第１回先進スーパーコンピューティング環境研究会（ASE 研究会）発表資料

ASE 研究会幹事　特任准教授　片桐孝洋

2008 年 3 月 3 日（月）13 時から 14 時 30 分まで、米国ローレンス・バークレー国立研究所の Osni Marques 博士をお招きして、第 1 回先進スーパーコンピューティング環境研究会（ASE 研究会）が開催されました。本号では、Marques 博士が発表した内容を資料としてまとめていただき、掲載させていただきます。（当日の配布資料は、スーパーコンピューティングニュース Vol.10, No.2, pp.20-40 に掲載済み。）

Marques 博士の講演は、「An Overview of the ACTS Collection Project」と題される発表でした。ACTS Collection Project とは、米国エネルギー省（US Department of Energy, DOE）が資金を提供し、米国のスーパーコンピュータ上で利用できる計算科学アプリケーションの開発、保守、および移植の際の負荷を軽減する、先進的な計算科学ソフトウエア（Advanced CompuTational Software、ACTS）群に関するプロジェクトです。ACTS 群は、DOE で開発したソフトウエアに限らず、他のファンドで開発されたソフトウエアとも連携して構成されており、計算科学アプリケーション開発時における高性能コード作成を容易にすることが目的です。

日本では、ソフトウエア開発自体に競争的資金は出しても、保守管理を目的に資金を出すことはほとんどありません。ACTS のような保守管理も目的にしているプロジェクトは、米国でも稀であるようです。いずれにせよ、当センターのようなスーパーコンピュータ提供機関にとって、スーパーコンピュータの利用を容易にするソフトウエアの開発のみならず、有益なソフトウエアの保守管理はきわめて重要な業務であり、米国の先例として大変参考になる講演でした。

具体的に ACTS プロジェクトでは、著名な数値計算ライブラリ ScaLAPACK、PETSc、および SuperLU などの提供だけを目的にしているのではなく、TAU (Tuning Analysis Utilities) のような性能解析ツールや、利用に関してノウハウの塊である数値計算ライブラリにおいて適するライブラリや適する性能パラメタを推薦したりするツールの研究開発を行っています。また、長期保守管理サービス、テストと評価環境サービス、アウトリーチと普及サービス、高度なユーザサポートサービスを指向して ACTS ツール群を活用するサービス指針 ACTS Center が示されています。このようなサービス指針は、ユーザを京速コンピューティングへと導くために日本のスーパーコンピュータ提供機関においても参考とすべきサービス指針といえるでしょう。

ASE 研究会では、ACTS プロジェクトのように先進的なスーパーコンピューティング環境を実現するための話題提供を目的に、年数回の活動を計画しております。これからもご支援のほどを、よろしくお願い申し上げます。

An Overview of the ACTS Collection Project[†]

Osni Marques
Lawrence Berkeley National Laboratory
1 Cyclotron Road, MS 50F-1650
Berkeley, CA 94720-8139

Abstract

The development of high performance application codes is an expensive process that often requires specialized support about the available computational resources and also the software tools that allow for an efficient use of those resources. Usually, the development effort is increased by the complexity of the phenomena that can be addressed by numerical simulations, along with the increase of computing resources and evolution of computer architectures. In this article we present an overview of the activities and mechanisms implemented by the US Department of Energy (DOE) Advanced Computational Software (ACTS) Collection Project that aim at mitigating the effort required for the development, maintenance and portability of computational science applications. The ACTS Collection comprises a set of DOE-developed software tools, sometimes in collaboration with other funding agencies, and that make it easier to write high performance codes for computational science applications. The ACTS Project implements additional services to guarantee the availability of the tools and also to enable their adequate utilization. This article summarizes the functionalities that the tools provide, applications that have benefited from their use, services provided by the project, and lessons learned through interactions with users that possess different levels of expertise.

1. Introduction

Traditionally, the models adopted by engineers and domain scientists for the development of simulation codes have ranged from fully in-house implementations to sophisticated computational environments, including calls to functionalities available in different libraries, with well-defined interfaces. The former approach usually leads to duplication of efforts, to software that may not perform satisfactorily (computationally and algorithmically), to software that may not even be portable, to a confinement of the know-how when the development is carried out by temporary personnel, and to monolithic codes that prevent users from extracting subsets of the implemented functionalities. In contrast, the latter approach allows for efficient software reuse and separation of application development concerns, and access to a much larger range of functionalities implemented in libraries possibly written in different or specialized languages, swapping between alternative (newer and/or faster) algorithmic implementations. Also, it makes it easier to incorporate new functionalities into the application codes and the adoption of emerging technologies for automatic tuning. Equally important, it frees application developers to focus on their specific areas of interest, and make.

