An Infrastructure for the Creation of High End Scientific and Engineering Software Tools and Applications

L. A. Drummond[1], O. A. Marques[2] and G. V. Wilson[3]

1. Introduction

This document has been prepared as a response to the High End Computing Revitalization Task Force (HECRTF) call for white papers. Our main goal is to identify mechanism necessary for the design and implementation of an infrastructure to support development of high-end scientific and engineering software tools and applications. This infrastructure will provide a plethora of software services to facilitate the efficient deployment of future HEC technology as well as collaborations among researchers and engineers across disciplines and institutions. In particular, we address here the following points:

- Key software technologies that must be advanced to strengthen the foundation for developing new generations of HEC systems (in response to charge 1). At core of these technologies we identify the development libraries for code profiling, flexible coupling of large scale applications, interface and language interoperability, and implementations of high-end numerical and computational algorithms.
- A Software Infrastructure for minimizing "time to solution" by users of HEC systems (in response to charge 2d). This infrastructure provides the mechanisms for software tools to mature, become robust, more acceptable by a pool multidisciplinary users, and responsive to the needs from real applications. More importantly is the ability for these tools to evolve to new emerging HEC systems to guarantee the long-term support of the tools. Consequently, users of HEC tools will utilize better the resources available to them, will be able to provide feedback to the tool development teams and focus their attention more on the challenges of their work rather than coping with advancements in HEC systems.

The technical facts to address the above points come from our gathered expertise designing, managing and working with the DOE Advanced CompuTational Software Collection (ACTS [01]). The ACTS Collection Project has been successful in outreaching the computational sciences community. Here, the high-end software infrastructure is our overarching approach to integrate the most appropriate solutions to arising problems in computational sciences, and to quickly respond to the dynamic nature of software evolution and HEC technology.

2. The Background.

The ACTS Collection comprises a set of computational tools developed primarily at DOE laboratories, sometimes in collaboration with universities and other funding agencies (NSF,

[1] Lawrence Berkeley National Laboratory, One Cyclotron Rd, MS 50F 1650. Berkeley, CA 94720-8139, USA, LADrummond@lbl.gov, (510) 486-7624
[2] Lawrence Berkeley National Laboratory, One Cyclotron Rd, MS 50F 1650. Berkeley, CA 94720-8139, USA, OAMarques@lbl.gov, (510) 486-5290
[3] Baltimore Technologies, Toronto, 43 Ryerson Avenue #2,Ontario M5T 2P4, Canada. GVWilson@baltimore.com, (416) 504-3654

DARPA), aimed at simplifying the solution of common and important computational problems. A number of important scientific problems, ranging in scale from the atomic to the cosmic, have been successfully studied and solved by means of tools available in the Collection (some examples of these are given in [02]-[05]).

Software development time includes the time required for individual tasks, such as coding and debugging, and the time needed for group coordination, such as reporting bugs, synchronizing source code, and preparing releases. The inefficiency of *ad hoc* approaches to coordinating group activities has already become a significant hidden cost in many large scientific computing projects. The creation of a high-end software infrastructure will not only enable and facilitate more collaborations, particularly geographically distributed ones, but it must provide ways to reduce code development, distribution and maintenance costs.

ACTS provides a unique opportunity to tool development projects to interact with a wider community of computational scientists as well as other tool development projects. Increased use means increased feedback, ultimately resulting in higher quality tools. The community of users of the ACTS Collection has provided the tool developers with measures of success for their tools through bug reports, features of interest and preferences. These interactions provide the proper dynamics for the tools to gradually mature, and become more robust and portable to state-of-the-art HEC environments. Further, this unique scenario facilitates a number of opportunities to take a leap forward in the use and acceptance of "experimental software" (ES), i.e. software from the research community that is not supported by computer vendors.

3. Key Software Technologies to Support HEC
What is needed and what the proposed infrastructure will deliver

The future trend of scientific and engineering applications will demand programming abstractions that are not necessarily sequential in nature and therefore not based on a single thread. In this scenario, the adoption of higher levels of abstractions (i.e., use of software tools and libraries, [07]) during the implementation of an algorithm will be essential for portability and sustainable performance.

Therefore, the high-end software infrastructure needs mechanisms to improve the rate in which new developments are passed from tool developers to users and likewise the way in which computational needs are communicated from the users to the tool, algorithm and HEC technology developers. These mechanisms must provide support for code development, testing, profiling and tuning.

