Ohio Supercomputer Center

REPORT ON HIGH PERFORMANCE COMPUTING TRAINING AND EDUCATION SURVEY

MAY 2009

Acknowledgements

This project is a joint effort by HPC University and the Ohio Supercomputer Center. It is aimed at providing a baseline assessment of the skills and concepts needed by computational scientists undertaking the challenging research problems of their science and engineering disciplines with the application of High Performance Computing (HPC) technology. The HPC University is a virtual organization of faculty and professionals from universities, state and federally sponsored HPC centers, HPC research laboratories, and other organizations interested in computational science education and training.

The TeraGrid Education, Outreach and Training effort provided support to compose and analyze the survey. We also are grateful to a number of people who reviewed drafts of the survey and helped us to refine its contents.

Steve Gordon and Leslie Southern
Ohio Supercomputer Center

Report on High Performance Computing Survey

MAY 2009

CONTENTS

Report on High Performance Computing Survey

MAY 2009

SUMMARY

As part of the HPC University initiative, the Ohio Supercomputer Center is helping to define the national training and education competencies for Petascale Computing. The development of the questionnaire was based on a similar effort undertaken by the PRACE collaborative in Europe (Stitt and Robinson, 2008) the TeraGrid HPC training and education Requirements Analysis Team (2008).

The survey is divided into different sections to provide information on skill sets in different areas. On each question, respondents rated the level of importance of a particular set of skills. They also were asked to rate the quality of education and training materials that exist on that topic and to provide comments on more specific information and missing items relating to each subarea.

This report describes the results of the High Performance Computing Survey as of May 18, 2009. The survey was available online at http://ww.w.rrscs.org/survey/petascale.shtml. Members of the HPC University initiative disseminated the information to researchers, developers, educators and students among a variety of disciplines in mid-April of 2009 by sending it to a number of mailing lists, electronic newsletters, and bulletin boards.

There are ten sections to the report corresponding to each of the areas in which we asked for information. They are:

1. Background and Demographics
2. Computer Programming
3. Advanced Programming and Parallel Programming
4. Numerical Libraries and Algorithms
5. Debugging, Profiling, and Optimization
6. Data Management, Parallel I/O, and Fault Tolerance
7. Scientific Visualization
8. Grid Computing
9. Education
10. Critical Skills and Development

Each of the questions asked the respondents to rate the importance of a list of skills related to that area. The level-of-importance ratings were Very Important, Somewhat Important, and Not Important. Where relevant, a fourth choice, Don't Know, also was given. Respondents could also skip questions that they did not wish to answer. To provide consistency to the ratings, all of the responses were recoded so that 1 represents the Very Important category or highest rating or ranking and 3 represents Not Important category or lowest ranking.

In some instances, where the selections are broad, respondents were asked to identify their top three choices or enter software of importance. This is the case for both programming languages and algorithms. Education skill sets are rated in the ninth section, and the tenth section focuses on critical skills that are required for students entering the field.

In addition to importance ratings for each skill set, respondents were asked to provide their appraisal of the availability of education and training materials for that topic. They were asked to select

whether good quality education and training materials exist, some quality materials exist, few quality materials exist, or unknown. The results of the findings may be found in relevant sections of the report.

Finally, people were asked to provide comments on each of the subareas. A summary of the comments is provided in the text, and all comments are provided in the appendix. The appendix also contains a copy of the complete questionnaire.

Survey results will be instrumental in defining a set of priorities for education and training activities needed to meet the requirements for Petascale computing competence. The results show that quality educational materials exist for a few categories of skills but there is no clear consensus on the availability of quality of materials in a number of areas. Assistance will be needed by the broader community to help review and document educational materials for use in a variety of instructional settings. Contributions to that effort can be made through the HPC University portal.

There are several implementation strategies for creating and delivering the appropriate educational materials that will need wider discussion across the community. The alternative approaches are discussed as they relate to the variety of skills and knowledge touched on in the survey.

BACKGROUND AND DEMOGRAPHICS

As of May 18, 2009, there were 134 respondents to the survey. The majority of respondents are in academia (84%). About 8% of the respondents were from industry and 7.5% from the government sector.

Of the academic respondents, 33% are graduate students (28.5% overall), 18% are educators from four-year institutions, 8% are research scientists, 21% are software developers and the remaining 20% are HPC facility managers and staff. Of those respondents in industry and government, 60% are software developers and 40% fall into other roles (i.e., systems architect, HPC applied programs manager, or HPC development engineer).

Seventy respondents described their expertise from a list of options. They could select as many as were appropriate. Sixty-nine percent regularly use HPC in their research, 56% develop new codes, 46% apply existing codes, 44% use multiple HPC sites/facilities, 41% add to existing codes, and 23% are just beginning to use HPC in their research.

COMPUTER PROGRAMMING

This section included importance ratings for ten skill sets for computer programming, the quality of education and training materials available for the skill sets, and 15 computer programming languages. Respondents also were able to specify other programming languages and provide comments on the section.

IMPORTANCE OF SKILLS RATINGS

Respondents were asked to rate the importance of ten computer-programming skills. The outcomes ranged from 1.10 (very important) to 2.16 (somewhat important). The ability to write codes in a programming language was rated the highest, and the ability to design human-computer interfaces was rated the lowest. The outcomes are shown below in rank order. Note that the closer the average is to 1 the more respondents felt the skill was very important.

Table 2.1 Importance Ratings for Computer Programming Skills (Survey Question #3)

Category	How Important are these skills?	Total Responses	Average Score	Standard Deviation
PROG	Ability to write code in a programming language	100	1.10	0.30
PROG	Ability to understand and interpret existing code	100	1.17	0.38
PROG	Basic debugging techniques	100	1.18	0.39
PROG	Serial/parallel program optimization	100	1.29	0.59
PROG	Basic understanding and use of numerical libraries	100	1.48	0.58
PROG	Verification and validation principles	100	1.53	0.72
PROG	Software engineering best practices	100	1.62	0.82
PROG	Understanding workflows	99	1.93	0.92
PROG	Ability to use simulators	100	2.11	0.89
PROG	Ability to design human-computer interfaces	99	2.16	0.57

Table 2.2 shows the results of this question cross-tabulated by respondent status and sector of the respondents. The top three ranked items by each group is highlighted in bold and shows general agreement across all categories with a slight difference in the third choice for industry respondents.

Table 2.2 Computer Programming Skills by Respondent Status *

	Respondent		Respondent		
Computer Programming Skills	Everyone Else	Graduate Students	Academia	Government	Industry
	97	37	113	10	11
	MEAN	MEAN	N	N	N
Ability to write code in a programming language	**1.08**	**1.14**	**1.10**	**1.00**	**1.20**
Basic debugging techniques	**1.10**	**1.39**	**1.17**	**1.11**	1.40
Ability to understand and interpret existing codes	**1.15**	**1.21**	**1.19**	**1.00**	**1.20**
Serial/parallel program optimization	1.29	1.29	1.26	1.67	**1.20**
Basic understanding and use of numerical libraries	1.38	1.75	1.47	1.67	1.40
Verification and validation principles	1.43	1.79	1.51	1.78	1.40
Software engineering best practices	1.57	1.75	1.59	1.89	1.60
Understanding workflows	1.83	2.18	1.97	1.67	1.75
Ability to design human-computer interfaces	2.13	2.26	2.18	2.00	2.20
Ability to use simulators	2.14	2.04	2.15	2.11	1.40

**Top three choices in each group shown in bold font.*

AVAILABILITY OF EDUCATION AND TRAINING MATERIALS ON PROGRAMMING

According to the respondents, good education and training materials exist for the ability to write code in a programming language. Some materials exist for the basic debugging techniques, basic understanding and use of numerical libraries, serial/parallel program optimization, and verification and validation principles. In contrast, the respondents indicated that few materials exist for the ability to design human-computer interfaces, understanding workflows, and ability to use simulators. Please note that the skills denoted with the fewest materials have the broadest distribution of responses, with nearly 40% not knowing whether or not materials existed. There were not any major differences in the rating of educational materials by sector or respondent status.

RATINGS OF PROGRAMMING LANGUAGES

Respondents were asked to rate 15 programming languages in order of importance. There also was an option provided for unknown. More than 50% of the respondents were unfamiliar with Delphi. The respondents indicated that the Fortran programming language was most important, followed closely by C, and then the Unix Shell. The ranked list of programming languages is provided as Table 2.3.

Table 2.3 Programming Language Importance Ratings

Category	How Important is it to you?	Total Responses	Average Score	Standard Deviation
PROG	Fortran	99	1.58	0.89
PROG	C	97	1.60	0.80
PROG	Unix Shell	96	1.64	0.77
PROG	C++	95	1.80	0.96
PROG	Python	94	2.18	0.95
PROG	Perl	94	2.30	0.80
PROG	Java	92	2.41	0.92
PROG	SQL	92	2.73	1.00
PROG	PHP	92	2.91	0.91
PROG	C#	86	2.93	0.84
PROG	CUDA	90	2.96	1.03
PROG	Assembly	87	3.05	0.78
PROG	Tcl	93	3.13	0.81
PROG	Ruby	92	3.16	0.84
PROG	Delphi	82	3.52	0.50

One respondent specified MATLAB as another programming language of importance. The industry respondents rated python as the third most important language. Otherwise, all groups had approximately the same rankings.

When asked to pick the top three languages of importance, the results were similar, as shown in Table 2.4. The top-ranked languages were Fortran, C, and C++, with a significant number of respondents picking other languages, the Unix Shell, and other scripting languages as their top choices.

COMMENTS ON PROGRAMMING LANGUAGES

Nineteen respondents provided comments for this section. More than 50% of the respondents noted a lack of educational courses in programming languages and limitations in those that are taught. For example,

> *computer science programs don't teach C++ and Fortran any more, so the HPC centers are having to pick up the slack.*

> *Critical lack of Fortran training classes.*

> *Poor programming languages teaching, and they expect the student born with a built in programming language.*

Students either seem to have or seem to not have programming skills. The courses taught e.g. in the physics curriculum teach at most basic concepts, but are by far not sufficient for programming in a research project.