Application developers can resort to combinations of compiler directives, language extensions, non portable library calls and even code rewrites to optimize the performance of their applications. Overall, this part of the application development is operationally very expensive and with very short term impact on the application. The cost of optimizing an application renders this kind of development almost impractical for current and future computational challenges, in particular when this cost is added to the costs of the initial development, debugging, synchronization of source code and consecutive version upgrades. As an alternative to this, high quality scientific

[†] This article is based on a presentation given at the Information Technology Center (ITC) of the University of Tokyo on March 3, 2008, as part of the author's two-month visit to various institutions in Japan. The writer is greatly indebted to his host at ITC, Prof. Takahiro Katagiri, and also to the Japan Society for the Progress of Science, which made the visit to Japan possible.

software libraries have been increasingly used in high-end simulation codes. This practice can successfully address issues pertaining to: a) research in computational sciences is fundamentally interdisciplinary; b) the development of complex simulation codes on high-end computers is not a trivial task; c) productivity, i.e. the time required to produce the first solution (prototype phase) to runs with realistic models (production phase); d) increasingly sophisticated models and fidelity of the models; and e) increasingly complex algorithms and computer architectures.

The US Department of Energy (DOE) Advanced Computational Software (ACTS) Collection comprises a set of computational software tools that aim at simplifying the solution of common and important computational problems [1][2]. ACTS tools are freely available[1] and targeted for distributed (MPI-based) computing; some of the tools also provide implementations for sequential architectures. The collection evolved from the former DOE 2000 Project, which had two main components, the Advanced Computational Testing and Simulation Toolkit and the National Collaboratory Project. One of the goals of the project was to change the way scientists work together and address major challenges of scientific computation by developing and exploring new computational tools and libraries. Therefore, most of the tools currently available in the ACTS Collection were primarily developed at DOE laboratories, in some cases in collaboration with universities. In addition, some tools have been co-funded by DOE and other agencies like the US National Science Foundation (NSF) and the US Defense Advanced Research Projects Agency (DARPA). The development of some of the tools required a long-term investment and effort, therefore the role played by DOE laboratories. Although some of the tools initially focused on problems of interest of DOE, they soon evolved and enabled the solution of a much larger variety of applications.

The ACTS Project complements the library research and development efforts by adding technical support, quality assurance and outreach. The technical support provided by the ACTS Project ensures that development efforts are better employed. As a result of this effort, ACTS libraries have reached higher acceptance levels among computational scientists and institutions. In turn, this outreach provides the necessary means for individual tool projects to interact with more users. As a result, the tools can gradually mature, becoming both more robust and portable to state-of-the-art high performance computing environments. In this overview article, we discuss our experience with the ACTS Collection Project towards the creation of a reliable software infrastructure for computational science computations. The article deals with various aspects of the ACTS tools, in particular categories of problems that they solve, functionalities that they provide, applications that have benefited from their use, lessons that we have learned by interacting with computer vendors and users with different levels of expertise, and mechanisms to minimize the usually costly application development effort.

2. ACTS: Functionalities Currently Available

Software libraries in the ACTS Collection fall in one of four (broad) categories: numerical tools, tools for code development, tools for code execution and tools for library development. The numerical tools implement numerical methods and algorithms, and include sparse linear system solvers, ODE solvers, optimization solvers, etc. The tools in the code development category provide infrastructure that manages some of the complexity of distributed programming (such as distributing arrays, communicating boundary information, etc) but do not actually implement numerical methods. Execution support is a category for application-level tools; which include tools for performance analysis. Library support tools provide an infrastructure for tool developers and probably will not be used or seen directly in scientific applications. The tools are selected, considered for inclusion and evaluated by taking into account efficiency, scalability, reliability, portability, flexibility, and ease-of-use [3]. Efficiency refers to the optimal use of computational resources in the system. Scalability is the ability to increase the number of processes and processors as the size and complexity of the problem being solved is also increased without compromising efficiency. Reliability refers to the failure free features of the library and proper handling

[1] Pointers and more information on how to download the tools can be found at the ACTS Information Center, *http://acts.nersc.gov.*

Table 1. *Set of tools that form the "solid base" of the ACTS Collection, i.e. the tools that have been installed (based on users' requests) and used on DOE's computer facilities.*