In the last five years, there has been a growth of Open Source software development. Open Source projects share two key features: a) programs are distributed in source form, so that any interested party can inspect or modify the underlying code; b) distributed collaborative development is the norm. However, Open Source software frameworks like SourceForge and its imitators have several significant flaws, which we believe make them inadequate for the dynamic nature HEC needs. Firstly, these systems have been built piecemeal by gluing legacy tools together. As a result, installing them on a local server requires high levels of operating system skills. Secondly, none of the existing Open Source frameworks includes a regression testing mechanism to enable users to run batches of tests, compare their results with previous runs, and archive or publish the results. Thirdly, none of these systems include basic project scheduling tools or time-tracking tools. Lastly, systems like SourceForge are now moving to a *Closed Source* approach in which the software is own by an enterprise. In this scenario users are unable to write extension and have to pay a price for the services.

To effectively identify and produce HEC Core Software Technologies, we anticipate a need for:

- A next-generation collaborative environment for distributed development of large-scale scientific software. This environment will be *100% Open Source*, so that stakeholders can extend it to meet the needs of their specialized domains, or new needs as they emerge. The Open Source model will also facilitate uptake by academic collaborators, who might find the high cost of commercial systems a barrier. Finally, making the system Open Source will allow parts of it to be used as examples in training material.
- Increase the rate of adoption of ES by the computational science community by providing improved mechanisms for outreach, documentation, education, dissemination of results and development of the tools and their applications.
- Develop an infrastructure to agilely respond to the feedback cycle between computational sciences problems, algorithm development and software evolution and development.
- High End Software support and consultancy to provide guidance in tool selection and utilization to the tool users. This high level support for tool development to facilitate the integration of tools and efforts.

4. Towards a Software Infrastructure for Minimizing Time to Solution

The use of the robust software tools, as the ones in the ACTS Collection, substantially reduces the development time for new codes. ACTS has promoted reusability over duplication of efforts and tool interoperability over "hard-wired" and intrusive library interfaces that only serve a specific project.

As a result of our activities under the ACTS Collection Project, tutorials and workshops that we have participated or organized [06], and also by interacting with users and tool developers, we have learned various lessons that we believe must be taken into consideration for improving and guaranteeing a sustainable software infrastructure aiming at high-end computing scientific simulations: a) *the tools available in a software collection should be seen as dynamically configurable collections and should be grouped into collections upon user/application demand*; b) *users demand long-term support of the tools*; c) *tools evolve or are superseded by other tools*; and d) *there is a need for an intelligent and dynamic catalog/repository of high performance tools.* Based on these lessons, we believe that the creation of a reliable software infrastructure for HEC should take into consideration the following items:

1. *A Solid Base of Tools.* As in the ACTS Collection Project, we will make calls to all the tool development communities for tool submission. The matured tools will form a solid base that will be used by the computation sciences community and can therefore set standards for high performance software development. These standards, together with peer-reviewed tool certification mechanisms, will continue to build the scientific community's confidence in software tools.
2. *High Quality Software Certification.* We will build in the ACTS's implementation of a peer-reviewed certification process to push the frontiers of tools forward, and the tools or subsets of the tools will be eventually integrated in vendor-supported software libraries. The software certification process should be defined in terms of correctness, robustness, functionality and applicability, portability, documentation, availability and distribution, interoperability, and extendibility [07]. The goal is not to preclude new software paradigms

but to promote existing and new paradigms based on what works best for the computational science community.

