Evaluation of High-Performance Computing Software

Shirley Browne* Jack Dongarra† Tom Rowan‡

Abstract

The absence of unbiased and up to date comparative evaluations of high-performance computing software complicates a user's search for the appropriate software package. The National HPCC Software Exchange (NHSE) is attacking this problem using an approach that includes independent evaluations of software, incorporation of author and user feedback into the evaluations, and Web access to the evaluations. We are applying this approach to the Parallel Tools Library (PTLIB), a new software repository for parallel systems software and tools, and HPC-Netlib, a high performance branch of the Netlib mathematical software repository. Updating the evaluations with feedback and making it available via the Web helps ensure accuracy and timeliness, and using independent reviewers produces unbiased comparative evaluations difficult to find elsewhere.

1 Introduction

Selecting the appropriate software for a high-performance computing task is difficult. Packages differ in capabilities, features, and quality. Comparative evaluations, when they are available, usually come from the author of one of the packages. As a result, comprehensive, independent, and unbiased evaluations are not normally readily available, despite the obvious value such information would be to users.

The National HPCC Software Exchange (NHSE) [1, 2] a Center for Research on Parallel Computation (CRPC) project for the collection, distribution, and evaluation of software and information produced by HPCC programs, is currently undertaking comparative evaluations of high-performance computing software with a view to satisfying this need. Our goal is to provide independent, unbiased comparative evaluations of HPC software of wide applicability. Users get easy access to side-by-side comparative evaluations based on consistent and objective criteria.

Our current evaluation focus is on the Parallel Tools Library (PTLIB), a new software repository for parallel systems software and tools, and HPC-Netlib, a high performance branch of the Netlib [3, 4] mathematical software repository. We refine the NHSE high-level evaluation framework to the domains in these two areas and for each package in a particular domain, we apply a consistent set of criteria to assess various characteristics of the software. The evaluations, as well as author and user feedback, are made available via the Web.

Although the software evaluation part of NHSE activities is still in the early stages, many packages have already been evaluated. However, the evaluations will be on ongoing task. Our evaluation criteria and procedures will evolve as the software pool grows and as we gather comments from software authors and users.

Section 2 describes our approach to evaluating high-performance computing software in more detail. Sections 3 and 4 describe the evaluation criteria and current status for our evaluations of software in PTLIB and HPC-Netlib, and Section 5 summarizes our results.

2 Approach

Our approach differs in several respects from the traditional presentations of comparative evaluations.

- Comparative evaluations currently available are typically done by an author of one of the packages and can be subject to bias and possible inconsistencies across evaluations. By performing our evaluations as consistently and objectively as possible, we should be able to avoid even the appearance of bias.

*Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301 (*browne@cs.utk.edu*)

†Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301 and Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN 37821-6367 (*dongarra@cs.utk.edu*)

‡Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN 37821-6367 and Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301 (*rowan@cs.utk.edu*) Author to whom correspondence should be directed.

- We incorporate feedback from package authors and users into our evaluation. This ensures that the evaluations are both fair and up to date.
- Our evaluations are not static. As additional information is gathered, either through our author/user feedback mechanism or through enhancements to our evaluation procedures, we will update the evaluations.
- The collection of evaluations will be easily accessible at a centralized location via the Web. Users can do side-by-side comparisons according to selected characteristics.

We decided that users would benefit most if we concentrated our evaluations on the software with broadest applicability. For this reason we have focused our evaluations on parallel systems software and tools, and on mathematical software. Many packages selected for evaluation were drawn from the collection of software already available through Netlib and the NHSE. We also solicited other promising packages not yet available from our repositories.

Our first step in designing a systematic, well-defined evaluation criteria was to use a high-level set of criteria that can be refined as needed to particular domains. Our starting point for establishing the high-level set of criteria was to build on the software requirements described in the Baseline Development Environment [5]. The criteria were appropriately tailored to a particular domain by those doing the evaluations and by others with expertise in the domain. We expect that the evaluation criteria for a given domain will evolve over time as we take advantage of author and user feedback, and as new evaluation resources such as new tools and problem sets become available.

The NHSE software evaluation process consists of the following steps.

1. Reviewers and other domain experts refine the high-level evaluation criteria to this domain.
2. We select software packages within this domain and assign each to an NHSE project member knowledgeable in the field for evaluation.
3. The reviewer evaluates the software package systematically, typically using a well-defined evaluation criteria checklist. Assessment of certain criteria will necessarily be subjective. To facilitate comparisons, the reviewer assigns a numerical score for each of those criteria based on his judgment of how well the criterion was met. Assessment of criteria that can be easily measured are typically reported directly as those measured results.
4. We solicit feedback from the package author, giving him the opportunity to make corrections, additions, or comments on the evaluation. In effect we ask him to review our review, and we revise the review to correct any errors or omissions.
5. We make the review and the author's feedback available via the Web.
6. We add to the evaluation and author feedback any comments users wish to submit through the NHSE Web pages.

3 Evaluation of PTLIB Software

So far our evaluation of PTLIB software has covered parallel debuggers and performance analyzers. We give a detailed description of the evaluation criteria below. Note that it is has been refined and expanded to a level of detail to enable it to serve as an evaluation checklist.

