High-level User Interfaces for the DOE ACTS Collection

L.A. Drummond[1], Vicente Galiano[2], Violeta Migallón[3], and José Penadés[3]

[1] Lawrence Berkeley National Laboratory,
One Cyclotron Road, Berkeley CA 94703, United States of America
LADrummond@lbl.gov
[2] Departamento de Física y Arquitectura de Computadores,
Universidad Miguel Hernández, 03202 Elche, Alicante, Spain
vgaliano@umh.es
[3] Departamento de Ciencia de la Computación e Inteligencia Artificial,
Universidad de Alicante, E-03071 Alicante, Spain
{violeta, jpenades}@dccia.ua.es

Abstract. The ACTS collection project comprises a set of state of the art software tools to speed up the development of High-Performance Computing Applications in science and engineering. Here we look at the development of High Level user interfaces using scripting languages like Python, to facilitate the access to ACTS technology to a wide community of computational scientists. PyACTS is our main project here, but we also visit other efforts between the community of developers of ACTS tools.

1 Introduction to the ACTS Collection and PyACTS

The Advanced CompuTational Software (ACTS) [1] Collection comprises a set of computational tools developed primarily at DOE laboratories, sometimes in collaboration with universities and other funding agencies (NSF, DARPA), aimed at simplifying the solution of common and important computational problems. A number of important scientific problems have been successfully studied and solved by means computer simulations built on top of tools available in the ACTS Collection [2]. The ACTS Collection brings robust and high-end software tools to the hands of application developers to accelerate the development of computational science codes and consequent results. However, this transfer of technology is not always successful due in part to the intricacy in understanding the interfaces associated with the software tools and the time an application scientists spends installing and learning the use of a given tool. PyACTS [3–5] provides a didactical user interface to assist with their first application prototype and following production code development. Here we look at the PyACTS development project and existing functionalities.

We begin with brief descriptions of the ACTS tools for which have been building a PyACTS, and also other ACTS tools that already provide Python Interfaces. The reader is refer to the ACTS Information Center [6] for more details on these tools and others available in the collection.

2 Some of The Tools in the ACTS Collection

ScaLAPACK [7] is a library of high-performance linear algebra routines for distributed-memory message-passing Multiple Instruction Multiple Data (MIMD) computers and networks of workstations. The ScaLAPACK library contains routines for solving systems of linear equations, least squares, eigenvalue problems and singular value problems. It also contains routines that handle many computations related to those, such as matrix factorizations or estimation of condition numbers.

SuperLU [8] is a general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines. The library is written in C and is callable from either C or Fortran. The library routines perform an LU decomposition with numerical pivoting and triangular system solves through forward and back substitution. The LU factorization routines can handle non-square matrices but the triangular solves are performed only for square matrices.

PETSc The **P**ortable, **E**xtensible **T**oolkit for **S**cientific **c**omputation [9], provides sets of tools for the parallel, as well as serial, numerical solution of PDEs that require solving large-scale, sparse linear and nonlinear systems of equations. PETSc includes nonlinear and linear equation solvers that employ a variety of Newton techniques and Krylov subspace methods. PETSc provides several parallel sparse matrix formats, including compressed row, block compressed row, and block diagonal storage.

SUNDIALS (**SU**ite of **N**onlinear and **DI**fferential/**AL**gebraic equation **S**olvers) refers to a family of four closely related solvers; CVODE [10, 11], for systems of ordinary differential equations; CVODES [12], variant of CVODE for sensitivity analysis; KINSOL [13], for systems of nonlinear algebraic equations; and IDA [14], for systems of differential-algebraic equations.

These solvers have some code modules in common, primarily a module of vector kernels and generic linear system solvers, including one based on a Scaled Preconditioned GMRES method. All of the solvers are suitable for either serial or parallel environments. All message passing calls are made through MPI.

