Computational Science Education Focused on Future Domain Scientists

Lucas A. Wilson
Texas Advanced Computing Center
Department of Statistics and Data Sciences
The University of Texas at Austin
Austin, Texas, U.S.A.
Email: lucaswilson@acm.org

S. Charlie Dey
Texas Advanced Computing Center
The University of Texas at Austin
Austin, Texas, U.S.A.
Email: charlie@tacc.utexas.edu

***Abstract*—The majority of university courses which educate students in high performance, parallel, and distributed computing are located within computer science departments. This can potentially be a hurdle to students from other disciplines who need to acquire these critical skills.**

We discuss a sequence of application-driven courses designed to educate undergraduate and graduate students who do not necessarily have a computer science background on developing scientific research software, with an emphasis on using high performance, parallel, and distributed computational systems.

I. Introduction

Many university courses focused on educating students on high performance, parallel and advanced computing are rightly located within computer science/engineering departments. These courses primarily focus on the dissemination of theoretical concepts, algorithmic design, and more abstract concepts such as asymptotic complexity.

However, computation has become a critical component of the scientific process for nearly all disciplines, regardless of whether they have traditionally held a significant interest in computational simulation and experimentation. The age of Big Data has forced disciplines from economics, to music, to anthropology to embrace computation in order to test, analyze, and curate the enormous troves of information collected or the enormous complexity of the research questions to be asked.

Unfortunately, many students outside of computer science/engineering departments do not take courses in high performance, parallel, and advanced computation when they are hosted within the computer science department. The reasons for this are varied and debatable, but the reality remains that students who need to learn computational techniques are not exposed to the courses which could potentially provide them with these tools and concepts.

In this paper we discuss a sequence of courses that we have developed at the Texas Advanced Computing Center (TACC) and teach at the University of Texas at Austin which focus on providing students in varied majors the skills necessary to participate in computationally-driven research and development in both academia and industry. These courses are application-driven, meaning that they tend to focus more heavily on hands-on exercises, labs, and projects than traditional courses. In large part, the content of these courses has been informed by the needs to students and research staff who make use of TACC's computational, analysis, and visualization resources, while being sufficiently general as to apply these skills to other systems and environments.

The rest of this paper is organized as follows: Section II provides a longer description of the choices we have made when structuring this sequence of courses and the reasons why those choices were made, section III provides descriptions of the courses in the sequence, including topics covered and semester project structure. Section IV breaks down the student demographics of our courses, and highlights the varied nature of the students we are targeting. Finally, section V highlights some of the lessons we have learned through teaching these courses and potential changes to our course materials in order to address those lessons.

II. Rationale

TACC is committed to lifelong learning in advanced computation through programs targeted at audiences from elementary school students through tenured faculty and research scientists. Figure 1 shows the progression of TACC's learning programs in a current or potential future computational researcher's path:

- **K-12 Outreach** programs target elementary, middle, and high school students and attempt to drive interest in Science, Technology, Engineering, and Mathematics (STEM) learning through interesting and novel uses of computation ranging from robotics to smart phone app development. In the summer of 2016 TACC hosted 25 high school students for 4 weeks during the summer as part of a Verizon Foundation Innovative Learning Summer School Experience [1], focusing on methods for using technology to develop novel business or philanthropic ideas;
- **Undergraduate Programs** are focused primarily on undergraduate students as a way of providing exposure to potential careers in STEM fields. Programs in this area include NSF's Research Experiences for Undergraduates (REU) program [2];