A complete list of comments from all sections is provided in the appendix.

Table 2.4 Top Three Most Important Programming Languages

What are your top three most important programming languages?	Most important	Second most important	Third most important	Total Response	Weighted MEAN	STD
Fortran	35	15	12	62	1.6	0.8
C	26	20	12	58	1.8	0.8
C++	19	17	12	48	1.9	0.8
Tcl	1		1	2	2.0	1.4
Unix Shell	6	16	11	33	2.2	0.7
Python	5	11	9	25	2.2	0.7
SQL		4	1	5	2.2	0.4
Java	4	4	8	16	2.3	0.9
Other	1		2	3	2.3	1.2
Assembly	1	3	8	12	2.6	0.7
Perl	1	3	11	15	2.7	0.6
CUDA		1	5	6	2.8	0.4
PHP			2	2	3.0	0.0
C#				0	0.0	0.0
Ruby				0	0.0	0.0
Delphi	1			1	1.0	0.0

ADVANCED PROGRAMMING AND PARALLEL PROGRAMMING

Respondents were asked to rate the importance and availability of education materials for 17 advanced programming and parallel programming skill sets. They also were provided space for adding comments about this section. Sixty-five percent of the respondents completed this section.

IMPORTANCE RATINGS ON ADVANCED AND PARALLEL PROGRAMMING

Respondents were asked to rate the importance of 17 advanced programming skills. Four skills were rated Very Important by the majority of respondents and include the ability to understand parallel concepts/algorithms, ability to scale applications, MPI, and the ability to improve the performance of applications. Forty-five percent of those responding were unfamiliar with Single Program Multiple Data (SPMD) (i.e., UPC, CAF, Titanium) and dynamic (i.e., XIO, Fortress, Chapel, and Charm++) languages. The ranked outcomes are in Table 3.1. The tools and terms are defined in the Glossary in this report. This implies a major educational effort if those approaches to scaling applications are to gain significant use.

Table 3.1 Importance Ratings for Advanced and Parallel Programming Skills

Category	How important is it to you?	Total Responses	Average Score	Standard Deviation
APROG	Understanding parallel concepts/algorithms	87	1.23	0.50
APROG	Ability to scale applications	87	1.24	0.46
APROG	MPI	88	1.28	0.62
APROG	Ability to improve the performance of applications	87	1.31	0.51
APROG	Techniques for load balancing	87	1.56	0.69
APROG	Knowledge of memory models for parallel programming	87	1.63	0.76
APROG	Understanding parallel I/O	87	1.64	0.65
APROG	Communications including all-to-all communicationst	87	1.67	0.83
APROG	OpenMP	87	1.75	0.91
APROG	Multi-core programming	86	1.80	1.03
APROG	Mixed Mode MPI-OpenMPt	87	1.94	0.92
APROG	Many-core programming	84	1.96	1.08
APROG	Frameworks for large-scale parallel code development	86	2.05	1.09
APROG	Multi-scale modeling	85	2.19	1.10
APROG	Global arrays	86	2.41	1.03
APROG	SPMD languages (ie: UPC, CAF, Titanium)	86	3.05	1.00
APROG	Dynamic languages (ie: XIO, Fortress, Chapel, Charm++)	86	3.09	0.95

The cross-tabulation of these skills by respondent type is shown as Table 3.2. Here, one can see that there are some differences of opinion with respect to which of the skills are most important. Overall, the graduate students and everyone else agree on the two skills of *Understand parallel concepts and algorithms* and *Ability to scale applications*. However, graduate students value *Ability to improve the performance of applications* over MPI as do the industry respondents. The government respondents favor *Knowledge of memory models* as one of their top choices.

AVAILABILITY OF EDUCATION AND TRAINING MATERIALS

Fifty-four percent of the respondents indicated good materials exist for MPI, and 39% of the respondents indicated that good materials also exist for OpenMP. Nearly 55% of the respondents were unfamiliar with materials for SPMD and Dynamic languages. Thirty-seven percent were unfamiliar with materials for global arrays. An average of 35% of respondents indicated that few materials exist for mixed mode MPI-OpenMP, multi-core programming, many-core programming, and frameworks for large-scale parallel code development.

COMMENTS ON ADVANCED PROGRAMMING SKILLS

Five respondents provided comments. Two related to parallel programming. One respondent's application required the use of many single-threaded machines, and the other indicated a lack of parallel computing skills taught within computer science. Another respondent commented on a need for more good materials on advanced topics. The other comments suggested improvements to the survey. All comments for all sections are listed in the appendix.

Table 3.2 Advanced Programming and Parallel Programming Skills by Respondent

Advanced Programming and Parallel Programming Skills	Respondent		Respondent		
	Everyone else	Graduate Students	Academia	Government	Industry
	62	25	76	7	4
	MEAN	MEAN	MEAN	MEAN	MEAN
Understanding parallel concepts/algorithms	**1.19**	**1.32**	**1.24**	**1.29**	**1.00**
MPI	**1.22**	1.44	**1.30**	**1.00**	1.50
Ability to scale applications	**1.27**	**1.16**	**1.24**	1.43	**1.00**
Ability to improve the performance of applications	1.31	**1.32**	**1.30**	1.57	**1.00**
Techniques for load balancing	1.53	1.64	1.55	1.71	1.50
Knowledge of memory models for parallel programming:	1.58	1.76	1.67	**1.29**	1.50
OpenMP	1.61	2.08	1.79	1.57	1.25
Communications including all-to-all communications	1.66	1.68	1.66	1.86	1.50
Understanding parallel I/O	1.66	1.60	1.66	1.43	1.75
Multi-core programming	1.77	1.88	1.82	1.71	1.67
Mixed Mode MPI-OpenMP	1.79	2.32	1.97	1.71	1.75
Many-core programming	1.90	2.13	1.99	2.00	1.33
Frameworks for large-scale parallel code development	2.03	2.08	2.05	2.14	1.67
Multi-scale modeling	2.08	2.44	2.20	2.14	2.00
Global arrays	2.57	2.00	2.42	2.43	2.00
SPMD languages (i.e.: UPC, CAF, Titanium)	3.10	2.92	3.05	3.14	2.67
Dynamic languages (i.e.: XIO, Fortress, Chapel, Charm++)	3.11	3.04	3.14	3.14	1.67

NUMERICAL LIBRARIES AND ALGORITHMS

This section asked respondents to rate the importance and availability of education and training materials for skills associated with numerical libraries and classes of numerical algorithms. As with other sections, a space for comments also was provided. Additionally, respondents could specify numerical libraries of most interest to them.

IMPORTANCE RATINGS ASSOCIATED WITH NUMERICAL LIBRARIES

The majority of respondents rated all skills associated with numerical libraries very important. These skills include understanding of mathematics and advanced algorithms for scientific computing, understanding the principles of how to match algorithms with applications and architectures, and advanced understanding of numerical libraries and their appropriate application.

AVAILABILITY OF EDUCATION AND TRAINING MATERIALS ASSOCIATED WITH NUMERICAL LIBRARIES

The majority of respondents indicated that some materials existed for understanding of mathematics and advanced algorithms for scientific computing and advanced understanding of numerical libraries and their appropriate application; and few materials existed for understanding the principles of how to match algorithms with applications and architectures.

Table 4.1 Importance of Skills in Numerical Libraries and Algorithms

How important is it to you?	Total Responses	Average Score	Standard Deviation
Advanced understanding of numerical libraries and their appropriate application	75	1.53	0.58
Understanding of mathematics and advanced algorithms for scientific computing	75	1.36	0.54
Understanding the principles of how to match algorithms, applications, and architectures	75	1.36	0.61

NUMERICAL LIBRARIES OF IMPORTANCE

Respondents provided the names of 18 numerical libraries of importance to them. The BLAS library was listed seven times. PetSc and Lapack were listed 6 times. FFTW was listed 5 times. Scalapack and NAG were listed three times. MKL and FFT were listed twice. Other libraries listed include acml, root, clawpack, imsl, ma28, cplex, parMETIS, mass, and gsl.

IMPORTANCE RATINGS OF CLASSES OF ALGORITHMS

The majority of responses rated iterative solvers and FFT algorithms the most important. The seven other classes of algorithms were rated important. Table 4.2 shows a ranked list of the classes of algorithms.

Table 4.2 Importance Ratings for Classes of Algorithms

Category	How important is it to you?	Total Responses	Average Score	Standard Deviation
ALG	Iterative solvers	73	1.93	1.00
ALG	FFT	72	1.96	1.00
ALG	Direct solvers	72	2.15	1.00
ALG	Monte Carlo	69	2.22	1.07
ALG	Structured Grids	71	2.28	1.14
ALG	Sparse algebra	71	2.39	1.15
ALG	Unstructured Grids	69	2.45	1.13
ALG	N-body Calculations	70	2.50	1.11
ALG	Dense Algebra	70	2.53	1.06

When asked to rank the top three most important algorithmic classes, structured grids, monte carlo, and direct solvers were ranked the highest overall. However, a significant number of respondents chose one of the other categories as their highest ranks, distributing the rankings across all of the categories. We hypothesize that the need for these skills are distributed by domain topic, and thus there are no real leaders across the board. As shown in Table 4.3, there are some areas that did not attract a large number of rankings, but too much cannot be read into these results as the sample is not necessarily representative of all research domains. The cross-tabulations across respondents show a similar wide distribution of the rankings.