Tool	Functionalities
ATLAS	Automatic tuning of basic linear algebra subroutines. Developed at the University of Tennessee, Knoxville
Aztec	Algorithms (based on Krylov subspaces) for the iterative solution of large sparse linear systems. Developed at Sandia National Laboratories.
Hypre	Algorithms (based on Krylov subspaces) for the iterative solution of large sparse linear systems, intuitive grid-centric interfaces, and dynamic configuration of parameters. Developed at Lawrence Livermore National Laboratory.
Global Arrays	Library for writing parallel programs that use large arrays distributed across processing nodes; "shared-memory" programming interface for distributed-memory computers. Developed at Pacific Northwest National Laboratory.
OPT++	Object-oriented package of nonlinear optimization algorithms. Developed at Sandia National Laboratories.
PETSc	Tools for the solution of PDE problems that require solving large-scale, sparse linear and nonlinear systems of equations. Developed at Argonne National Laboratory.
SUNDIALS	Solvers for large systems of ordinary differential equations, nonlinear algebraic equations, differential-algebraic equations, and sensitivity analysis. Developed at Lawrence Livermore National Laboratory.
ScaLAPACK	Library of high performance dense linear algebra routines for distributed-memory architectures. Developed at the University of Tennessee, Knoxville, and UC Berkeley.
SLEPc	Software package built on top of PETSc for the solution of large sparse eigenvalue problems. Developed at the Universidad Politecnica de Valencia, Spain.
SuperLU	General-purpose library for the direct solution of large, sparse, systems of linear equations. Developed at UC Berkeley and Lawrence Berkeley National Laboratory.
TAO	Tools for the solution of large-scale optimization problems, including nonlinear least squares, unconstrained minimization, bound constrained optimization, and general nonlinear optimization. Developed at Argonne National Laboratory.
TAU	Tools for analyzing the performance of programs written in C, C++, Fortran or Java. Developed at the University of Oregon and Los Alamos National Laboratory.

of error bounds. Portability refers to the almost adaptability of the libraries to a wide variety of computational environments. Flexibility is the feature that allows users to construct new routines, libraries and codes from well-defined tool modules. The use of flexible software automatically leads to extensible software. Ease-of-use delivers interfaces that users outside the community of developers can adopt and become familiar with. All these considerations make ACTS tools an important resource for a wide spectrum of applications.

Table 1 summarizes the functionalities of the tools that currently form the core of the ACTS Collection, i.e., the tools that we have installed and supported at DOE's National Energy Research Scientific Computing Center (NERSC). Other computer facilities also provide installation of the tools, including DOE's National Center for Computation Sciences (NCCS), the San Diego Supercomputing Center (SDSC), the Pittsburgh Supercomputing Center (PSC), the National Center for Supercomputing Applications (NCSA), and Virginia Tech. Although this list is not comprehensive, it gives a taste of the range of solutions provided by the tools. Also, it suggests that the tools have been used in different scenarios, for example for code development, prototyping, and large simulations that have the potential to lead to scientific breakthroughs. To a large extent, the tools in Table 1 reflect the state-of-the-art software and hardware technologies. More information about the functionalities that they provide can be found in [1][2]. We note that the collection itself is not static: tools need to be retired (for example when development has ceased or porting to new computing scenarios is no longer possible), or replaced by new technologies.