Trilinos [15] is a framework for the development of parallel solvers and libraries within an object-oriented environment. AztecOO is one of the libraries available in Trilinos and it is part of the ACTS Collection. The Trilinos framework offers a variety of mechanisms for a software package to interact with other software packages. Trilinos includes a large set of functional libraries and packages that including AztecOO they provide numerical functionalities and support for the solution of large-scale, complex multi-physics engineering and scientific applications.

TAU - Tuning and Analysis Utilities [16] is a toolkit for performance analysis of parallel programs written in C, C++, Java and Python. The main advantage of this toolkit is that it is portable to many computer platforms offering portability when tracing the performance of given code. TAU basically gathers performance information through manual or automatic instrumentation of functions, basic coding blocks, methods, statements, or full programs. The au-

tomatic instrumentation is implemented via an automatic instrumentation generator called Program Database Toolkit (PDT). TAU provides a visualization tool to view the outpput from the traces. It can also generate tracer data to be used other third party visualization programs like Vampir, Paraver or JumShot.

3 PyACTS: A Python Interface to The ACTS Collection

Python [17] is an interpreted, interactive, object-oriented programming language. Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. New built-in modules are easily written in C or C++. Python is also usable as an extension language for applications that need a programmable interface. Python is designed to make integration with other software components in a system as simple as possible. Programs written in Python can be easily blended with other languages. For instance, Python scripts can call out existing C and C++ libraries, Java classes, and much more. Actually, it is this feature of Python that is employed in our current work.

Additionally, Python is portable: it runs on many brands of UNIX, on Windows, Mac, and many other platforms. Python is copyrighted but freely usable and distributable, even for commercial use. Python is an ideal language for prototype development and other ad–hoc programming tasks, without compromising maintainability and it uses an elegant syntax for readable programs. All of the ACTS tools listed in the previous section use MPI as one of the methods for supporting message passing. In the PyACTS, we use PyMPI, which enables us to use the same Python modules and rich functionality.

Currently, we have developed an interface to ScaLAPACK and SuperLU, PyScaLAPACK [3] and PySuperLU, respectively, and have used a design implementation of PyACTS as shown in Figure 1. This design allows for easily handling of different versions of the same package and also the interoperability with other Python interfaces from other ACTS tool developers. For instance, PETSc and SUNDIALS provide their own python extensions. Trilinos provides PyTrilinos [18] which uses Epetra, Trilinos basic data object class, extensions to provide access to the full Trilinos functionality. TAU can also profile programs written in Python. Thus, the python structure depicted in Figure 1, still allows for integration of existing PyACTS functionality with the ones being developed by other ACTS tool developers.

4 Conclusions and Future Work

Although the overhead of the interface in the examples run is not significant, PyACTS is not yet intended for large production runs in high-end system, rather it is a didactical tool for generating a first prototype of the application code. It helps the user to become familiar with a particular interface and also access in an interoperable manner other ACTS tools interface without having to learn it.

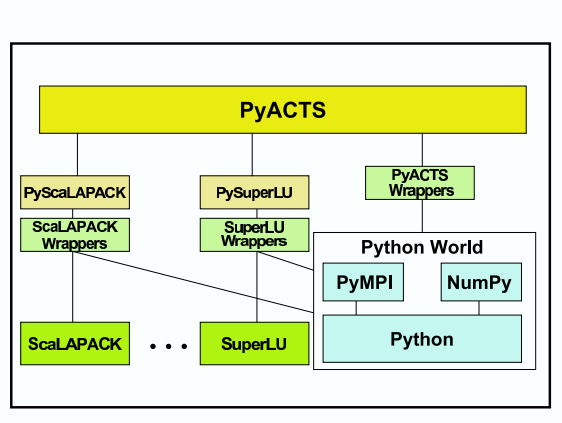

Fig. 1. Design of the PyACTS Interface.

We are currently working on a PyACTS *scribe* that allow to write out the fortran and C language equivalent functions of the High-Level PyACTS routines. Therefore, a user that prototypes an application using PyACTS will be able to get the exact fortran or C calling interface sequence in order to produced a code that can be compiled and use for production runs in a large number of system. In the future, we will be working closely with other ACTS tool developers and integrating more functionality to PyACTS.