EduHPC2016; Salt Lake City, Utah, USA; November 2016
978-1-5090-3827-5/16/$31.00

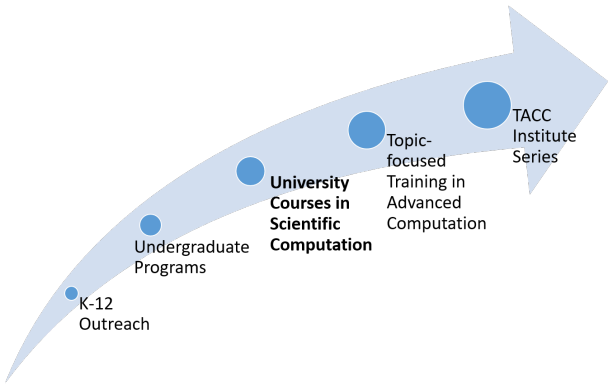

Fig. 1. Lifelong Learning in Advanced Computation

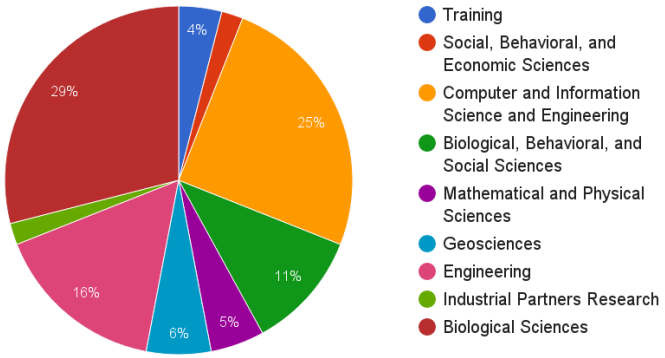

Fig. 2. Breakdown of Projects on TACC Systems by Field of Science

- **University Courses** – which are the primary focus of this paper – are targeted at upper-division undergraduate and graduate students in all majors who would like to learn tools, techniques, methods, and best practices for incorporating advanced computation into their scientific research endeavors;
- **Topic-focused Training** programs are held monthly, both in-person and broadcast via the web, and focus on specific tools or practices for high performance computing, large-scale data analysis, scientific data visualization, or computational life sciences;
- **TACC Institutes** are 5-day training programs focused on a particular subject area, where attendees are presented with sufficient breadth and depth of information as to be able to begin incorporating advanced computation into their academic or industrial research.

With respect to university courses, TACC has been working in conjunction with departments and divisions within the College of Natural Sciences at the University of Texas at Austin for nearly a decade to offer semester-long courses which impart foundational HPC skills. While only a single course initially existed which focused on parallel computing using OpenMP and MPI, additional courses have been developed which have increased the breadth of education in advanced scientific computation, as well as backfill the necessary prerequisite skills to be successful in developing parallel numerical applications.

Today, TACC researchers work in conjunction with UT's department of Statistics and Data Sciences (SDS) to offer five courses in scientific computation, which are targeted directly at students with a need to develop applied skills, and who do not necessarily receive instruction in theoretical computer science. Instead these courses target students in engineering, natural sciences, social sciences, and liberal arts, and seek to impart critical computational skills – including sequential/parallel application development, large-scale data analysis, and visualization – to students whose disciplines are being rapidly and radically transformed by the incorporation of advanced computation into the research pipeline.

Figure 2 gives a breakdown – as of the time of this writing – of the fields of science represented on TACC systems by project. While research projects in computational and information science make up a sizeable fraction of TACC's computational projects (25%), the number of projects in other domains where students do not already receive computational knowledge as part of the standard curriculum is significantly larger.

As more scientific domains embrace computation to improve the pace and quality of discovery, students in these disciplines will need a comprehensive education in the fundamentals of scientific computation, parallel programming, data analysis, and visualization.

While some courses which cover similar general concepts exist within UT Austin's computer science department through the Elements classes for non-CS majors [3], these courses are more focused on broader concepts (e.g., CS323E-Elements of Scientific Computing is algorithms-focused and based on Matlab) rather than on specific tools and skills for using large cyberinfrastructure. There is no computer science course for parallel computing that is intended for non-CS majors.

III. Available Courses

TACC and SDS currently offer four courses in scientific computation which are targeted at undergraduate and graduate students from all majors. They can be taken individually as electives, depending on the particular needs of the student. Alternatively, courses can be taken as a sequence leading to either a certificate in scientific computation (undergraduate), or a portfolio in scientific computation (similar to a certificate but for Master's students).

Courses are dual listed for undergraduate and graduate students, and are cross-listed with course number in other departments which have determined that the skills imparted in these courses is critical to the long-term success of their students.

All courses have Integral Calculus (UT-Austin course M408D [4]) as a prerequisite. Although we present these courses as a sequence, they do not serve as prerequisites to one another. Instead, a student may begin with whichever course they deem best fits their needs and existing skill set.