Table 4.3 Top Three Most Important Numerical Algorithmic Classes (Survey Question #15)

What are your top three most important numerical algorithmic classes?	Most Important	Second Most Important	Third Most Important	Total Response	MEAN	STD
Structured Grids	12	6	7	25	1.8	0.9
Monte Carlo	13	2	9	24	1.8	1.0
Direct solvers	10	13	7	30	1.9	0.8
Iterative solvers	14	16	12	42	2.0	0.8
Sparse algebra	2	4	2	8	2.0	0.8
Other	1	2	1	4	2.0	0.8
FFT	10	8	12	30	2.1	0.9
Unstructured Grids	3	7	4	14	2.1	0.7
N-body Calculations	3	7	6	16	2.2	0.8
Dense algebra	2	3	7	12	2.4	0.8

AVAILABILITY OF EDUCATION AND TRAINING MATERIALS ASSOCIATED WITH CLASSES OF ALGORITHMS

There is a technical problem with this survey question. The electronic survey allowed respondents to mark only one entry for each rating. Based on the limited responses, the fewest materials exist for dense and sparse algebra and n-body calculations.

COMMENTS

Seventeen respondents provided comments for this section and 90% of them indicated a problem with question 16. One comment related to numerical libraries and algorithms is provided below and indicates a need for additional algorithm development associated with data analytics.

> *Our need for computational numerical algorithms are rather poor. Algorithms for controlling and manipulating data and data persistency are much more important. We are also getting heavily into controlling and using distributed computing techniques.*

All comments for all sections are provided in the appendix.

DEBUGGING, PROFILING, AND OPTIMIZATION TOOLS

Respondents were asked to rate the importance and availability of education materials for seven debugging, profiling, and optimization skill sets. They also were provided space for adding comments about this section. Additionally, respondents could specify tools of importance to them.

Fifty-three percent of the respondents completed this section.

IMPORTANCE RATINGS

The average score for the seven skills rated by the respondents ranged from a high of 1.51 to a low of 1.79. All of the debugging, profiling, and optimization skills are somewhat important to the respondents. The table below shows the ranked list.

Table 5.1 Importance Ratings for Debugging, Profiling, and Optimization Tools

Category	How important is it to you?	Total Responses	Average Score	Standard Deviation
DBPFO	Ability to improve efficiency	71	1.51	0.67
DBPFO	Ability to debug parallel environments	70	1.53	0.70
DBPFO	Ability to debug memory problems	71	1.65	0.74
DBPFO	Understanding performance analysis tools	71	1.66	0.75
DBPFO	Understanding compilers and optimizations	71	1.68	0.71
DBPFO	Benchmarking techniques	71	1.72	0.70
DBPFO	Understanding debuggers	71	1.79	0.72

Table 5.2 shows the cross-tabulation by respondent for this skill set. There are differences in the ratings, but all groups rate all of these skills highly.

Table 5.2 Debugging, Profiling, and Optimization Tools by Respondent

	Respondent		Respondent		
Debugging, Profiling, and Optimization Tools	Everyone Else	Graduate Students	Academia	Government	Industry
	51 MEAN	20 MEAN	62 MEAN	7 MEAN	2 MEAN
Ability to debug parallel environments	**1.47**	**1.68**	**1.51**	**1.71**	**1.50**
Ability to improve efficiency	**1.47**	**1.60**	**1.48**	**1.71**	**1.50**
Understanding compilers and optimizations	**1.59**	1.90	1.69	**1.43**	2.00
Ability to debug memory problems	**1.59**	1.80	**1.65**	**1.71**	**1.50**
Understanding performance analysis tools	1.61	1.80	1.68	**1.71**	1.00
Understanding debuggers	1.71	2.00	1.79	**1.71**	2.00
Benchmarking techniques	1.71	**1.75**	1.69	1.86	2.00

AVAILABILITY OF EDUCATION AND TRAINING MATERIALS ON DEBUGGING AND OPTIMIZATION TOOLS

The majority of responses indicated that some materials exist for six of the seven skills. Respondents indicated that few materials exist for the ability to debug parallel environments.

COMMENTS ON DEBUGGING AND OPTIMIZATION TOOLS

Four comments were provided for this section. They are provided below:

They need computer science graduates. But nowadays most computer science curriculums are so old-fashioned; not adapted to HPC.

We do set limits on how far we can go in these areas. First the code must be portable and run without modification at other centers under other platforms. Second, the results of the code cannot be allowed to change when optimized even when on different platforms. Otherwise bug-free fast code is a high requirement.

Most users do not have expertise to optimize or profile codes

These are badly underrepresented in formal education settings, and a surprising fraction of what can be found easily is just wrong. A good set of skills to address.

TOOLS OF IMPORTANCE

Respondents were asked to indicate which tools are important to them: "Please indicate the debugging, profiling and optimization tools that are most important to you". Respondents identified 19 tools of importance. The GNU Project Debugger (GDB) was listed six times, both Totalview and TAU (Tuning and Analysis Utilities) were listed four times, GNU profiler (gprof) was listed three times. Those tools listed twice include Valgrind, mpiP, print statements, calls to timing routines, and compiler options. Other tools listed include Vampir, shark, imp, histx, hardware counters, dbx, CUDA profiler, chud, and cout.

DATA MANAGEMENT, PARALLEL I/O, AND FAULT TOLERANCE

Respondents were asked to rate the importance and availability of education materials for seven data management skill sets, four parallel I/O and fault tolerance skill sets, and parallel I/O libraries. They also were provided space for adding comments about this section.

DATA MANAGEMENT IMPORTANCE RATINGS

The average score of the data management skills ranged from 1.35 (very important) to 2.13 (somewhat important). Understanding data types and structures ranked the highest and understanding metadata ranked the lowest. The table below shows the ranked list of skills.

Table 6.1 Importance Ratings for Data Management

Category	How important is it to you?	Total Responses	Average Score	Standard Deviation
DM	Understanding data types and structures	68	1.35	0.62
DM	Understanding data distributions and movement	68	1.56	0.82
DM	Ability to identify efficient file formats	68	1.74	0.82
DM	Ability to mine and analyze data	67	1.75	0.79
DM	Efficient use of caches	68	1.79	0.96
DM	File compression	68	2.12	0.80
DM	Understanding metadata	68	2.13	0.94

Table 6.2 shows the cross-tabulation of the results for data management skills. There was general agreement on the top skill in this area but some differences in the second and third ranked skill in the category.

Table 6.2 Data Management Skills

Data Management Skills	Respondent		Respondent		
	Everyone Else	Graduate Students	Academia	Government	Industry
	49 MEAN	19 MEAN	59 MEAN	7 MEAN	2 MEAN
Understanding data types and structures	**1.45**	**1.11**	**1.37**	**1.29**	**1.00**
Understanding data distributions and movement	**1.55**	**1.58**	**1.56**	**1.57**	**1.50**
Efficient use of caches	**1.76**	1.89	1.78	2.00	**1.50**
Ability to mine and analyze data	1.79	1.63	1.78	**1.57**	**1.50**
Ability to identify efficient file formats	1.80	**1.58**	**1.75**	**1.43**	2.50
File compression	2.04	2.32	2.14	2.00	2.00
Understanding metadata	2.18	2.00	2.12	2.43	**1.50**

AVAILABILITY OF EDUCATION AND TRAINING MATERIALS FOR DATA MANAGEMENT

The respondents indicated that good quality materials are available for understanding data types and structures; some quality materials exist for the ability to identify efficient file formats, understanding data distributions and movement ability to mine and analyze data, understanding metadata, and file compression; and few quality materials are available for efficient use of caches.

COMMENTS ON DATA MANAGEMENT

One respondent indicated that ROOT, an object-oriented program and library developed by CERN, included many data management capabilities and addressed many of the skills.

IMPORTANCE RATINGS FOR PARALLEL I/O AND FAULT TOLERANCE

Respondents were asked to rate four major skills relating to parallel I/O and fault tolerance. They rated the four skill sets as somewhat important on average (i.e., 1.69-2.25). Understanding how to implement parallel I/O was rated highest, understanding how to implement checkpointing was next, then understanding network topology, followed by understanding fault tolerance.

AVAILABILITY OF EDUCATION AND TRAINING MATERIALS FOR PARALLEL I/O AND FAULT TOLERANCE

Respondents indicated that some quality materials existed for understanding how to implement parallel I/O. For the other skill sets, the majority of responses were unfamiliar with materials on those topics.

IMPORTANCE RATINGS FOR PARALLEL I/O LIBRARIES

Respondents were asked to rate the importance of HDF5, Parallel NetCDF, and MPI-IO libraries. Nearly 50% of those responding were unfamiliar with the libraries. MPI-IO was rated the highest at 2.61, followed by Parallel NetCDF at 2.94, and then HDF5 at 2.98.

AVAILABILITY OF EDUCATION AND TRAINING MATERIALS FOR PARALLEL I/O LIBRARIES

Fifty-seven percent of those responding were unfamiliar with quality materials for any of the parallel I/O libraries listed. Twenty percent of those responding indicated that there were some quality materials available for the listed libraries.

COMMENTS

One respondent questioned the need for fault tolerance skills by users and developers.

SCIENTIFIC VISUALIZATION

Respondents rated the availability of quality education and training materials and importance of four scientific visualization skills and had the option to provide comments. Respondents also were asked to list scientific visualization tools of interest.

IMPORTANCE RATINGS

Since the importance ratings ranged from 1.67 to 2.32, all skills were deemed somewhat important. The ranked list is in the table below

Table 7.1 Importance Ratings for Scientific Visualization

Category	How Important is it to you?	Total Responses	Average Score	Standard Deviation
VIZ	Understanding of visualization tools	67	1.67	0.81
VIZ	Understanding the differences between surface and volume rendering	67	2.19	1.08
VIZ	Bottlenecks to visualization	67	2.19	1.06
VIZ	Ability to use visualization APIs	66	2.32	1.04

There was general agreement in the rankings by respondent groups for these questions.

AVAILABILITY OF EDUCATION AND TRAINING MATERIALS

Forty-six percent of the responses for understanding visualization tools indicated that some materials existed. Most of the respondents were unfamiliar with materials for the other skills.

