Results of the PERI Survey of SciDAC Applications

Bronis R. de Supinski[1], Jeffrey K. Hollingworth[2†], Shirley Moore[3], and Patrick H. Worley[4]

[1]Lawrence Livermore National Laboratory, Livermore, CA 94550
[2]University of Maryland, College Park, MD 20742
[3]University of Tennessee, Knoxville, TN 37996
[4]Oak Ridge National Laboratory, Oak Ridge, TN

Corresponding author e-mail: hollings@cs.umd.edu

Abstract. The Performance Engineering Research Institute (PERI) project focuses on achieving superior performance for Scientific Discovery through Advanced Computing (SciDAC) applications on leadership class machines through cutting-edge research in performance modeling and automated performance tuning. This focus requires coordinated activities to engage SciDAC application teams. The initial application engagement activity was a survey of these teams to determine their performance goals, the criticality of those goals, current performance of their applications, application characteristics relevant to performance and their plans for future optimization. Using a web-based questionnaire, PERI researchers have worked with application developers to provide this information for over twenty-five applications. This paper describes the initial analysis of the application characteristics and performance goals, as well as current and future application engagement activities driven by these results. While the survey was conducted primarily to meet PERI needs, the results represent a snapshot of the state of SciDAC code development and may be of use to the DOE community at large. Overall, the results show that SciDAC application teams are engaged in significant new code development, which will require flexible performance optimization techniques that can improve performance as the applications evolve.

1. Introduction

The Scientific Discovery through Advanced Computing (SciDAC) Performance Engineering Research Institute (PERI) [2] has three main thrusts:

- application modeling and performance prediction;
- automated performance tuning; and
- direct engagement with SciDAC application projects.

The initial PERI engagement activity was a survey of the SciDAC application teams. The primary goal of the survey was to capture information that can be used to guide the other application engagement activities, ensuring that realistic SciDAC performance priorities are met. This information also informs and drives PERI long-term research on performance modeling and automated tuning, ensuring their relevance to these applications and, thus, to the program in general. While we aimed to capture a broad enough set of information for these purposes, modeling our survey after [1], we also tried to minimize the time required by the application teams by including only performance relevant questions and by assigning a member of the PERI team to work with each application contact on completing the survey. The survey implementation consists of a web-based questionnaire, with the results stored in a mySQL database. The results are also accessible from a web-based interface. While the survey itself is publicly accessible from the PERI website [2], the survey results are password-protected to maintain the confidentiality of the survey participants.

† Corresponding Author.

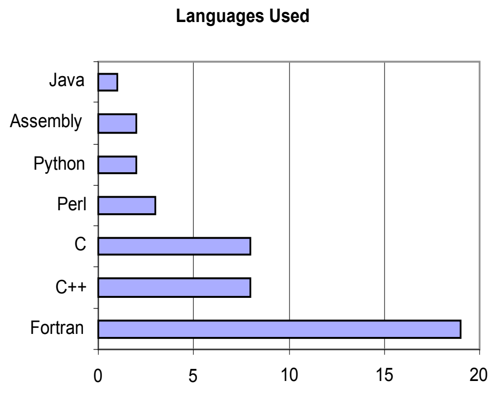
The questionnaire consists of three main sections: project contact information, performance issues, and code characteristics and structure. The code characteristics and structure are essential for designing automated performance tuning support. In addition, they can shape the form of performance models. Finally, characteristics common across many of the applications, such as the use of LAPACK, indicate opportunities to improve performance of many applications with one focused effort.

The remainder of this paper describes results of analysis of the initially collected survey data in sections 2 and 3, followed by current application engagement activities driven by these results in section 4, followed by conclusions and future work in section 5.