Materials and lecture notes for several of our courses have been curated and placed in the public domain through a donation from Chevron Corporation [5] and are available at https://www.tacc.utexas.edu/education/academic-courses/.

A. Introduction to Scientific Programming

The *Introduction to Scientific Programming (ISP)* course is focused on teaching basic logical and computational principles using C and Fortran. The concepts focused on in ISP include:

- Basic BASH and command-line usage,
- Compilation of C and Fortran and use of optimization flags,
- Variable declaration and assignment,
- Intrinsic data types in both C and Fortran,
- Boolean logic and comparison operators,
- Conditional statements (if/else, switch/case),
- Fixed-iteration and conditional loops (pre- and post-test),
- User-defined compound types,
- Single- and multi-dimensional arrays,
- Pointers
- Basic file I/O in fixed and free form, and
- Linking external libraries.

The Choice of C and Fortran: The choice of C and Fortran as teaching languages is the result of several considerations. First, C and Fortran are the only languages for which official MPI bindings exist, according the the MPI standard [6]. Similarly, C/C++ and Fortran are the only languages which are officially supported by OpenMP standard [7]. As MPI and OpenMP are the standards used for teaching parallel computation in our *Parallel Computing for Scientists and Engineers (PCSE)* course (see Section III-C), they are the natural choices for this first course in the track. Additionally, C and Fortran are the dominant compiled languages on many HPC systems [8], and many students go on to research assistant positions where modification of existing C and/or Fortran code will be a major component of their work. Having an existing knowledge of these fundamental HPC languages can be important to students' continued success in computational sciences research.

Semester Project: ISP requires students to complete an end of term programming project which combines all of the above elements into a single application. Students are either assigned or may choose for a small set of domain science-targeted options, typically including:

- a numerical application for solving partial differential equations (PDEs) using an explicit finite-difference method,
- a genomics-inspired single sequence alignment option, and
- a computer science and machine learning inspired application in graph theory

B. Scientific and Technical Computing

Scientific and Technical Computing (STC) focuses on teaching basic principles, tools, and techniques directly related to scientific and numerical computation. Taught simultaneously in C and Fortran, STC covers:

- Interoperability between C and Fortran,
- Boolean comparison of floating point numbers,
- Error accumulation in floating point numbers,
- Use of mathematical libraries, such as BLAS,
- Developing sequential applications to solve numerical problems,
- Common performance optimization techniques, and
- Use of make, git, and other tools common development tools,
- Some common nonlinear algorithms,
- Applied topics from linear algebra,
- Monte Carlo Method,
- Fast Fourier Transforms.

Students in STC are expected be comfortable in both C and Fortran, and assignments are typically expected to be completed in either or both languages (depending on the instructor). Upon successful completion of the course, students should be able to effectively develop physically realistic computational applications by using both forward and inverse methods, manage multiple source files using git and make, and make basic performance optimization decisions (e.g., loop ordering when traversing multi-dimensional arrays).

Semester Project: STC requires students to complete a sizable programming project which incorporates all of the topics and concepts learned throughout the course. Students in STC are given the option to propose their own semester project, so long as it covers the topics listed above. Some examples of past successful projects proposed by students include:

- Using LAPACK to solve simple partial differential equations by the implicit method,
- Developing agent-based traffic simulations for urban planning,
- Building simple clustering tools to study species phylogeny,
- Implementation of Greedy Dual Coordinate Descent in OpenMP,
- Multi linear regression and regularized optimization,
- Simulating damaged atomic lattices using random work method,
- Transient Navier Stokes Finite Element Solver, and
- Aquifer Modeling, Superposition Analysis.

C. Parallel Computing for Scientists and Engineers

The *Parallel Computing for Scientists and Engineers (PCSE)* course focuses on parallelization of scientific and numerical problems – first introduced in STC – using OpenMP and MPI. The course is also designed to change the student paradigm of programming, from sequential to parallel. Topics discussed are:

- Parallel computing concepts,
- Parallel computing architectures and hardware,
- Standard programming models for parallel computers,

- Large-scale system environments for development and execution of parallel applications,
- Understanding, recognizing, and adjusting for data races,
- Shared memory parallel programming with OpenMP,
- OpenMP directives,
- Parallelizing serial code with OpenMP,
- Synchronization and control,
- Performance and scalability of OpenMP,
- Distributed memory programming in MPI,
- MPI communication operations and data structures,
- Parallelizing serial code with MPI,
- Performance analysis of MPI, and
- Parallel algorithm design, optimization, and analysis.