3. *Interoperability and Software Distribution*. Interoperability potentially reduces time to solution and, most important, it assures longevity of the software. Interoperability can be used for the distribution of tools based on user demand. Currently, this has remained a difficult task given the different compiler requirements, makefiles, distributions, and installations posed by different tool developers. On these lines, the high-end software infrastructure must provide ways of conveying more information at library interfaces. This information could include, for instance, semantic information that can be provided by the user, or deduced by the compiler, or system performance information. Interoperability inside the high-end software infrastructure will be based on efforts like the CCA ([09]) and Babel ([13]). Semantic information across interfaces will populate the algorithm-tool-application performance database.
4. *Encourage Feedback from Users*. The ultimate measure of success for EHC tools is their role in the production of high quality scientific and engineering results. This can be assessed, for instance, by encouraging feedback from current and potential users. Besides, there is therefore a need for the creation and investigation of other useful metrics to support the usage of the tools. This includes, but is not limited to, the time that an application team spent learning about the use of a tool, and the time it took a development team to prototype an application with a particular tool.
5. *Proactive Collaboration among Software Initiatives*. Currently, there are a number of high-end computing programs that could interact among them, such as the SciDAC Program [10], NASA's High Performance Computing Cooperative Agreement Notice (CAN [11]) round III, NSF's Information Technology Research (ITR [12]) and NPACI [08]. All of these programs will develop software technology to address complex scientific problems in today's and the future's high-end hardware. Interactions among these initiatives should be further encouraged.
6. *Dissemination Plans.* Education of graduate students and postdoctoral fellows provides a viable resource to build a bridge between computer scientists and domain scientists. By educating other scientists on the use of a set of tools, they become familiar with the technology, accept this technology faster, are able to develop codes using state-of-the-art tools, and minimize the tool selection process.
7. *Information center and software repository.* Good quality software is usually labor-intensive to develop, test, maintain, and evolve. We envision a mechanism to improve the life cycle of HEC software. A software repository and information center must include: a) self-contained installers for the server software on a variety of platforms, and for any client-side software users may require; b) core project coordination functionality, including version control, issue tracking, release management, regression testing, check-pointing, and threaded discussion management; c) complete documentation for installers, administrators, and users, and for developers who wish to extend the system; d) Open Source technology support for code development as described in Section 3; and e) domain-expertise cross-indexing mechanism to facilitate the navigation through the information system.

5. Summary of Benefits

The issues that we address in this white paper attend the 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. Therefore, we foresee a great need not only for a state-of-the-art software repository, where tools and their ongoing developments are available and documented, but also a collaborating infrastructure in which knowledge and expertise is captured and shared.

A high-end software infrastructure will produce substantial performance information from interactions between algorithm, tool and application developers. We envision the creation of a database of performance data at the algorithm, tool and application levels. As a result, we will obtain better characterizations of the behavior of today and future software technologies in HEC systems (complements responses to charges 3a and 3b).

The benefits of deploying such an infrastructure can be measured in many ways. A wide range of scientific code developers and users will 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.

Bibliography

[01] ACTS Information Center, http://acts.nersc.gov.

[02] Canning, W. Mannstadt and A. J. Freeman, 2000. *Parallelization of the FLAPW Method*, Computer Physics Communications, 130:233-243, Elsevier, The Netherlands.

[03] T. Rescigno, M. Baertschy, W. Isaacs and W. McCurdy, 1999. *Collisional breakup in a quantum system of three charged particles*, Science, 286:2474-2479.

[04] P. de Bernardis, P. A. R. Ade, J. J. Bock, J. R. Bond, J. Borrill, A. Boscaleri, K. Coble, B. P. Crill, G. De Gasperis, P. C. Farese, P. G. Ferreira, K. Ganga, M. Giacometti, E. Hivon, V. V. Hristov, A. Iacoangeli, A. H. Jaffe, A. E. Lange, L. Martinis, S. Masi, P. V. Mason, P. D. Mauskopf, A. Melchiorri, L. Miglio, T. Montroy, C. B. Netterfield, E. Pascale, F. Piacentini, D. Pogosyan, S. Prunet, S. Rao, G. Romeo, J. E. Ruhl, F. Scaramuzzi, D. Sforna and N. Vittorio., 2000. *A flat Universe from high-resolution maps of the cosmic microwave background radiation*, Nature, 404: 955–959.

[05] D. M. Mitnik, D. C. Griffin and N. R. Badnel, 2001. *Electron-impact excitation of* Ne^{5+}, J. Phys. B: At. Mol. Opt. Phys. 34:4455-4473 (28 November).

[06] The ACTS Collection, *Presentations*, http://acts.nersc.gov/events.

[07] The ACTS Collection, *Guidelines for Tool Inclusion and Retirement*, http://acts.nersc.gov/documents/ToolCertification.pdf.

[08] NPACI, http://www.npaci.edu

[09] CCA-Forum, http://www.cca-forum.org

[10] SciDAC/DOE, http://www.er.doe.gov/scidac

[11] CAN/NASA, http://sdcd.gsfc.nasa.gov/ESS/CAN.html

[12] ITR/NSF, http://www.itr.nsf.gov

[13] Babel, http://www.llnl.gov/CASC/components/babel.html