Examples of Use

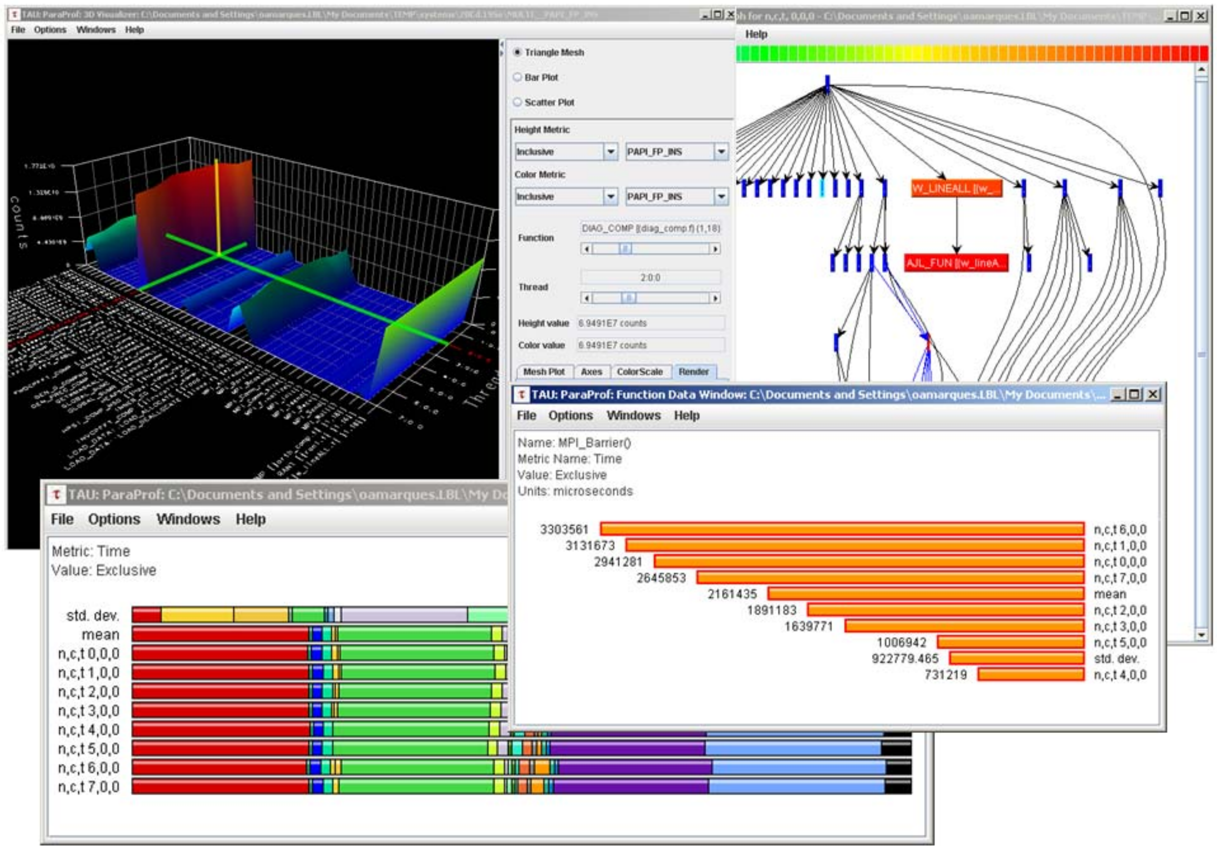
In the last several years, a number of important scientific and engineering problems have been successfully studied and solved by means of computational modeling and simulation [4]. Many of these computational models and simulations benefited from the use of available software tools and libraries to achieve high performance, in particular tools listed in Table 1. These tools have facilitated code portability across many computational platforms while guaranteeing the robustness and correctness of the algorithmic implementations. To illustrate, here we give a short list of applications that show how ACTS tools have been used:

- Aztec: solution of systems of linear-equations in for multi-phase fluid and heat flow in porous and fractured media (TOUGH2 code).
- Global Arrays: data distribution in molecular dynamics codes (including NWChem, GAMESS-UK, Columbus, Molpro and Molcas).
- OPT++: protein energy minimization problems.
- PETSc: model of the heart mechanics (blood-muscle-valve) by an adaptive and parallel version of the immersed boundary method, micro-finite element bone modeling, magma dynamics using a semi-lagrangian advection schemes.
- ScaLAPACK: solution of linear systems in spectral algorithms for the study of electromagnetic wave-plasma interactions; solution of eigenvalue problems in electronic structure calculations; solution of linear systems in astrophysics.
- SLEPc: equilibrium and stability of Tokamak plasmas, analyses of electronic and optoelectronic devices , Hessian-based model reduction.
- SuperLU: parallel exact shift-invert eigensolver based on PARPACK for the modeling and analysis of accelerator cavities; computation of preconditioners for the solution of linear systems in quantum mechanics problems.
- SUNDIALS: uncertainty quantification for groundwater simulation, sensitivity analysis in a population dynamics model, sensitivity analysis in neutral particle transport simulation.
- TAO: electronic structure optimization, linguistic processing, subsurface remediation problems, finite element micromagnetics simulations.