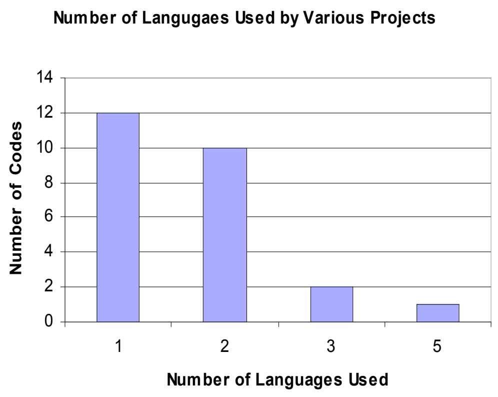
2. Application characteristics

To establish performance analysis and optimization tool requirements, and to gain a general understanding of SciDAC applications, we first report some general application characteristics. Characteristics of interest include programming languages, numerical methods, parallelization techniques (e.g., data decomposition techniques), parallel implementation language or infrastructure, libraries used, and code size. As with any voluntary survey, we must be careful when drawing conclusions because the applications included may not be representative of all SciDAC applications. Not all application teams answered all questions, so the number of responses varies slightly with each question.

Figure 1 shows the different languages used. The left side shows the occurrence of the languages. Fortran is by far the most common language. However, the combined total of C and C++ is almost as large. Several projects reported using a scripting language such as Python or Perl. Only one team uses Java. The right graph in Figure 1 shows the number of languages used by various projects. While 12 of the projects surveyed used only one language, 13 projects use more than one language. The implication of this for tool builders is that we need tools that support more than just Fortran, and tools generally must support a single code that is written in multiple languages.



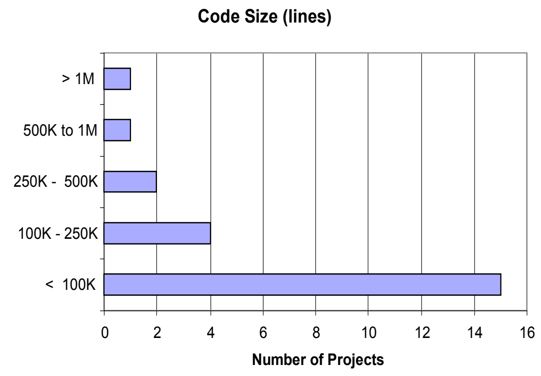
(a) Frequency of use of different languages.

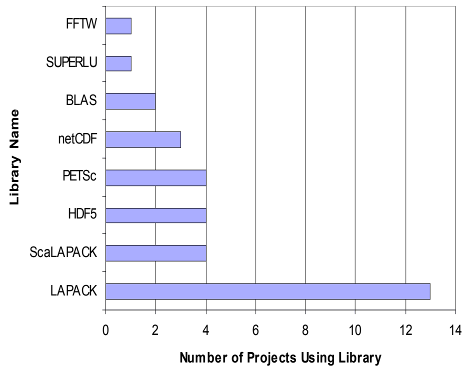


(b) Distinct languages used by a given project.

Figure 1: Languages Used.

We next examine the size of the applications being developed. Figure 2(a) shows a histogram of the number of lines of code in each of the projects surveyed. About half of the projects contain 100,000 lines of code or less. Only two of the projects contained more than 500,000 lines of code. While the number of projects with more than half a million lines of code is limited, the presence of many projects with more than 100k lines of codes shows the need for tools that can scale beyond a few modules.





(a) Size of code for different projects (lines). **(b) Libraries used by specific projects.**

Figure 2: Code Characteristics.

We also looked at the use of commonly available libraries by the application teams, as summarized in Figure 2(b). Almost half use the LAPACK library, and there were three or more uses of netCDF, PETSc, HDF5, and ScaLAPACK. For tool developers, information about library usage is important for two reasons. First, tools need to be able to present information about libraries distinct from the code because the libraries are not produced by the application team, and thus their knowledge of their internals is less. Second, tuning popular libraries offers an opportunity to leverage any gains in performance across all applications that use those libraries.

