manual: Tasmanian sparse grids," Oak Ridge National Laboratory, Tech. Rep. ORNL/TM-2015/596, 2015.

[19] S. Slattery, P. Wilson, and R. Pawlowski, "The data transfer kit: A geometric rendezvous-based tool for multiphysics data transfer," in *International conference on mathematics & computational methods applied to nuclear science & engineering (M&C 2013)*, 2013, pp. 5–9.

ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.