Semester Project: PCSE semester projects are proposed by the students, who can propose either one-person or two-person projects. Projects must use either MPI or OpenMP (or both), and provide a short report which details the project and shows any achieved parallel speedup. Some interesting semester projects in the past have included:

- PageRank [9] calculation using OpenMP,
- Hybrid MPI/OpenMP solver for Schrödinger's equations,
- Adaptive Mesh Refinement using OpenMP,
- Stochastic Gradient Descent using Hybrid Computing,
- Sliding Windows in Genome Sequencing, using OpenMP and MPI,
- Support Vector Machine through MPI, and
- Short-time Fast Fourier Transforms with Hybrid (MPI/ OpenMP) Computing.

D. Visualization and Data Analysis for Scientists and Engineers

The *Visualization and Data Analysis for Scientists and Engineers (VDASE)* is targeted at analysis of data by visual or statistical means. The data to be analyzed may be the output of computational simulations or data collected from various sources. Specifically, VDASE teaches students to analyze data using languages like Python and R, data-oriented tools and libraries like Hadoop/Spark, and visualization platforms such as VisIt and Paraview.

Semester Project: Like PCSE projects (see section III-C), VDASE projects can be done by one- or two-student teams, and must include a report detailing the project and displaying the resulting analysis or visualizations. Some interesting past project have included:

- Visualization of WRF [10] hurricane forecast datasets using ParaView [11],
- Analysis of geographic locations of hashtags in Twitter data,
- Analysis of large-scale ensemble results from genome sequencing, and
- Visualization of fluid flow through non-linear, flexible tubing.

E. Distributed and Grid Computing for Scientists and Engineers

The *Distributed and Grid Computing for Scientists and Engineers (DGCSE)* course has been offered infrequently compared to our other courses, and has not been offered in the last few years. The course is focused on exposing students to tools and concepts such as:

- Globus [12],
- Condor [13],
- TACC-developed HTC tools such as Launcher [14] and MyCluster [15],
- Job arrays in SGE [16], SLURM [17], and other resource managers,
- BOINC [18], and
- Managing files in distributed environments.

Semester Project: Students in DGCSE use these tools and concepts to build semester projects which consist of executing large ensembles of sequential jobs, organizing the resulting output data, and performing some basic statistical analysis of the ensemble results. Some examples include:

- Scoring docking affinity on a large number (¿10,000) of proteins/ligands using Autodock Vina [19],
- Text analysis of many separate short stories, and
- Rendering of animation frames.

IV. Demographics

Each semester, we collect basic background and demographic information from students through anonymous, optional surveys offered during the second week of class (in order to allow for most of the add/drop variability to subside). Specifically, we are interested in learning about students:

- Graduate or undergraduate status,
- Major or degree plan,
- Previous experience with computers, and
- Gender.

We sent out the survey to 92 of our currently enrolled students and asked them to anonymously answer 5 questions regarding their university status with the understanding that the survey was completely voluntary.

- Are you an undergraduate or graduate student?
- What college are you enrolled in?
- What is your current major?
- What is your gender?
- Have you ever written a computer program with over 500 lines of code?

We informed the students that the demographic information collected about the makeup of our course would be used for research and publication purposes. We had 57 students reply to the survey and our data is based on those responses.

A. Enrollment by Graduate or Undergraduate Status

The TACC/SDS courses were designed with the future domain scientist in mind, and such is offered to both graduate and undergraduate students. Many of the natural science and engineering graduate students are strongly encouraged to