Performance Includes accuracy, efficiency, and scalability.

- **Accuracy** A performance monitoring tool is accurate if it does not cause too great a change in the behavior and timing of the program it is monitoring.
- **Efficiency** The software runs fast enough, in that slow speed does not make it an ineffective tool.
- **Scalability** A parallel tool is scalable if its overhead grows in a reasonable manner with increases in system and problem sizes. In some cases, linear growth may not be acceptable.

Capabilities The tool has adequate functionality to effectively accomplish its intended tasks.

Versatility Includes heterogeneity, interoperability, portability, and extensibility

- **Heterogeneity** A heterogeneous tool can simultaneously be invoked on and/or have its components running on all platforms in a heterogeneous system.
- **Interoperability** A parallel tool is interoperable if its design is based on open interfaces and if it conforms to applicable standards.

Portability A parallel tool is portable if it works on different parallel platforms and if platform dependencies have been isolated to specific parts of the code.

Extensibility A performance analysis tool is extensible if new analysis methods and views can be added easily.

Maturity Includes robustness, level of support, and size of user base.

Robustness A parallel tool is robust if it handles error conditions without crashing and by reporting them and recovering from them appropriately.

Level of support The timeliness and quality of responses to questions from users or the reviewer should be adequate for typical package use.

Size of user base Indicators include the existence of newsgroups or mailing lists for the package, and the number of downloads of the package.

Ease of use The software has an understandable user interface and is easy to use for a typical NHSE user.

The software characteristics described in the criteria above are most appropriately assessed by reviewer judgment rather than by measured results. Each PTLIB software evaluation therefore contains a set of reviewer-assigned numerical scores indicating how well the package met the criteria.

Currently over 20 parallel debuggers and performance analyzers have been evaluated according to the above criteria. These packages include AIMS, DAQV, LCB, MQM, NTV, Pablo, Pangaea, Paradyn, ParaGraph, ParaVision, PGPVM, PVaniM, TotalView, Upshot, VAMPIR, VT, Xmdb, XMPI, and XPVM. We have solicited author feedback on these evaluations, and the initial evaluations have been updated based on the feedback received. Web access to the evaluations is available through the PTLIB homepage at *http://www.nhse.org/ptlib/*. See *http://www.nhse.org/sw_catalog/* for descriptions of the PTLIB software packages.

4 Evaluation of HPC-Netlib Software

Just as we selected high-performance mathematical software for evaluation because of its broad applicability for users, we have given priority to three mathematical software target domains for the same reason.

- Linear algebra, especially sparse linear system solvers
- Partial differential equations (PDEs)
- Optimization

Several issues need to be considered when establishing evaluation criteria for mathematical software. One observation is that, in contrast to the evaluation of parallel tools, the evaluation of mathematical software is inherently more quantitative. Assessing software by assigning scores, as was done for the evaluation of parallel tools, would be inappropriate for the evaluation of mathematical software.

Another consideration is that mathematical software packages often have different aims and different target applications. We must ensure that systematically and consistently checking the same criteria across all packages does not lead to comparing apples and oranges.

Another important observation is that some goals of evaluation are inherently conflicting. Satisfying a wish list of ideal goals is impossible, and tradeoffs will be necessary. Consider the following desirable and reasonable evaluation goals:

- consistency in evaluation procedures because it promotes objectivity and fairness in the evaluations, and
- tailoring evaluation procedures to packages because it promotes appropriate testing and optimal use of evaluation resources.

Now consider the following scenario. Package A is well established, widely known to be thoroughly tested, and the package authors are known to the reviewer. In contrast, everything about Package B is unknown to the reviewer. It clearly would be appropriate to run Package B through a battery of various simple tests to ensure it meets at least some minimal standards. Running the same tests on Package A might seem inappropriate because the package has clearly survived far more rigorous testing. Running the tests does not appear to offer much added value to the user and does not appear to be the best use of the reviewer's time. However, *not* running the same tests on both packages could lead to a double standard or the appearance of a double standard. A satisfactory resolution of this scenario will require some tradeoffs between the conflicting goals.

Our basic approach for meeting the conflicting goals is to test the packages on a relatively small set of standard test problems. The problem set will include problems with a wide range of difficulty levels, easy problems any package should be able to solve and extremely difficult problems that will test the packages' limits. Problems will also be selected to test special claims made by package authors. Problem sets will necessarily vary somewhat from package to package, but our aim is to have some small common core of test problems across similar packages so that users will have a basis for side-by-side comparison. Any other tests tailored to particular packages would be extra and optional.

Evaluation results will be presented as a reconfigurable package/problem Web-accessible table, with each cell of the table containing the results of that particular test. We expect the problem set used in our evaluations to evolve over time. We plan to update the tests and the results table when the problem set changes to ensure a continuing common basis for package comparison.

Characteristics of mathematical software can be divided into two categories - those characteristics that can be assessed by inspection of the code and documentation and those that can only be assessed through actual testing.

Ideally the software testing examines the following characteristics.