Even if an application does not require numerical functionalities like the ones provided by the tools listed in Table 1, application developers are often faced with questions related to computational performance, including[2]: a) significant performance variability when using different compilers, b) performance variability possibly related to operating system features, c) performance changes caused by recent changes in application code, d) performance variability under MPI variants, e) one version of the application is faster than another, f) similar runs exhibit different performances, and f) the architecture that is best suited for an application. TAU (see Table 1) is a tool that can be employed in the understanding of all these issues. TAU provides capabilities for *profiling* (recording of summary information during execution such as inclusive and exclusive time, number of calls, hardware statistics, etc) and *tracing* (recording of information about significant points, i.e. events, during program execution, such as function, loops, blocks, thread/process interactions, etc). It can be used for multi-level performance instrumentation, multi-language source instrumentation, multiple parallel programming paradigms, and object-

[2] Adapted from Sameer Shende's presentation, *http://acts.nersc.gov/events/Workshop2005/slides/Shende.pdf.*

Figure 1. *Snapshots of PARAPROF, a performance data visualizer provided with TAU. The pictures show performance data for PESCAN, a code that uses the folded spectrum method for non self-consistent nanoscale calculations of materials. It is parallelized using MPI and can calculate million atom systems. For illustration purposes, the performance data was collected on 8 processors on an IBM SP5; the data was later analyzed on a PC desktop.*



TAU: ParaProf: 3D Visualizer: C:\Documents and Settings\oamarques.LBL\My Documents\TEMP\...
File Options Windows Help
Triangle Mesh
Bar Plot
Scatter Plot
Mesh Plot
Axes
ColorScale
Render
W_LINEALL [{w_
AJL_FUN [{w_lineA
TAU: ParaProf: C:\Documents and Settings\oamarques.LBL\My Do
TAU: ParaProf: Function Data Window: C:\Documents and Settings\oamarques.LBL\My Documents\...

Height Metric: Inclusive PAPI_FP_INS
Color Metric: Inclusive PAPI_FP_INS
Function: DIAG_COMP [{diag_comp.f} {1,18}]
Thread: 2:0:0
Height value: 6.9491E7 counts
Color value: 6.9491E7 counts

Metric: Time
Value: Exclusive

std. dev.
mean
n,c,t 0,0,0
n,c,t 1,0,0
n,c,t 2,0,0
n,c,t 3,0,0
n,c,t 4,0,0
n,c,t 5,0,0
n,c,t 6,0,0
n,c,t 7,0,0

Name: MPI_Barrier()
Metric Name: Time
Value: Exclusive
Units: microseconds

3303561	n,c,t 6,0,0
3131673	n,c,t 1,0,0
2941281	n,c,t 0,0,0
2645853	n,c,t 7,0,0
2161435	mean
1891183	n,c,t 2,0,0
1639771	n,c,t 3,0,0
1006942	n,c,t 5,0,0
922779.465	std. dev.
731219	n,c,t 4,0,0

oriented and generic programming. In summary, TAU provides a variety of services for instrumentation and tuning, while requiring a minimum effort for code instrumentation. In fact, all instrumentation can be done automatically, by means of easy-to-use scripts provided with the package. The visualization and analysis of the performance results can also be done by means of tools included with the package. In Figure 1 we show snapshots of the performance data visualization capabilities included in TAU.

3. ACTS: Added Services

The ACTS Project is actively engaged in education activities, by organizing and participating in workshops, tutorials and other events related to computational sciences. The goal is to promote the reuse of robust software tools and at the same time provide guidance on their use. Starting in 1999, the ACTS Project has been hosting a yearly workshop at the Lawrence Berkeley National Laboratory, with DOE funds. The workshop consists of presentations, tutorials and hands-on sessions. Participants are chosen based on an application process, which aims at identifying and matching applications and computational needs. During the hands-on sessions, the participants have the opportunity to compile and run examples that show how ACTS tools can be used. This provides a unique opportunity for students to be become acquainted with tools intended for high end computer simulations.

The following is a list of recent events organized or co-organized by the ACTS Project:

- *An outlook on Scientific Software Libraries* and *Writing Scientific Software: Experience with Using, Developing and Teaching*, ICIAM2007 Conference, Zurich, Switzerland, 2007.
- *Workshop and Advanced School on Eigenvalue Problems, Software and Applications*, Porto, Portugal, 2007.
- *Tools, Frameworks and Applications for High Performance Computing*, minisymposium, PARA'06 Workshop, Umeå, Sweden, 2006.
- *Tutorial on Robust and High Performance Software Libraries for Computational Sciences*, VECPAR'06, Rio de Janeiro, Brazil.
- *ACTS Collection Workshop at SDSC*, San Diego, CA, 2006.
- *Short Course on the ACTS Collection*, SIAM CSE05 Conference, Orlando, FL.

In addition to training and outreach activities, the ACTS Project also investigates requirements for reusable high quality software tools, mechanisms for integration, maintenance and support, interfaces using script languages, and software automation strategies.