VISUALIZATION TOOLS OF IMPORTANCE

VisIt was identified four times. VTK, OpenGL, and IDL were each identified twice. Other tools listed include MATLAB, ROOT, CUDA, GLSL, NCL, Vapor, Amira, AVS Express, Paraview, vmd, and pymol.

COMMENTS

Two respondents provided comments. They are listed in the appendix.

GRID COMPUTING

Respondents were asked to rate the importance of three grid computing skill sets. They also were provided space for adding comments about this section.

IMPORTANCE RATINGS FOR GRID COMPUTING

The importance rating for grid computing skills ranged from 2.26 (somewhat important) to 2.51 (not important). The ranked list is in Table 8.1.

Table 8.1 Importance Ratings for Grid Computing

Category	How important is it to you?	Total Responses	Average Score	Standard Deviation
GRID	Ability to use a grid environment	68	2.26	1.13
GRID	Understanding security mechanics	68	2.35	1.03
GRID	Understanding Grid middleware	68	2.51	1.07

AVAILABILITY OF EDUCATION AND TRAINING MATERIALS

The majority of the responses for understanding security mechanics and grid middleware were unaware of education and training materials. Thirty-eight percent of the responses for ability to use a grid environment indicated some quality materials exist.

COMMENTS

Three respondents provided comments. One was an early adopter of grid computing, and two indicated a limited need for these skill sets.

EDUCATION

Respondents were asked a set of questions about the more general education and professional skills they would like to see in their students. Respondents were asked to rate the importance of five such skills in education. They also were provided space for adding comments about this section.

IMPORTANCE RATINGS FOR EDUCATIONAL SKILLS

The importance ratings for the education skill sets ranged from 1.35 (very important) to 1.59 Table 9.1 shows the rankings. The most important skill set indicated is the development of writing skills for research papers, grants, and presentations. This is followed by mentoring skills, good research practices, and collaborative research. Interestingly, the development of effective teaching pedagogy is rated the lowest, although the average score is still quite high.

Table 9.1 Importance Ratings for Educational Skills

Category	How important is it to you?	Total Responses	Average Score	Standard Deviation
ED	Developing professional skills in writing top-quality research papers, giving effective presentations, grant writing, etc.	68	1.35	0.62
ED	Developing mentoring skills to be able to assist others in advancing their studies and research	68	1.44	0.68
ED	Good research practices (e.g., working on large teams and multi-year projects)	67	1.46	0.77
ED	Developing teamwork skills in large, distributed, multi-institutional, multi-cultural, global research team environments	68	1.54	0.99
ED	Effective teaching pedagogy	68	1.59	0.70

Cross tabulations show the results across respondents are pretty similar, with one exception by the government respondents as shown in Table 9.2

Table 9.2 Educational Skills

	Respondent		Respondent		
Educational Skills	Everyone Else	Graduate Students	Academia	Government	Industry
	49 MEAN	18 MEAN	59 MEAN	6 MEAN	2 MEAN
Developing professional skills in writing top-quality research papers, giving effective presentations, grant writing, etc.	1.41	1.21	1.35	1.33	1.50
Good research practices (e.g., working on large teams and multi-year projects)	1.48	1.42	1.41	2.00	1.50
Developing mentoring skills to be able to assist others in advancing their studies and research	1.49	1.32	1.45	1.33	1.50
Developing teamwork skills in large, distributed, multi-institutional, multi-cultural, global research team environments	1.57	1.47	1.55	1.50	1.50
Effective teaching pedagogy	1.63	1.47	1.57	1.67	2.00

CRITICAL SKILLS AND DEVELOPMENT

Respondents then were asked to rate the importance of skill sets for different levels of educational attainment, rate the sections of the survey based on critical needs, and rate each topic with respect to the how much development is needed. Respondents also were asked to identify their domain-specific applications of interest and provide comments.

IMPORTANCE RATINGS FOR CRITICAL SKILLS AND DEVELOPMENT

Respondents were asked to rate the importance of basics in modeling and simulation principles, programming and algorithm basics and mathematics requirements for undergraduates and graduate/professionals. All were rated very important. For undergraduates, the importance ratings ranged from 1.29 to 1.49, with mathematics requirements rated highest. For graduates and professionals, the importance ratings ranged from 1.15 to 1.21, with programming and algorithm basics rated highest. The rankings are listed in the table below. The UGRAD category indicates the ratings for undergraduates, and GRADP are the ratings for graduates and professionals.

Table 10.1 Importance Ratings for Critical Skills for Undergraduate and Graduate Students

Category	How important is it to you?	Total Responses	Average Score	Standard Deviation
UGRAD	Mathematics requirements	68	1.29	0.57
UGRAD	Programming and algorithm basics	68	1.31	0.58
UGRAD	Basics of modeling and simulation principles	68	1.49	0.63
GRADP	Programming and algorithm basics	68	1.15	0.40
GRADP	Mathematics requirements	68	1.19	0.43
GRADP	Basics of modeling and simulation principles	68	1.21	0.44

CRITICAL NEED AND DEVELOPMENT

Respondents were asked to rate the main topics based on critical need and the need of development. Results are shown in Tables 10.2 and 10.3.

Table 10.2 Which Skill Sets Are Most Critical?

Critical Need?	Total Responses	Average Score	Standard Deviation
Programming	67	1.22	0.49
Advanced Programming and Parallel Programming	65	1.65	0.65
Numerical Libraries and Algorithms	64	1.80	0.60
Data Management	65	1.82	0.66
Education	65	1.89	0.90
Scientific Visualization	67	1.90	0.74
Parallel I/O and Fault Tolerance	64	2.11	0.82
Grid Computing	64	2.28	0.84

Table 10.3 Which Skill Sets Need the Most Development?

Skill Set	Total Responses	Average Score	Standard Deviation
Advanced Programming and Parallel Programming	65	1.38	0.65
Programming	67	1.75	0.66
Numerical Libraries and Algorithms	64	1.86	0.75
Data Management	62	1.89	0.89
Scientific Visualization	66	1.97	1.04
Parallel I/O and Fault Tolerance	63	2.00	1.14
Education	63	2.03	1.14
Grid Computing	65	2.15	1.05

Most responses for Computer Programming indicated there was a very critical need (81%) and some development (55%) was required. For Advanced Programming and Parallel Programming respondents indicated there was a somewhat critical need (46%) and also indicated that this area requires the most development (68% of respondents). Most responses for Numerical Libraries and Algorithms indicated there was a somewhat critical need (61%) and some development (56%) was required. The other categories were ranked lower for both critical need and development, but all averaged at least at the level of somewhat critical.

SPECIFIC-DOMAIN APPLICATIONS

Seventeen respondents identified their top three most important domain-specific applications. The outcomes are in Table 11.1

Table 11.1 Most Important Domain-Specific Applications

#1 Most Important App	***#2 Most Important App***	***#3 Most Important App***
Education	***Scientific Visualization***	***Grid Computing***
electromagnetics simulation in parallel architectures	Gridding techniques	Scientific visualization
There is no one such application. By using Grid computing, we rely on standard libraries and home grown libraries which support the user in the development of analysis/visualization specific code.		
MATLAB	gcc	
Fourier imaging (CUDA)	Monte Carlo	
NWChem	My own codes	None
Prototyping platforms for numerical analysis, e.g. MATLAB, SciPy	Finite element analysis software	Numerical optimization platforms
Geophysical fluid dynamics models		
Numerical weather prediction	Climate	Atmospheric chemistry
FLASH		
CFD codes	Astrophysics simulations	Gaussian
VASP	Gaussian	DLPOLY
Particle physics	Molecular dynamics	Data analysis
C++	MATLAB	MPI and/or OpenMP
openfoam	Fluent	Overflow
N-body simulation in astrophysics	Large data set reduction	Visualization and interpretation
Cactus		

IMPLICATIONS FOR FUTURE EDUCATIONAL DEVELOPMENT

Responses to the survey will be instrumental in helping to define a set of priorities for the education and training activities needed to meet the requirements for Petascale computing competence. Defining the priorities and the competencies will require further research as well as a discussion with the community about the most productive strategies for implementation. The HPC University effort welcomes community participation in this process – please contact us to discuss this further.

The major research question posed by the survey results is the actual status of quality educational and training materials dealing with the most important issues. For a few areas such as MPI and data structures, there is a clear consensus that quality educational materials already exist. Those materials are being actively sought from among the community to be referenced in the HPC University portal for the community at large (http://www.hpcuniv.org/). We encourage the community to share this information by posting references directly to the HPC University portal or by sending information to the authors of this paper. For most of the other areas of need, there is not a clear consensus on the availability of quality materials. This may be because they actually do not exist but for some areas could equally be to the lack of knowledge of the respondents. For each of those areas, a more in-depth search for such materials must be conducted and evaluated. Community contributions to fill out this collection are needed and welcome. This search should include a review of text books, industry produced documentation and tutorials, and training materials produced and used by high performance computing centers at the Federal, State, and university levels. As materials are identified, there needs to be a consistent review process that provides an evaluation of the materials and their suitability for use in a variety of instructional settings. The HPC University portal provides both an informal and a formal review process to help ensure the community is accessing quality materials.

Where major gaps in important areas are found among the available materials, strategies to create such materials must be put into place. These must be coupled with a review process as well as accreditation that the materials are suitable for particular audiences.

In either case, implementation strategies for fielding the instructional materials must be defined. Multiple strategies are possible. There are some parts of the materials that are oriented toward the use of particular tools or techniques that may have a relatively short lifetime as those tools and the systems they complement are modified. For these kinds of materials, the best strategy may be to sponsor short-term training workshops, prepare self-paced, online instructional guides, and provide short-term assistance to those using the materials through the current support infrastructure at HPC centers.