Correctness The code works correctly on the intended problems.

Efficiency The code is efficient with respect to both speed and storage.

Stability The code is stable, performing as efficiently and as accurately as the problem's conditioning allows.

Robustness The code handles error conditions reasonably. The ability to estimate a problem's condition, or otherwise providing a check on the computed answer's reliability, is also desirable.

Full examination of each characteristic for each package is clearly unrealistic. In addition, absolute quantitative assessments of the characteristics may mean little to a typical package user. Our approach of doing side-by-side comparisons on common standard problems provides *relative* assessments that are both more practical to obtain and more helpful to the user.

For testing sparse linear system solvers, several useful resources are available. The Harwell/Boeing [6] collection of sparse test matrices will be the source for many of our test problems. SPARSKIT [7] also contains a useful collection of test problems and in addition provides matrix generation and matrix format conversion utilities. The Harwell/Boeing and SPARSKIT collections are available through the Matrix Market [8].

The evaluation characteristics of sparse solvers that can be assessed largely from inspection of the code and its documentation include the following.

Capabilities Includes methods, formats

- **Methods** Identify which methods and preconditioners are used in the package.
- **Formats** Identify which matrix formats are supported. Packages that use non-standard matrix formats may be harder to test and to use, and will tend to have a relatively small base of users.

Portability Includes standards, architectures

- **Standards** Identify which standards (e.g. MPI, BLAS) are used.
- **Architectures** Identify on which architectures the packages has been tested and is supported.

Versatility Includes methods, interfaces

- **Methods** Identify the extent to which a user can design or specify the method or preconditioner to be used.
- **Interfaces** Identify how well the package interfaces with other packages, and whether it has multi-language support.

Ease of use Identify adequacy of documentation, examples, and support.

Our current emphasis in the HPC-Netlib evaluation is on sparse linear system solvers, although many of the sparse packages also fall into the PDE category. We are currently evaluating the iterative packages Aztec, PETSc, and PIM. We also plan to evaluate the iterative packages BlockSolve95, BPKIT, Elegant, IML++, ITPACK, LASPack, PARPRE, PCG, P-SPARSLIB, and Templates, and the direct packages CAPSS, SPARSE, SuperLU, and UMFPACK. The evaluations are available through the HPC-Netlib homepage at *http://www.nhse.org/hpc-netlib/*. See *http://www.nhse.org/sw_catalog/* for descriptions of the HPC-Netlib software packages.

5 Summary

Evaluating software accurately and in a way that is both useful to users and fair to authors is difficult and time consuming. However, there are many benefits to users from such an effort. Some of these are a direct consequence of our approach. The evaluations are easily accessible via the Web. Our mechanism of incorporating feedback from authors and users helps ensure accuracy in the evaluation and keeps it up to date. Independent reviewers systematically evaluating software against well thought out criteria will produce an objective, unbiased comparative evaluation difficult to find elsewhere.

Acknowledgments

This project has benefited greatly from the efforts of many people. We thank Vasilios Alexiades, Chris Hastings, Christian Halloy, and Kevin London for evaluating software and for helping to establish the evaluation criteria, Paul McMahan for his invaluable systems support, and Ron Boisvert and Esmond Ng for many valuable discussions.

References

[1] Shirley Browne, Jack Dongarra, Stan Green, Keith Moore, Tom Rowan, Reed Wade, Geoffrey Fox, Ken Hawick, Ken Kennedy, Jim Pool, Rick Stevens, Bob Olsen, and Terry Disz, "The National HPCC Software Exchange", *IEEE Computational Science and Engineering*, vol. 2, pp. 62–69, 1995, Project Web page at http://www.nhse.org/.

[2] Shirley Browne, Henri Casanova, and Jack Dongarra, "Providing access to high performance computing technologies", *in Proceedings of the PARA96 Workshop on Applied Parallel Computing in Industrial Problems and Optimization*, Lyngby, Denmark, August 1996.

[3] J. Dongarra and E. Grosse, "Distribution of mathematical software via electronic mail", *Communications of the ACM*, vol. 30, pp. 403–407, May 1987, Project Web page at http://www.netlib.org/.

[4] Shirley Browne, Jack Dongarra, Eric Grosse, and Tom Rowan, "The Netlib mathematical software repository", *D-Lib Magazine*, Sep. 1995, http://www.dlib.org/magazine.html.

[5] C. M. Pancake, "Specification of baseline development environment", http://www.cs.orst.edu/~pancake/SSTguidelines/baseline.html.

[6] I. S. Duff, R. G. Grimes, and J. G. Lewis, "Sparse matrix test problems", *ACM Transactions on Mathematical Software*, vol. 15, pp. 1–14, 1989.

[7] Y. Saad, "SPARSKIT: A basic tool kit for sparse matrix computations", Technical Report 90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990.

[8] R. F. Boisvert, R. Pozo, K. Remington, R. F. Barrett, and J. J. Dongarra, "Matrix Market: A web resource for test matrix collections", in R.F. Boisvert, editor, *The Quality of Numerical Software: Assessment and Enhancement.* Chapman and Hall, London, 1997.