4. ACTS: Lessons Learned

Important lessons have emerged from the activities carried out by the ACTS Project. While some of the lessons are perhaps specific to DOE communities, they are nonetheless valuable to the computational science and engineering communities as a whole, and in particular to the software development and support projects:

- *There is still a gap between tool developers and application developers that leads to duplication of efforts*. Without projects like ACTS, application developers will continue to design and implement codes using techniques that are already available from other sources. Quite often these in-house implementations are not optimal because of the application developers' inexperience with all the different issues that lead to optimal performance.
- *Users demand long-term support of the tools*. One of the main concerns that users have expressed is the longevity of support from tool developers and required evolution of the software as the hardware technology continues to evolve and the complexity of the application continues to grow.
- *Applications and users play an important role in hardening tools*. The main parameters for software maturity are portability, robustness, acceptance, and long-term support. It is particularly the interactions with real users and real applications that have made the software mature, portable, robust and better documented. In turn, mature software will be widely accepted inside a given scientific community.
- *Tools evolve or are superseded by other tools*. As technology continues to advance, there are some tool functionalities that are either no longer needed or are improved as direct consequences of users' requirements.
- *There is a demand for tool interoperability and more uniformity in the documentation and user interfaces*. Users want to experiment with functionalities available in a subset of tools. Finding similar user interfaces and comparable levels of support and documentation makes this task even simpler and riskless. The computational challenges at hand also demand new software developments that interact with legacy code practices, data and computer languages.

5. ACTS: The Present and the Future

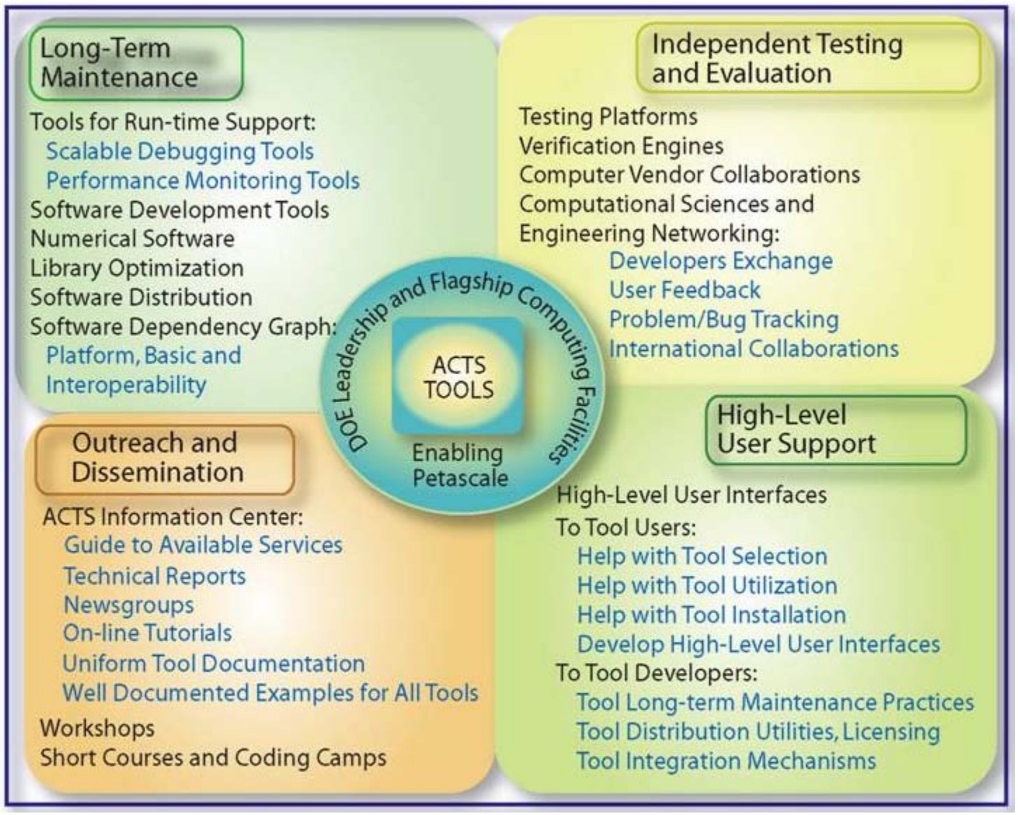
The ACTS Collection Project has been serving as a mechanism for the development, support, and promotion of quality high performance software tools. The goals and successes of the project have been enhanced by the increasing demand for complex high performing computer simulations, closer interactions between computer scientists and other domain scientists, less duplication of efforts, and interoperability to pick-and-play with kernels from a variety of computational services. We have been working to continuously expand the scope of the ACTS Collection Project based on the lessons that we have learned. The evolution of the ACTS Project towards a reliable software infrastructure for scientific computing takes into consideration the following items:

- *A Solid Base for the ACTS Collection*. The mature tools in the ACTS Collection form a solid base that has contributed to the acceptance of the tools in computation sciences. We have defined mechanisms for the inclusion of new tools in the collection and a process that certifies the tools as part of the collection.
- *ACTS High Quality Software Certification*. We have created a software certification process for tools in the collection and that is defined in terms of software availability, robustness, functionality, portability, documentation, and interoperability.
- *Inclusion of New Solutions to Computational Problem*. The development of good quality complex parallel codes is usually very expensive. The ACTS Collection aims at promoting code reusability for the solution of common and important computational problems as a means of accelerating scientific discoveries. It also aims at fostering the development of tools that are not available and recommends good quality tools developed by parties not necessarily funded by DOE.
- *Interoperability and Software Distribution.* Interoperability may affect performance in some cases but it potentially reduces time to solution and, most important, it assures longevity of the software. Moreover, language choices made by the tool developers must not dictate the choice of language used by the application developers.
- *Encourage Feedback from Users*. The ultimate measure of success for the high performance computing tools provided by ACTS is their role in the production of high quality scientific results. This can be assessed, for instance, by encouraging feedback from current and potential users as well from participants in the workshops and other activities related to ACTS, and by giving presentations and fomenting discussions in major events covering computational sciences topics.
- *Expertise Tracking*. The ACTS Information Center attempts to collect and make available all information from tool developers, *acts-support* and users' experiences with the tools. In some cases, tool developers keep their own e-mail lists of users, by means of which questions and problems are posed and later either the developers or other experienced users propose answers and solutions. We have been expanding this capability in the ACTS Information Center by continuing to participate in these e-mail lists and looking into ways to employ more specialized search tools to retrieve this information from feedback, evaluations, and reports collected on a particular tool.

The key component of this long-term user support is the coordination of efforts between software and hardware vendors, tool developers, users and ACTS. We have realized that the main parameters for software maturity are portability, robustness, acceptance, and long-term support. And it is in turn the interactions between application and tool developers that have made the software tools more mature, portable, robust and better documented. Therefore, we have been working with commercial software developers and computer vendors to guarantee the long-term support for the tools and to effectively reach out more user communities. To complement the above activities, and among others, we have also implemented a quarterly electronic newsletter, worked on tentative collaboration agendas with various research centers and computer facilities, collaborated with tool developers towards the publication of special issue of journal featuring ACTS tools, responded to requests for ideas on how to efficiently deploy future high-end computing technologies, and worked on the development of a Python Interface to the numerical libraries in the ACTS Collection.

In the ACTS Collection Project our efforts have been characterized by an intense outreach and dissemination of the tools, and high-level user support to application developers while writing their codes and selecting functionalities available in the collection. We aim at expanding these efforts with new services to enable the long-term availability, performance and readiness of the tools. In these lines, we have been working on the creation of a center for the sustainable maintenance and support of a collection of software tools, the ACTS Collection. Figure 2 presents the scope of the *ACTS Center* and its formulated set of services. We group these services in four main

Figure 2. *Schematic of the proposed ACTS Center and its set of services*.


Long-Term Maintenance
Tools for Run-time Support:
Scalable Debugging Tools
Performance Monitoring Tools
Software Development Tools
Numerical Software
Library Optimization
Software Distribution
Software Dependency Graph:
Platform, Basic and Interoperability
Independent Testing and Evaluation
Testing Platforms
Verification Engines
Computer Vendor Collaborations
Computational Sciences and Engineering Networking:
Developers Exchange
User Feedback
Problem/Bug Tracking
International Collaborations
DOE Leadership and Flagship Computing Facilities
ACTS TOOLS
Enabling Petascale
Outreach and Dissemination
ACTS Information Center:
Guide to Available Services
Technical Reports
Newsgroups
On-line Tutorials
Uniform Tool Documentation
Well Documented Examples for All Tools
Workshops
Short Courses and Coding Camps
High-Level User Support
High-Level User Interfaces
To Tool Users:
Help with Tool Selection
Help with Tool Utilization
Help with Tool Installation
Develop High-Level User Interfaces
To Tool Developers:
Tool Long-term Maintenance Practices
Tool Distribution Utilities, Licensing
Tool Integration Mechanisms

areas: *long-term maintenance*, *independent testing and evaluation*, *outreach and dissemination* and *high-level user support*. We goal is to provide a distributed implementation of the ACTS Center across the DOE leadership and flagship computing facilities by building a strategically planned infrastructure to support our collaborations with these facilities.