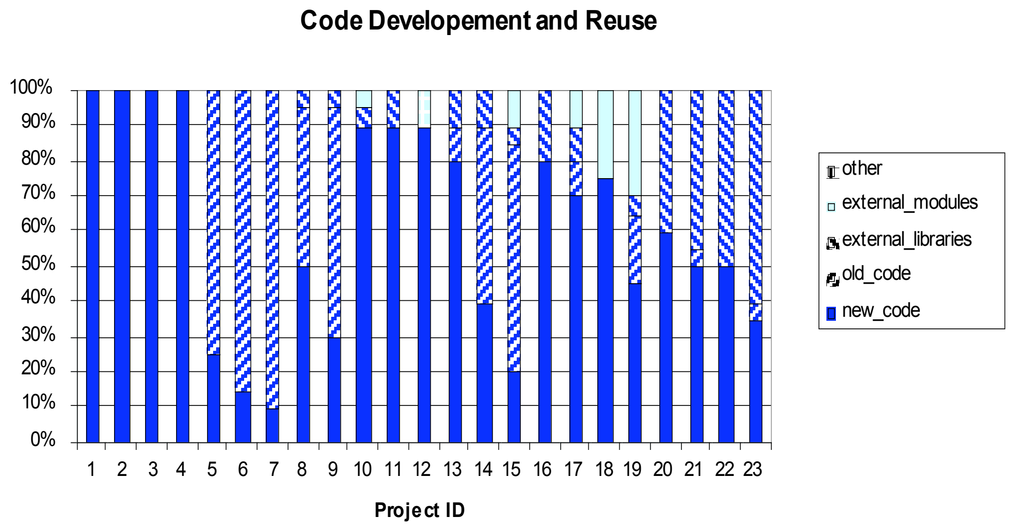
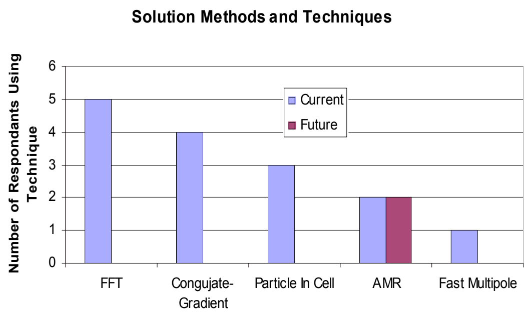
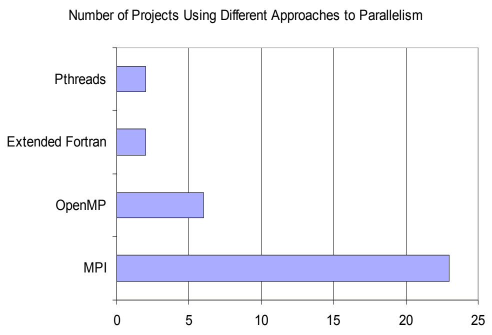


Figure 3: Code Provenance.

Because addressing some performance problems could require nontrivial code restructuring, we needed to determine the fraction of the code that is under active development by each application team. To understand this ratio, we asked each respondent to indicate what fraction of their code base is new code currently under development, and what fractions are old code, external code (not developed by the team), and any other type of code. The summary of the responses to these questions is shown in Figure 3. Just over half of the teams (12 of 23 teams) reported that more than 50% of the code in their applications is new code. For the rest of the teams, the majority is either old code that they developed previously or code from other sources. The amount of older code not under active development is an interesting challenge for tool builders because application teams may be reluctant to change this code even if it proved to be a performance problem (because they may lack familiarity with it). On the other hand, if it could be tuned, the lack of active development could mean that gains made from tuning would be stable. Alternatively, the amount of new code development also represents a challenge: we must provide performance optimization methodologies appropriate to an evolving code base. This requirement validates PERI's tool-based automated tuning approach.

Finally, we looked at the solution methods used by the teams, and the sources of parallelism. Figure 4 summarizes these results. For the solution methods, our survey included open-ended questions, and thus we suspect the numbers reported under-represent the utilization of the reported techniques. Fast Fourier Transforms (FFT) were the most commonly called-out technique, with the Conjugate Gradient iterative solver being the second most common. For parallel implementation infrastructure, as expected, the vast majority of the teams report using MPI (23 of 26 teams). However, almost one quarter of the teams (6 of 26) report using OpenMP. With the expected increase in multi-core and multi-processor nodes in future generations of leadership class architectures, this number could increase. Most of these uses of OpenMP were in addition to MPI, and thus tool builders need to think about the implications of tools for mixed mode programming models.





(a) Solution Methods Used. **(b) Types of Parallelism**

Figure 4: Techniques Used.

3. Performance goals and issues

The survey gathered information about application performance goals and issues through open-ended questions, resulting in somewhat less well-defined data than for the application characteristics. Table 1 shows the responses for performance goals. Note that a significant number of applications expressed their performance goals in terms of application-specific metrics.