There is another subset of materials that focuses on conceptual principles and practices where there is a longer-term legacy of developments in mathematics, computer science, and science and engineering discipline research on the topics. Examples include database design and management principles, software engineering best practices, knowledge and use of appropriate algorithms, and model validation and verification. These principles are most productively taught in a more traditional course setting, not only a traditional classroom, but in courses that last over a semester and allow time for a greater depth of understanding and the building of expertise. Implementation strategies for these skills will need to create incentives for faculty to create and use the educational materials, to incorporate portions of the materials into their existing courses, and to create new courses in several areas. Since a number of the subtopics are quite specialized, the problem will be in creating a critical mass of faculty and students to make the instruction financially feasible in light of current teaching loads for disciplinary courses. Here, the examples of shared instruction across departments and institutions offered in Ohio by the Ralph Regula School of Computational Science (RRSCS, 2009) or the emerging Virtual School of Computational Science and Engineering formed by the Great Lakes Consortium for Petascale Computation (GLCPC, 2009) offer good models for implementation strategies. The RRSCS has developed a set of competencies for undergraduate computational science (RRSCS, 2009a). A similar set of competencies for graduate level programs are under development by both of these organizations.

There is a third subset of materials that in some ways falls between the first two in terms of the focus on techniques versus general principles and best practices. These are topics that are subsets of traditional mathematics and computer science education materials that need to be introduced to scientists and engineers with a different focus and approach. Here, one could include instruction in computer programming languages, advanced data management and parallel file systems work, aspects of numerical methods, differential equations, statistics, and optimization techniques. Although there may be existing courses that cover these topics, they often emphasize theoretical approaches, have prerequisite structures aimed at graduate majors in the discipline, and do not draw examples that apply to high performance computing. Universities in general have not been successful at bridging the gaps in interdisciplinary instruction, leading to what could be described as a chaotic mix of approaches. These can range from successful service courses taught by disciplinary experts to one-of-a-kind, narrow courses taught by faculty from a science or engineering discipline that focuses only on their few students. Courses such as “statistics for biologists” or “numerical methods for engineers” emerge without sufficient, long-term commitment in the curriculum to be effective.

Some of the same strategies mentioned above might help to resolve these problems. The availability of excellent, peer-reviewed instructional materials can make it easier for faculty to adopt those materials into their courses. Sharing instruction across disciplines and institutions can create the critical mass necessary to make a set of specialized courses more viable as long-term parts of the curriculum and teaching loads. Some aspects of these topics might be covered in short-term workshops or through online, self-paced courses.

Regardless of which subset of the materials is being addressed, it is clear that significant efforts will be required to fill the gaps in current education and training materials to prepare present and future generations of computational scientists to take advantage of Petascale computing resources. As reflected in several national reports (NSF, 2007; SBER, 2006) over the past several years, this will mean significant investment of resources by the federal agencies, HPC centers, and the universities.

We welcome your suggestions and your participation in this effort to enhance petascale training and education.

References Cited

Great Lakes Consortium for Petascale Computation, 2009. http://www.greatlakesconsortium.org/index.html.

National Science Foundation, 2007. "Cyberinfrastructure Vision for 21st Century Discovery", NSF 07-28, Washington, D.C.: March 2007

Ralph Regula School of Computational Science, 2009. http://www.rrscs.org.

Ralph Regula School of Computational Science, 2009a. Summary of Undergraduate Minor Program Requirements, http://www.rrscs.org/minor/competencyfinal.pdf.

SBER 2006. Simulation Based-Engineering Science: Report of the NSF Blue Ribbon Panel on Simulation-Based Engineering Science, NSF, February 2006. Available online at http://www.ices.utexas.edu/events/SBES_Final_Report.pdf.

Stitt and Robinson, "A Survey on Training and Education Needs for Petascale Computing", 2008. http://www.prace-project.eu/documents/D3.3.1_document_final.pdf.

TeraGrid HPC training and education Requirements Analysis Team, 2008. Unpublished report.

Appendix Materials

APPENDIX 1 – SURVEY COMMENTS

Comments about Computer Programming

1 computer science programs don't teach C++ and Fortran any more, so the HPC centers are having to pick up the slack.

2 Critical lack of Fortran training classes.

3 Due to the current outsourcing of programming projects, programming education is suffering nowadays.

4 Each programming language is upon each programmer's personality as from http://www.bbspot.com/News/2006/08/language_quiz.php I myself like prologue up to the link.

5 For most of the users in my field, I see a lot of C++ code written with Fortran form. (Fortran with ";" :-). There are a number of people who come in and write elaborate C++ which few can follow or understand. This make code maintainence a real problem. Between these problems and our need to have very portable code, we find it necessary to restrict some of C++'s capabilities.

6 I don't think the core understanding of central algorithms and analysis techniques is yet mature enough in parallel/hybrid/distributed domains that designing a curriculum will be easy. You can (and should) teach tools, but there is no equivalent of Knuth to drive fundamentals. So far, they are provisionary instead of grounded in proofs.

7 I have a decent programming skill set, but I would like to expand it further to optimize my programs for running on parallel processors and High Performance Computers.

8 I see a tendency toward giving somewhat of a cult status to selected programming languages. I.e., too large emphasis is placed on the (perceived/claimed) virtues of one's favorite language at the expens of establsihing fundamental and (nearly) universal principles.

9 Lack of good introductory courses to HPC languages

10 Lamost no formal (for credit) courses exist in programming skills. There are lots of short workshops.

11 Little guidance of best practices when writing large scientific applications with multiple developers.

12 Lots of material exists for programming in general but very little is available for high-performance numerical codes. There is Goedecker & Hoise and Dowd & Severance but neither is complete. Sadly, the people who are qualified these books are too busy to write them.

13 MPI

14 My greatest difficulty has always been determining how to use various numerical libraries, especially when problems arise linking and when it is appropriate to ask questions vs. RTFMing.

15 Poor programing languages teaching , and they expect the student born with a built in programing language..

16 Since FORTRAN is still most used language in scientific large-scale computations, I would like codes in FORTRAN version for all C and C++ codes.

17 Students either seem to have or seem to not have programming skills. The courses taught e.g. in the physics curriculum teach at most basic concepts, but are by far not sufficient for programming in a research project.

18 There is a need for training in high performance functional languages

19 Very little training is available on Fortran 90

Comments about Advanced Programming and Parallel Programming

1 Even most PhD.'s in Computer Science or Informatics still do not know anything about parallel computing.

2 In my work. Parallel programing is not important. We have only trivial parallel needs. We do have very large data sets which need to be analyzed using large clusters of single threaded machines.

3 You might want to define, ie disambiguate, Multi-core and Many-core next time.

4 Much material exist to learn the basics. But I don't know of good material that goes behind the basics and to the "real" problems/bottlenecks: adaptation of the code to the machine architecture, i.e. shared memory access, caches sizes etc. It is hard to find for example the number of cycles necessary to access the L2 cache on a specific CPU.

You need an extra category. You pass straight from good materials exist to some materials exist. There are cases where many materials are available, but I know of none I would call good. The first category is about quality, while the other two are about quantity. There are darn few materials in this field I consider high quality. 5

Comments about Numerical Libraries and Algorithms

1 Chinese have numerical skills maybe the best in the World, maybe because their language are symbolic like math.

2 The radio buttons of #16 are broken (won't let me select the same column values for more than row)

3 Our need for computational numerical algorithms are rather poor. Algorithms for controlling and manipulating data and data persistency are much more important. We are also getting heavily into controling and using distributed computing techniques.

4 You mixed the behavior of 15 and 16 such that I cannot answer 16 properly whereas I am permitted to answer 15 improperly (but did not do so).

5 Question 16 was column restricted instead of row restricted thus it cannot be answered correctly.

6 I believe Q16's check buttons have not been implemented correctly, or I dont understand the question since it only lets me pick one button per column and that doesnt make sense to me.

7 Question 16, above, is broken. One can only select one box in each column.

8 Question 16 doesn't permit picking appropriate responses, so the selections are not valid

9 Table 16 could not be properly filled in (no multiple checks per column possible).

10 Above table is flawed, only lets you put one answer in per column

11 NOTE: ITEM 16 DOES NOT ALLOW ONE TO CHOOSE THE SAME OPTION (GOOD/SOME/FEW) MULTIPLE TIMES, I.E., YOU CANT CHOOSE (GOOD) FOR DIFFERENT TOPICS...

12 Bug in #16, cannot enter my answer

13 Q16 does not allow multiple selection(?)

14 I assume in my answers you are referring to parallel versions of the above. There are good materials for all, in the sequential domain. Actually, you have a setting problem on the multiple select, so I can't answer directly. Some materials exist for each, but nothing I would call good.

15 I cannot fill in the table for question 16; the dependencies between the rows and columns are set up wrong.

16 The buttons on question 6 work like "radio" butons - I am allowed to check one item per column

17 This question is broken; can't answer more than 1.

Comments about Debugging, Profiling, and Optimization Tools

1 They need computer science graduates. But nowadays most computer science curriculums are so old-fashined; not adapted to HPC.

2 We do set limits on how far we can go in these arias. First the code must be portable and run without modification at other centers under other platforms. Second, the results of the code can not be allowed to change when optimized even when on different platforms. Otherwise bug-free fast code is a high requirement.

3 Most users do not have expertise to optimize or profile codes

4 These are badly underrepresented in formal education settings, and a surprising fraction of what can be found easily is just wrong. A good set of skills to address.

Comments about Data Management

1 based on programming language course and experiences in programming.

2 We are lucky to have very good relations to the software developers of root. They have developed some of the best methods for all of these issues.

Comments about Parallel I/O and Fault Tolerance

1 Should know C.

2 Fault tolerance is important, but the questions above relate to how well these have to be understood by users/developers. Should be automatic/transparent.

3 A weakness for me, and so not my most useful answers.

Comments about Scientific Visualization

1 should use Photonics technology; as that in Time Machine movie.