6. CONCLUSIONS

This article has summarized needs of the computational science community at large, where significant efforts are focused on the development of complex parallel codes and their optimization. This expensive process often requires specialized support and information about software tools, and becomes crucial if we consider that more complex physical and societal phenomena, along with the growth of computing resources, is driving the continuous growth of the gallery of computational sciences applications. We foresee a great need not only for state-of-the-art software, but also a collaborating infrastructure in which knowledge and expertise is captured and shared. A high-end software infrastructure can produce substantial performance information from interactions between algorithm, tool and application developers. In general, a wide range of scientific code developers and users benefit from a) information and education about state-of-the-art, high-end computational tools; b) the development and promotion of robust, effective, portable, usable, and durable software; c) the increased interoperability of tools, which promotes the evolution and adoption of current software development projects into future software technologies; d) multidisciplinary collaborations and the consequent accumulation of expertise; and e) spending less time on code development and having more time to devote directly to scientific discovery.

As discussed in Section 4, users demand long-term support of the tools, which implies not only bug fixes, but also addition of new features, porting and documentation updating. Ideally, the tool developers should be in charge of these activities. It turns out that this model raises important issues about funding opportunities, since software maintenance has not been properly addressed in the various funding agencies' agendas. Nonetheless, the ACTS Project implements a viable solution to bring all these efforts to the computational science community while exploring mechanisms for extending the longevity of the software beyond their development phase. Therefore, projects like ACTS are important for maintaining a high quality software collection by not only attesting the quality of tools but also ensuring that users select the more suitable tools and tool functionalities, and make proper use of these.

In terms of the various types of users, here we borrow Prof. Nakajima's broad ranking of users:

- A-rank: can develop parallel codes by themselves from scratch (using ScaLAPACK, Aztec, PETSc, etc).
- B-rank: can develop codes by themselves using more sophisticated frameworks (like HPC-MW, GeoFEM, etc).
- C-rank: users of parallel codes (STAR-CD, GAUSSIAN,etc).
- S-rank: A-rank + can make contribution on HPC-MW.

We believe that a project like ACTS is important for maintaining a high quality software collection by not only attesting the quality of tools but also ensuring that users – in particular the A and B groups listed in the above ranking – select the more suitable tools and tool functionalities, and make proper use of these.

If terms of productivity, we borrow a metric from [5]

$$\psi = \frac{U(T)}{C_s + C_o + C_m}$$

where Ψ is the overall system productivity, $U(T)$ is the utility of the solution (a function of time, the longer the time to solution, the lower the utility of that solution will be), C_S is the cost of software, C_O is the cost of operator, and C_M is the cost of the machine. Readily available, robust and high-performing software tools can contribute to reducing the time to solution and the cost of the application development, therefore leading to a better system productivity.

Acknowledgments

The author would like to express his gratitude to all ACTS tools developers for their feedback over the years, as well as to Prof. Kengo Nakajima for stimulating discussions on software tools.

For more information about the ACTS Project the reader is invited to visit *http://acts.nersc.gov* or send an e-mail to *acts-support@nersc.gov.*

7. References

[1] T. Drummond and O. Marques, *The Advanced CompuTational Software (ACTS) Collection*, ACM TOMS, 31:282-301, 2005.

[2] The International Journal of High Performance Computing Applications, Special Issue on ACTS Tools in the ACTS Collection, Vol. 20, 2006.

[3] T. Drummond and O. Marques, *The ACTS Collection, Robust and High-Performance Tools for Scientific Computing: Guidelines for Tool Inclusion and Retirement*, Technical Report LBNL-PUB-3175, November 2002.

[4] T. Drummond, O. Marques, J. Roman and V. Vidal. *A Study of Robust Scientific Libraries For The Advancement of Sciences and Engineering*, Lecture Notes in Computer Science, Vol. 3402, Springer-Verlag, Berlin, 2005. VECPAR 2004, Valencia, Spain, June 28-30, 2004, revised selected and invited papers.

[5] A. Funk, MIT .V. Basili, .L. Hochstein.J. Kepner, *Analysis of Parallel Software Development using the Relative Development Time Productivity Metric*, CTWatch, November 2006 A.