References

1. Drummond, L.A., Marques, O.A.: An overview of the Advanced CompuTational Software (ACTS) Collection. ACM Transactions on Mathematical Software **31** (2005) 282–301
2. Drummond, L., Hernandez, V., Marques, O., Roman, J., Vidal, V.: A Survey of High-Quality Computational Libraries and Their Impact in Science and Engineering Applications. In: Lecture Notes in Computer Science. Volume 3403., Valencia, Spain, Springer-Verlag (2005) 37–50
3. Galiano, V., Drummond, L.A., Migallón, V., Penadés, J.: High Level User Interfaces for High Performance Libraries in Linear Algebra: PyBLACS and PyPBLAS. In: Proceedings from 12th International Linear Algebra Society Conference, University of Regina,Regina, Saskatchewan, Canada (2005)
4. Drummond, L.A., Galiano, V., Migallón, V., Penadés, J.: Improving ease of use in BLACS and PBLAS with Python. In: Proceedings from Parallel Computing 2005 (ParCo 2005), Malaga, Spain (2005)
5. Kang, N., Drummond, L.A.: A first prototype of PyACTS. Technical Report LBNL-53849, Lawrence Berkeley National Laboratory (2003)
6. Marques, O.A., Drummond, L.A.: The ACTS Information Center. http://acts.nersc.gov (2001)

7. Blackford, L.S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J.W., Dhillon, I., Dongarra, J.J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R.C.: ScaLAPACK User's Guide. SIAM, Philadelphia, Pennsylvania (1997)
8. Demmel, J.W., Gilbert, J.R., Li, X.: SuperLU User's Guide. University of California, Berkeley. (2003)
9. Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F.: Efficient management of parallelism in object oriented numerical software libraries. In Arge, E., Bruaset, A.M., Langtangen, H.P., eds.: Modern Software Tools in Scientific Computing, Birkhauser Press (1997) 163–202
10. Cohen, S.D., Hindmarsh, A.C.: CVODE User Guide. Technical Report UCRL-MA-118618, Lawrence Livermore National Laboratory (1994)
11. Byrne, G.D., Hindmarsh, A.C.: User documentation for PVODE, an ODE solver for parallel computers. Technical Report UCRL-ID-130884, Lawrence Livermore National Laboratory (1998)
12. Hindmarsh, A.C., Serban, R.: User Documentation for CVODES, An ODE Solver with Sensitivity Analysis Capabilities. Technical Report UCRL-MA-148813, Lawrence Livermore National Laboratory (2002)
13. Taylor, A.G., Hindmarsh, A.C.: User Documentation for KINSOL, A nonlinear solver for sequential and parallel computers. Technical Report UCRL-ID-131185, Lawrence Livermore National Laboratory (1998)
14. Hindmarsh, A.C., Taylor, A.G.: User Documentation for IDA, a Differential-Algebraic Equation Solver for Sequential and Parallel Computers. Technical Report UCRL-MA-136910, Lawrence Livermore National Laboratory (1999)
15. Heroux, M.A., Bartlett, R.A., Howle, V.E., Hoekstra, R.J., Hu, J.J., Kolda, T.G., Lehoucq, R.B., Long, K.R., Pawlowski, R.P., Phipps, E.T., Salinger, A.G., Thornquist, H.K., Tuminaro, R.S., Willenbring, J.M., Williams, A., Stanley, K.S.: An Overview of the Trilinos Project. ACM TOMS **V** (2004) 1–27
16. Malony, A., Shende, S., Trebo, N., Ray, J., Armstrong, R., Rasmussen, C., Sottile, M.: Performance Technology for Parallel and Distributed Component Software. Concurrency and Computation: Practice and Experience **17** (2005) 117–141
17. G. van Rossum, F.D.J.: An Introduction to Python. Network Theory Ltd (2003)
18. M.Sala: Distributed Sparse Linear Algebra with PyTrilinos. Technical Report SAND2005-3835, Sandia National Laboratories (2005)