2 Visualization is something one worries about after the application is running well. It should not be taught as primary material.

Comments regarding Grid Computing

1 Google use networking of some thousands of personal computers; no need of supercomuter.

2 We are very much involved in Grid development and are an early user of these tools.

3 Grid Computing is completely inaccessible to the average or advanced user. The arcane certificate mechanism would scare off anybody.

Comments regarding Education

1 They need to know a big picture; because delving down into details will blunt them sometime as for unlimited work hours. But work brings skills any way.

2 Most students will never need these skills because they can't program and thus will never be in charge of a project (or at least shouldn't be)

3 How is working on large teams on multi-year projects a good research practice? Or a bad one? Seems orthogonal.

4 The only reason I downgraded pedagogy is because those who do not work in academia has far less need of it. Inside academia, it is crucial.

Comments regarding Critical Skills

1 Artistic or right-side brain from art, excercise, meditation or orintal culture may bring innovations or breakthroughs.

2 lumping parallel I/O and fault tolerance together is a disservice to both, but especially fault tolerance.

3 Software skills are usually poor, especially for a programing language like C++. Only time and experience will solve this, unfortunately.

4 Computer architecture, particularly memory hierarchies, and C-level programming are prerequisites for everything else. Anyone who doesn't know those two things is wasting there time trying to learn the rest.

5 Question 41. Critical means "indispensable, essential." Something's either critical or it isn't. Therefore, some items I checked as not critical I still consider important.

APPENDIX 2 – SURVEY INSTRUMENT

2009 Defining Competencies for Petascale Computing

Pg 1 High Performance Computing Survey

We are working to address workforce needs for petascale computing with an eye towards exascale and beyond for a long-term view of workforce development needs for academia, industry, government and society. The purpose of this questionnaire is to help define the scope of the skills required and to document plans for training and education programs aimed at meeting them. The questionnaire will take 20-30 minutes to complete. Results will be publicly shared and used to help formulate training and education programs for Petascale computing.

1 **Please specify the sector that you best represent.**

- o Academia
- o Industry
- o Government

2 **Please specify which term best describes your current position.**

- o Undergraduate student
- o Graduate student
- o Educator at associate's college
- o Educator at doctorate-granting university
- o Educator at master's colleges and universities
- o Educator at baccalaureate college
- o Educator at tribal college
- o Postdoc
- o Research scientist
- o Software developer
- o HPC facility manager
- o HPC facility staff
- o Other, please specify

For each of the eleven following areas, please rank their importance to your research. You are also invited to provide comments for each area.

Pg 2 Computer Programming Skills

3 **How important are these skills?**	**Not Important**	**Somewhat important**	**Very Important**	**Don't Know**
Ability to write code in a programming language	o	o	o	o
Ability to understand and interpret existing codes	o	o	o	o
Basic debugging techniques	o	o	o	o
Basic understanding and use of numerical libraries	o	o	o	o
Software engineering best practices	o	o	o	o
Verification and validation principles	o	o	o	o
Ability to design human-computer interfaces	o	o	o	o
Understanding workflows	o	o	o	o
Ability to use simulators	o	o	o	o
Serial/parallel program optimization	o	o	o	o

4	**Are quality education and training materials available on these topics?**	**Good Materials Exist**	**Some Materials Exist**	**Few Materials Exist**	**Don't Know**
	Ability to write code in a programming language	o	o	o	o
	Ability to understand and interpret existing codes	o	o	o	o
	Basic debugging techniques	o	o	o	o
	Basic understanding and use of numerical libraries	o	o	o	o
	Software engineering best practices	o	o	o	o
	Verification and validation principles	o	o	o	o
	Ability to design human-computer interfaces	o	o	o	o
	Understanding workflows	o	o	o	o
	Ability to use simulators	o	o	o	o
	Serial/parallel program optimization	o	o	o	o
5	**Rate the importance of programming languages**	**Most Important**	**Important**	**Somewhat Important**	**Don't Know**
	Assembly	o	o	o	o
	C	o	o	o	o
	C#	o	o	o	o
	C++	o	o	o	o
	Delphi	o	o	o	o
	Fortran	o	o	o	o
	Java	o	o	o	o
	Perl	o	o	o	o
	PHP	o	o	o	o
	SQL	o	o	o	o
	Python	o	o	o	o
	Ruby	o	o	o	o
	Tcl	o	o	o	o
	Unix Shell	o	o	o	o
	CUDA	o	o	o	o

6	**What are your top 3 most important programming languages?**	**Most Important**	**2nd Most Important**	**3rd Most Important**
	Assembly	o	o	o
	C	o	o	o
	C#	o	o	o
	C++	o	o	o
	Delphi	o	o	o
	Fortran	o	o	o
	Java	o	o	o
	Perl	o	o	o
	PHP	o	o	o
	SQL	o	o	o
	Python	o	o	o
	Ruby	o	o	o
	Tcl	o	o	o
	Unix Shell	o	o	o
	CUDA	o	o	o
	Other	o	o	o
	If "Other", please specify:			
7	**If you would like to comment on programming skills and or the current state of training on programming skills, please do so here:**			

Pg 3 *Advanced Programming and Parallel Programming*

8	**How important are these skills?**	**Not Important**	**Somewhat Important**	**Very Important**	**Don't Know**
	Understanding parallel concepts/algorithms	o	o	o	o
	Ability to scale applications	o	o	o	o
	Ability to improve the performance of applications	o	o	o	o
	Techniques for load balancing	o	o	o	o
	Communications including all-to-all communications	o	o	o	o
	Understanding parallel I/O	o	o	o	o
	Knowledge of memory models for parallel programming	o	o	o	o
	MPI	o	o	o	o
	OpenMP	o	o	o	o
	Mixed Mode MPI-OpenMPt	o	o	o	o
	SPMD languages (i.e.: UPC, CAF, Titanium)	o	o	o	o
	Dynamic languages (i.e.: XIO, Fortress, Chapel, Charm++)	o	o	o	o
	Global arrays	o	o	o	o
	Multi-core programming	o	o	o	o
	Many-core programming	o	o	o	o
	Frameworks for large-scale parallel code development	o	o	o	o
	Multi-scale modeling	o	o	o	o

9	**Are quality education and training materials available on these topics?**	**Good Materials Exist**	**Some Materials Exist**	**Few Materials Exist**	**Don't Know**
	Understanding parallel concepts/algorithms	o	o	o	o
	Ability to scale applications	o	o	o	o
	Ability to improve the performance of applications	o	o	o	o
	Techniques for load balancing	o	o	o	o
	Communications including all-to-all communications	o	o	o	o
	Understanding parallel I/O	o	o	o	o
	Knowledge of memory models for parallel programming	o	o	o	o
	MPI	o	o	o	o
	OpenMP	o	o	o	o
	Mixed Mode MPI-OpenMPt	o	o	o	o
	SPMD languages (i.e.: UPC, CAF, Titanium)	o	o	o	o
	Dynamic languages (i.e.: XIO, Fortress, Chapel, Charm++)	o	o	o	o
	Global arrays	o	o	o	o
	Multi-core programming	o	o	o	o
	Many-core programming	o	o	o	o
	Frameworks for large-scale parallel code development	o	o	o	o
	Multi-scale modeling	o	o	o	o

10 **If you would like to comment on advanced programming skills, please do so here:**

Pg 4 ***Numerical Libraries and Algorithms***

		Not Important	Somewhat Important	Very Important	Don't Know
11	**How important are these skills?**				
	Advanced understanding of numerical libraries and their appropriate application	o	o	o	o
	Understanding of mathematics and advanced algorithms for scientific computing	o	o	o	o
	Understanding the principles of how to match algorithms, applications, and architectures	o	o	o	o

		Good Materials Exist	Some Materials Exist	Few Materials Exist	Don't Know
12	**Do quality education and training materials exist for these topics?**				
	Advanced understanding of numerical libraries and their appropriate application	o	o	o	o
	Understanding of mathematics and advanced algorithms for scientific computing	o	o	o	o
	Understanding the principles of how to match algorithms, applications, and architectures	o	o	o	o

13 **Please indicate the numerical libraries that are most important to you.**

		Most Important	Important	Somewhat Important	Don't Know
14	**Rate the importance of numerical algorithmic classes.**				
	Direct solvers	o	o	o	o
	Iterative solvers	o	o	o	o
	Dense algebra	o	o	o	o
	Sparse algebra	o	o	o	o
	Structured Grids	o	o	o	o
	Unstructured Grids	o	o	o	o
	Monte Carlo	o	o	o	o
	FFT	o	o	o	o
	N-body Calculations	o	o	o	o

		Most Important	2nd Most Important	3rd Most Important
15	**What are your top 3 most important numerical algorithmic classes?**			
	Direct solvers	o	o	o
	Iterative solvers	o	o	o
	Dense algebra	o	o	o
	Sparse algebra	o	o	o
	Structured Grids	o	o	o
	Unstructured Grids	o	o	o
	Monte Carlo	o	o	o
	FFT	o	o	o
	N-body Calculations	o	o	o
	Other	o	o	o

If "Other", please specify.