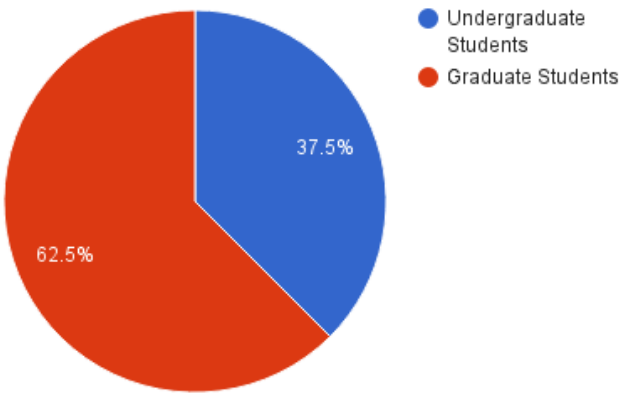

Fig. 3. Breakdown in Enrollment by Graduate or Undergraduate Status

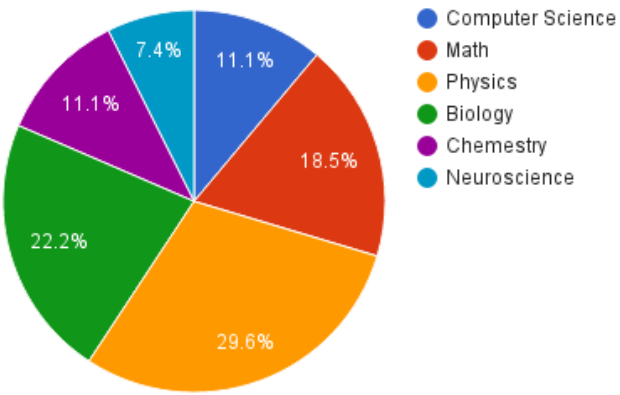

Fig. 5. Breakdown in Enrollment from College of Natural Sciences

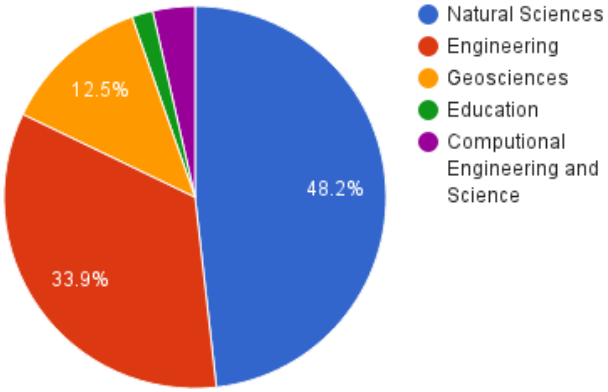

Fig. 4. Breakdown in Enrollment by College

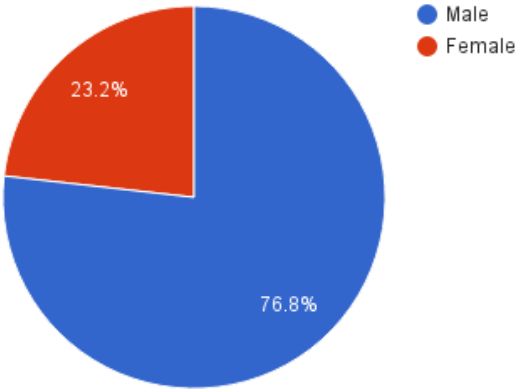

Fig. 6. Breakdown in Enrollment by Gender

enroll in the TACC/SDS program to get a better understanding of the technologies and programming techniques they will be applying in their given domain science. Hence there are slightly more graduate students enrolled than undergraduates. Figure 3 details the breakdown between enrolled undergraduates students and graduate students.

B. Enrollment by College

Information on department association is collected from both graduate and undergraduate students, and is aggregated based on the association of departments to colleges at the University of Texas at Austin. It is worth noting that computer science falls under natural sciences at UT Austin, rather than engineering (as would be the case at many other institutions). Figure 4 shows the breakdown of students by college enrolled. Nearly half (48.2%) of all of the students in our courses are from the college of natural sciences (CNS), which includes math, physics, astronomy, chemistry, biology, neuroscience, and computer science. The category *Computational Engineering and Science* is the interdisciplinary Institute for Computational Engineering and Sciences (ICES) at UT Austin, which is composed of faculty and students from both natural sciences and engineering disciplines. If we break down the enrollment from the college of natural science into its independent departments, we see that only 11.1% of students from the college of natural science are seeking degrees in the computer science department (Figure 5. This translates to only 5.3% of total enrollment in the scientific computing sequence of courses are students seeking computer science degrees (BS, MS, PhD). Another interested note is that all of the students listed as *Neuroscience* (7.4% of CNS, 3.6% total) are pre-medical students.