Performance goal	**Number of applications citing this goal**
scaling	6 (1 1000s, 3 10000s, 2 100000s of processes)
% of peak	8 (1 20%, 1 40%, 3 50% of peak)
speedup	1 (4X)
lower overhead	1
application specific	7 (e.g., calendar years simulated/day)

Table 1. Performance goals cited

In response to the survey request to describe known performance bottlenecks, the following types of performance problems were listed: FFTs (3 responses), linear algebra (7), AMR (1), memory access (3), serial code (2), load balancing (3), interprocess communication (6), collective communication (1), graph partitioning (1), I/O (3), and code coupling (2).

4. PERI application engagement

PERI has a two-pronged application engagement strategy. Our first strategy is establishing long term *liaison* relationships with many of the application teams. PERI liaisons who work with application teams without significant, immediate performance optimization needs provide application teams with advice on how to collect performance data and track performance evolution, and ensure that PERI becomes aware of any changes in these needs. For application teams with immediate performance needs, the PERI liaison works actively with the team to help them meet their needs, utilizing other PERI personnel as needed. The level of activity for each PERI liaison changes over time as the performance needs of each application team changes. As of June 2007, PERI is working *actively* with 6 application teams and *passively* with 10 others, though the exact nature of each interaction is specific to each application team.

The other primary PERI application engagement strategy is PERI *tiger teams*. A tiger team works directly with application teams with immediate and high profile performance requirements to achieve specific, short-term goals. Our tiger teams, consisting of several PERI researchers, improve application performance by applying the full range of PERI capabilities, including not only performance modeling and automated tuning tools but also in-depth familiarity with state of the art performance analysis tools. Tiger team assignments are of a relatively short duration, lasting between 6 and 12 months. As of June 2007, PERI tiger teams are working with two application codes that will be part of the 2007 JOULE report: S3D [3] and GTC_s [4]. We have already identified significant opportunities for performance improvements for both applications. Current work is focused on providing these improvements through automated tools that support the continuing code evolution required by the JOULE criteria.

5. Conclusions and future work

While application surveys have been done by many groups, for many reasons, the PERI survey is meant to be a low-overhead mechanism to collect the information specific to the needs of PERI. A secondary benefit is that some of this information is useful to the overall Department of Energy Office of Science community. The survey has been successful, both in the number submitted and in the quality of the information collected, and has proved to be a vital tool for the efficient allocation of PERI engagement resources.

The PERI application survey is an ongoing activity. Not all SciDAC application teams have submitted surveys as of yet, and many new SciDAC application projects have or will soon start and will be invited to fill out a survey. Also, the existing surveys must be updated periodically, as they will provide the basic data used in determining how to readjust PERI engagement resources in the future.

Acknowledgements
We thank the SciDAC application developers who participated in filling out the survey for their applications, as well as the PERI team for comments on the application survey and for acting as liaisons to the SciDAC application teams. The work of all authors was supported by the Office of Mathematical, Information, and Computational Sciences, Office of Science, U.S. Department of Energy. The work of de Supinski was supported under contract No. W-7405-Eng-48 with the University of California Lawrence Livermore National Laboratory. The work of Worley was supported under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. This paper is LLNL report UCRL-CONF-231-869.

References

[1] Carver, J. C., Hochstein, L. M., Kendall, R. P., Nakamura, T., Zelkowitz, M. V., Basili, V. R., Post, D. E. "Observations about Software Development for High End Computing," CTWatch Quarterly, Vol. 2, Number 4A, November 2006 A. http://www.ctwatch.org/quarterly/ articles/2006/11/observations-about-software-development-for-high-end-computing/
[2] Performance Engineering Research Institute (PERI) website, http://www.peri-scidac.org/
[3] T. Echekki and J. H. Chen, "DNS of Autoignition in Nonhomogeneous Hydrogen-Air Mixtures," Combust. Flame 134: 169-191 (2003).
[4] W W Lee, S Ethier, W X Wang, W M Tang and S Klasky. Gyrokinetic particle simulation of fusion plasmas: path to petascale computing, J. Phys.: Conf. Ser. 46 73-81, 2006.