16	Do quality education and training materials exist for these topics?	Good Materials Exist	Some Materials Exist	Few Materials Exist	Don't Know
	Direct solvers	o	o	o	o
	Iterative solvers	o	o	o	o
	Dense algebra	o	o	o	o
	Sparse algebra	o	o	o	o
	Structured Grids	o	o	o	o
	Unstructured Grids	o	o	o	o
	Monte Carlo	o	o	o	o
	FFT	o	o	o	o
	N-body Calculations	o	o	o	o
	Other	o	o	o	o

17 If you would like to comment on numerical skills, please do so here:

Pg 5 ***Debugging, Profiling, and Optimization Tools***

18	How important are these skills?	Not Important	Somewhat Important	Very Important	Don't Know
	Understanding compilers and optimizations	o	o	o	o
	Understanding debuggers	o	o	o	o
	Ability to debug memory problems	o	o	o	o
	Ability to debug parallel environments	o	o	o	o
	Understanding performance analysis tools	o	o	o	o
	Ability to improve efficiency	o	o	o	o
	Benchmarking techniques	o	o	o	o

19	Do quality education and training materials exist for these topics?	Good Materials Exist	Some Materials Exist	Few Materials Exist	Don't Know
	Understanding compilers and optimizations	o	o	o	o
	Understanding debuggers	o	o	o	o
	Ability to debug memory problems	o	o	o	o
	Ability to debug parallel environments	o	o	o	o
	Understanding performance analysis tools	o	o	o	o
	Ability to improve efficiency	o	o	o	o
	Benchmarking techniques	o	o	o	o

20 If you would like to comment on debugging, profiling and optimization skills, please do so here:

21 Please indicate the debugging, profiling and optimization tools that are most important to you.

Pg 6 ***Data Management***

22	How important are these skills?	Not Important	Somewhat Important	Very Important	Don't Know
	Understanding data types and structures	o	o	o	o
	Ability to identify efficient file formats	o	o	o	o
	Understanding data distributions and movement	o	o	o	o
	Ability to mine and analyze data	o	o	o	o
	Understanding metadata	o	o	o	o
	File compression	o	o	o	o
	Efficient use of caches	o	o	o	o

23	**Do quality education and training materials exist for these topics?**	**Good Materials Exist**	**Some Materials Exist**	**Few Materials Exist**	**Don't Know**
	Understanding data types and structures	o	o	o	o
	Ability to identify efficient file formats	o	o	o	o
	Understanding data distributions and movement	o	o	o	o
	Ability to mine and analyze data	o	o	o	o
	Understanding metadata	o	o	o	o
	File compression	o	o	o	o
	Efficient use of caches	o	o	o	o

24 **If you would like to comment on data management skills, please do so here:**

Pg 7 ***Parallel I/O and Fault Tolerance***

25	**How important are these parallel i/o and fault tolerance skills?**	**Not Important**	**Somewhat Important**	**Very Important**	**Don't Know**
	Understanding how to implement parallel i/o	o	o	o	o
	Understanding how to implement checkpointing	o	o	o	o
	Understanding network topology	o	o	o	o
	Understanding fault tolerance	o	o	o	o

26	**Do quality education and training materials exist for these topics?**	**Good Materials Exist**	**Some Materials Exist**	**Few Materials Exist**	**Don't Know**
	Understanding how to implement parallel i/o				
	Understanding how to implement checkpointing				
	Understanding network topology				
	Understanding fault tolerance				

27	**Rate the importance of parallel i/o libraries.**	**Not Important**	**Somewhat Important**	**Very Important**	**Don't Know**
	HDF5	o	o	o	o
	Parallel NetCDF	o	o	o	o
	MPI-IO	o	o	o	o

28	**Do quality education and training materials exist for these topics?**	**Good Materials Exist**	**Some Materials Exist**	**Few Materials Exist**	**Don't Know**
	HDF5	o	o	o	o
	Parallel NetCDF	o	o	o	o
	MPI-IO	o	o	o	o

29 **If you would like to comment on parallel i/o and fault tolerance skills, please do so here:**

Pg 8 ***Scientific Visualization***

30	**How important are these skills?**	**Not Important**	**Somewhat Important**	**Very Important**	**Don't Know**
	Understanding of visualization tools	o	o	o	o
	Ability to use visualization APIs	o	o	o	o
	Understanding the differences between surface and volume rendering	o	o	o	o
	Bottlenecks to visualization	o	o	o	o

31	**Do quality education and training materials exist for these topics?**	**Good Materials Exist**	**Some Materials Exist**	**Few Materials Exist**	**Don't Know**
	Understanding of visualization tools	o	o	o	o
	Ability to use visualization APIs	o	o	o	o
	Understanding the differences between surface and volume rendering	o	o	o	o
	Bottlenecks to visualization	o	o	o	o

32 **Please indicate the visualization tools that are most important to you.**

33 **If you would like to comment on scientific visualization skills, please do so here:**

Pg 9 ***Grid Computing***

34	**How important are these skills?**	**Not Important**	**Somewhat Important**	**Very Important**	**Don't Know**
	Understanding security mechanics	o	o	o	o
	Understanding Grid middleware	o	o	o	o
	Ability to use a grid environment	o	o	o	o

35	**Do quality education and training materials exist for these topics?**	**Good Materials Exist**	**Some Materials Exist**	**Few Materials Exist**	**Don't Know**
	Understanding security mechanics	o	o	o	o
	Understanding Grid middleware	o	o	o	o
	Ability to use a grid environment	o	o	o	o

36 **If you would like to comment on grid computing skills, please do so here:**

Pg 10 ***Education***

37	**How important are these skills?**	**Not Important**	**Somewhat Important**	**Very Important**	**Don't Know**
	Developing teamwork skills in large, distributed, multi-institutional, multi-cultural, global research team environments	o	o	o	o
	Developing mentoring skills to be able to assist others in advancing their studies and research	o	o	o	o
	Developing professional skills in writing top-quality research papers, giving effective presentations, grant writing, etc.	o	o	o	o
	Effective teaching pedagogy	o	o	o	o
	Good research practices (e.g., working on large teams and multi-year projects)	o	o	o	o

38 **If you would like to comment on education skills for students, please do so here:**

Pg 11 ***Critical Skills and Development***

39	**How important are these skills for undergraduates?**	**Not Important**	**Somewhat Important**	**Very Important**	**Don't Know**
	Mathematics requirements	o	o	o	o
	Programming and algorithm basics	o	o	o	o
	Basics of modeling and simulation principles	o	o	o	o

40	**How important are these skills for graduates and professionals?**	**Not Important**	**Somewhat Important**	**Very Important**	**Don't Know**
	Mathematics requirements	o	o	o	o
	Programming and algorithm basics	o	o	o	o
	Basics of modeling and simulation principles	o	o	o	o

41	**What skill sets are most critical?**	**Not Critical**	**Somewhat Critical**	**Very Critical**	**Don't Know**
	Programming	o	o	o	o
	Advanced Programming and Parallel Programming	o	o	o	o
	Numerical Libraries and Algorithms	o	o	o	o
	Data Management	o	o	o	o
	Parallel I/O a Fault Tolerance	o	o	o	o
	Scientific Visualization	o	o	o	o
	Grid Computing	o	o	o	o
	Education	o	o	o	o

42	**What skill sets need the most development?**	**No Development**	**Some Development**	**Much Development**	**Don't Know**
	Programming	o	o	o	o
	Advanced Programming and Parallel Programming	o	o	o	o
	Numerical Libraries and Algorithms	o	o	o	o
	Data Management	o	o	o	o
	Parallel I/O a Fault Tolerance	o	o	o	o
	Scientific Visualization	o	o	o	o
	Grid Computing	o	o	o	o
	Education	o	o	o	o

43 **What is your #1 most important domain-specific application?**

44 **What is your #2 most important domain-specific application?**

45 **What is your #3 most important domain-specific application?**

46 **If you would like to comment on critical skills and development, please do so here:**

Pg 12 Your Background

Please answer a few questions about your background and expertise.

47 **Which of the following describes your expertise in the application of high performance computing (check all that are applicable):**

- o Just beginning to use HPC in my research
- o Regularly use HPC in my research
- o Apply existing codes for my work
- o Add to existing codes for my work
- o Develop new codes
- o Use multiple HPC sites/facilities to accomplish my work

48 **Please provide pointers to excellent training and education materials that you have or know of, that are available to be shared by the community.**

49 **Optional: If you would like to work with us on creating education and training materials on these topics, please either fill in your email address below or send an e-mail to Steve Gordon and Scott Lathrop:**

E-Mail Addresses: Steve Gordon (sgordon@osc.edu) Scott Lathrop (lathrop@mcs.anl.gov)

Glossary

ACML	ACML (AMD Core Math Library) provides a free set of thoroughly optimized and threaded math routines for HPC, scientific, engineering and related compute-intensive applications. These include linear algebra routines, Fast Fourier Transforms, and random number generators. (Optimized for AMD processors.) http://developer.amd.com/cpu/Libraries/acml/Pages/default.aspx
Amira	Amira is a powerful, multifaceted software platform for visualizing, manipulating, and understanding Life Science and bio-medical data. http://www.amiravis.com/
AVS Express	AVS/Express is a comprehensive and versatile data visualization tool offering rapid data analysis and rich visualization techniques combined with an intuitive, graphical application development environment. http://www.avs.com/software/soft_t/avsxps.html
CAF	Co-Array Fortran (CAF) is an SPMD parallel programming model based on a small set of language extensions to Fortran 90/95. CAF supports access to non-local data using a natural extension to Fortran 90 syntax, lightweight and flexible synchronization primitives, pointers and dynamic allocation of shared data, and parallel I/O. http://www.hipersoft.rice.edu/caf/
Chapel	Chapel (The Cascade High-Productivity Language) is a parallel programming language being developed that supports a multithreaded execution model via high-level abstractions for data parallelism, task parallelism, concurrency, and nested parallelism. http://chapel.cs.washington.edu/
Charm++	Charm++ is a machine-independent parallel programming system. Programs written in Charm++ are decomposed into a umber of cooperating message-driven objects call chares. It provides high-level mechanisms and strategies to facilitate the task of developing even highly complex parallel applications. Charm++ programs are written in C++ with a few library calls and an interface description language for publishing Charm++ objects. http://charm.cs.uiuc.edu/research/charm/
CHUD	CHUD (Computer Hardware Understanding Development) is a package of performance and hardware analysis tools from Apple. http://books.google.com/books?id=rYjv9MKZ3JwC&pg=PA421&lpg=PA421&dq=chud&source=bl&ots=oe20uWZhra&sig=WrdMQqZLJFU35o1UDdD8rkxVdfQ&hl=en&ei=xMs3StCzOsbBlAe8rqjjDQ&sa=X&oi=book_result&ct=result&resnum=6#PPA422,M1
Clawpack	CLAWPACK (Conservation LAWs PACKage) is a software package designed to compute numerical solutions to hyperbolic partial differential equations using a wave propagation approach. It is a package of Fortran subroutines for solving time-dependent hyperbolic systems of partial differential equations in 1, 2, and 3 space dimensions, including nonlinear systems of conservation laws. http://kingkong.amath.washington.edu/claw/www/claw50beta.html

cout	Standard C++ output command
CPLEX	CPLEX is a math-programming engine designed to solve mathematical optimization problems. It was originally named for the simplex method and C programming language. It has interior point methods and interfaces for C, C++, C#, Java, Visual Basic and FORTRAN. (There is also a parallel CPLEX for multi-core machines.) http://www.ilog.com/products/cplex/index.cfm
CUDA	CUDA is a general purpose parallel computing architecture that leverages the parallel compute engine in NVIDIA graphics processing units (GPUs) to solve many complex computational problems. It is based on C and includes a set of standard numerical libraries. http://www.nvidia.com/object/cuda_what_is.html
CUDA profiler	The CUDA Visual profiler is a graphical tool that enables profiling C applications running on the GPU. It supports full measurement of memory bandwidth within a kernel, a performance-critical area of NVIDIA CUDA general-purpose parallel computing architecture (see above). http://developer.download.nvidia.com/compute/cuda/CUDA_Visual_Profiler_1.0_mac/CudaVisualProfiler_README_1.0_13June08.txt
dbx	Unix-based line-by-line debugger for C, C++, and Fortran
FFT	The Fast Fourier Transform (FFT) is an algorithm for efficiently computing the discrete Fourier transform and its inverse. Implementations vary by package.
FFTW	FFTW ("Fastest Fourier Transform in the West") is a free C subroutine library for computing the discrete Fourier transform in one or more dimensions, of arbitrary input size, and of both real and complex data. http://www.fftw.org/
Fortress	Fortress is a draft specification of a programming language designed for high-performance computing (HPC) with high programmability. Its syntax and features, such as transactions, specification of locality, and implicit parallel computation, were designed with HPC in mind. http://projectfortress.sun.com/Projects/Community
GLSL	The OpenGL Shading Language (GLSL) is a C-based high level programming language that allows the coding of shading programs for the graphics card. http://www.opengl.org/documentation/glsl/
GSL	The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers. It provides a wide range (over 1,000 functions) of mathematical routines such as random number generators, special functions and least-squares fitting. http://www.gnu.org/software/gsl/
Hardware counters	Hardware performance counters are registers built in to the CPU to track low-level operations or events within the processor with minimal overhead (http://perfsuite.ncsa.uiuc.edu/publications/LJ135/x27.html). They can be accessed by a variety of processor-specific applications. For example, the *hpmcount* command (http://publib.boulder.ibm.com/infocenter/systems/index.jsp?topic=/com.ib

	m.aix.cmds/doc/aixcmds2/hpmcount.htm) (for IBM systems) provides the execution wall clock time, hardware performance counters information, derived hardware metrics, and resource utilization statistics for any given program.
histx	histx is a small set of tools and libraries that can assist with performance analysis and bottleneck identification designed to run on SGI Altix machines.
IDL	IDL is a computing environment for data analysis, visualization, and application development. This cross-platform tool includes image processing, signal processing, and math and statistics routines, as well as a variety of pre-built visualization options. http://www.ittvis.com/ProductServices/IDL.aspx
IMSL	The IMSL Libraries are a set of comprehensive mathematical and statistical functionality for software applications that require numerical analysis. The IMSL Libraries are written in C, C#, Java and Fortran. http://www.vni.com/products/imsl/features.php
LAPACK	LAPACK (Linear Algebra PACKage) is written in Fortran90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. http://www.netlib.org/lapack/
MA28	MA28 consists of four user callable routines for the solution of sparse linear equations where the coefficient matrix may be asymmetric and numerical pivoting may be required for stability. The MA28 package is not in the public domain. It is part of the Harwell Subroutine Library (http://hsl.rl.ac.uk/archive/hslarchive.html), which is a general library of mathematical subroutines written in Fortran. http://www.netlib.org/harwell/readme
MASS	Mathematical Acceleration Subsystem (MASS) consists of libraries of tuned mathematical intrinsic functions; they offer improved performance over the standard mathematical library routines, are thread-safe and support compilations in C, C++, and Fortran applications. (Available for AIX and Linux.) http://www-01.ibm.com/software/awdtools/mass/
MKL	Intel's Math Kernel Library (MKL) is a library of highly optimized, extensively threaded math. Core math functions include BLAS, LAPACK, ScaLAPACK, Sparse Solvers, Fast Fourier Transforms, Vector Math, and more. (Optimized for Intel microprocessors.) http://software.intel.com/en-us/intel-mkl/
MPI	MPI (Message Passing Interface) is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementers, and users. http://www.mcs.anl.gov/research/projects/mpi/ Specific implementations vary (example: http://www.open-mpi.org/).
NAG	The NAG Library has over 1,450 numerical and statistical algorithms and is available in C and Fortran for parallel and Symmetric Multi-Processor (SMP) systems. http://www.nag.com/numeric/numerical_libraries.asp

NCL	NCAR Command Language (NCL) is a free interpreted language designed specifically for scientific data processing and visualization. http://www.ncl.ucar.edu/overview.shtml
OpenGL	OpenGL is the industry's most widely used and supported 2D and 3D graphics API, incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. http://www.opengl.org/about/overview/
OpenMP	OpenMP (OpenMP.org) is an API specification for multi-platform shared-memory parallel programming in C/C++ and Fortran on all architectures, jointly defined by a group of major computer hardware and software vendors. http://openmp.org/wp/about-openmp/
ParaView	ParaView is an open-source, multi-platform data analysis and visualization application. The data exploration can be done interactively in 3D or programmatically using the software's batch processing capabilities. ParaView was developed to analyze extremely large datasets using distributed memory computing resources. http://www.paraview.org/
parMETIS	ParMETIS is an MPI-based parallel library that implements a variety of algorithms for partitioning unstructured graphs, meshes, and for computing fill-reducing orderings of sparse matrices. http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview
PETSc	PETSc is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It employs the MPI standard for parallelism. http://www.mcs.anl.gov/petsc/petsc-2/
PyMOL	PyMOL is a flexible open-source molecular graphics and modeling package that can be used to render 3D images of small molecules and biological macromolecules such as proteins and generate animated sequences. http://www.pymol.org/
ROOT	ROOT is an object-oriented program and library that provides functionality needed to handle and analyze large amounts of data in a very efficient way. It includes visualization capabilities (histograming methods in 1, 2 and 3 dimensions, curve fitting, function evaluation, minimization, graphics and visualization classes) as well as a set of numerical algorithms (for optimization, interpolation, etc.). http://root.cern.ch/drupal/content/about
ScaLAPACK	ScaLAPACK includes a subset of LAPACK routines redesigned for distributed memory MIMD parallel computers. http://www.netlib.org/scalapack/scalapack_home.html
Shark	Shark is the preeminent performance analysis tool in Apple's Xcode development environment. http://books.google.com/books?id=rYjv9MKZ3JwC&pg=PA421&lpg=PA421&dq=shark&source=bl&ots=oe20uWZhra&sig=WrdMQqZLJFU35o1UDdD8rkxVdfQ&hl=en&ei=xMs3StCzOsbBlAe8rqjjDQ&sa=X&oi=book_result&ct=result&resnum=6#PPA422,M1

Titanium	Titanium is an explicitly parallel dialect of Java developed at UC Berkeley to support high-performance scientific computing on large-scale multiprocessors, including massively parallel supercomputers and distributed-memory clusters with one or more processors per node. http://titanium.cs.berkeley.edu/
UPC	Unified Parallel C (UPC) is an extension of the C programming language designed for high performance computing on large-scale parallel machines. The language provides a uniform programming model for both shared and distributed memory hardware. http://upc.lbl.gov/
Vampir	Vampir provides an analysis framework that implements optimized event analysis algorithms and customizable displays to interactively render complex performance monitoring data. http://www.vampir.eu/
VAPOR	VAPOR (Visualization and Analysis Platform for Ocean, Atmosphere, and Solar Researchers) provides a visual data discovery environment tailored towards the specialized needs of the geosciences community, advanced interactive 3D visualization tightly coupled with quantitative data analysis, and other functionality. http://www.vapor.ucar.edu/
VisIt	VisIt is a free interactive parallel visualization and graphical analysis tool for viewing scientific data on Unix and PC platforms. It can be used to visualize scalar and vector fields defined on two- and three-dimensional (2D and 3D) structured and unstructured meshes. Users can generate visualizations from their data, animate them through time, manipulate them, and save the resulting images for presentations. https://wci.llnl.gov/codes/visit/about.html
VMD	VMD (Visual Molecular Dynamics) is a multi-platform molecular visualization program for displaying, animating, and analyzing large biomolecular systems using 3-D graphics and built-in scripting. http://www.ks.uiuc.edu/Research/vmd/
VTK	The Visualization Toolkit (VTK) is an open-source, freely available software system for 3D computer graphics, image processing and visualization. It consists of a C++ class library and several interpreted interface layers including Tcl/Tk, Java, and Python. VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods; and advanced modeling techniques such as implicit modeling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation. http://www.vtk.org/
X10	X10 is a programming language being developed that is specifically designed for parallel programming. It focuses on a hardware-software co-design methodology to integrate advances in chip technology, architecture, operating systems, compilers, programming language and programming tools to deliver improvement in development productivity for parallel applications. http://domino.research.ibm.com/comm/research_projects.nsf/pages/x10.index.html