C. Enrollment by Gender

When looking at student genders, we find that nearly a quarter (23.2%) of all students enrolled are female. This is roughly in line with diversity statistics from several advanced computing centers, including TACC (27% female employees) [20] and the National Center for Supercomputer Applications (NCSA) in Illinois (25% female employees) [21].

V. Lessons Learned

A. Embracing Emerging Computational Disciplines

As Figure 2 illustrated, a significant fraction of projects on TACC systems are categorized under scientific domains who are just beginning the journey to computational dominance. Unlike physics, chemistry, and engineering – which were near exclusive users of large computational systems in the past –

these emerging computational disciplines, which range from biological sciences to social sciences, are in desperate need of education in computational and data-enabled techniques. Moving forward, we plan to transition our application examples towards these new disciplines, while still providing existing examples from physics, chemistry, and engineering.

B. Meeting the Needs of Data-enabled Science and Engineering

Emerging computational disciplines continue to transition to computational and data-enabled experimentation, and the needs of these emerging disciplines is significantly different from the needs of students entering more established computational science domains. Specifically, lack of *legacy* toolsets and applications, as well as less algorithmic reliance on OpenMP and MPI, make knowledge of languages such as C and Fortran less critical. Instead, many students in these emerging disciplines need knowledge of Python, R, and Hadoop/Spark in order to be effective data-enabled domain scientists.

C. Shift from Graduate to Undergraduate Participation

While our HPC course offerings were initially focused exclusively on graduate students, continued interest from upper-division undergraduate students convinced us that we should dual-list our courses for both graduate and undergraduate students.

In the last five years, early track courses (ISP, STC) have seen a shift from majority graduate to parity graduate and undergraduate participation, while late track courses (PCSE, VDASE) still maintain majority graduate participation. However, participation by undergraduate students in late track courses continues to increase, and we expect that computational and data-enabled science and engineering education will continue shifting toward the undergraduate level.

VI. Conclusion

TACC's courses in scientific computation – while now a decade old – continue to evolve in order to address current languages, tools, and techniques in use by domain scientists in order to perform their computational experiments. These courses are very targeted at application, and tend to eschew theoretical concepts in favor of having students develop scientific applications and learn through extensive hands-on exercises.

Students and student research advisers have been extremely pleased with this approach, and the demand for our courses has increased significantly in the last several years. Currently demand for our courses exceeds facility capacity every semester. And while gender diversity in our courses is on par with gender diversity at major advanced computing centers around the country, we continue to strive for more gender parity every semester.

Going forward, we will focus on continuing to retool our application-driven courses to focus on the tools and techniques most needed by students in domain sciences. Python and R will become primary languages in data analysis and distributed computing courses, and tools for data reduction and analysis, machine learning, and collaborative development/analysis will become important components of our courses – where appropriate – going forward.

References

[1] M. Easter, "High Schoolers Head to College for Immersive Entrepreneurship and STEM Camp," June 2016.

[2] R. Gomez, "Integrative Computational Engineering and Research Traineeship," https://www.tacc.utexas.edu/icert-reu, 2014),.

[3] Department of Computer Science, University of Texas at Austin, "CS Classes for Non-Majors," http://www.cs.utexas.edu/undergraduate-program/academics/curriculum/courses#non-majors, 2016.

[4] Department of Mathematics, University of Texas at Austin, "M408D Syllabus," https://www.ma.utexas.edu/academics/courses/syllabi/M408D.php, 2016.

[5] McCombs Today, "McCombs, TACC Receive Gifts from Chevron to Educate the Next Generation of Scientists," 2009.

[6] M. P. I. Forum, "MPI: A Message Passing Interface Standard," https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf, 2015.

[7] OpenMP Architecture Review Board, "OpenMP Specifications," http://openmp.org/wp/openmp-specifications/, 2015.

[8] R. Budiardja, M. Fahey, R. McLay, P. M. Don, B. Hadri, and D. James, "Community use of xalt in its first year in production," in *Proceedings of the Second International Workshop on HPC User Support Tools*, ser. HUST '15. New York, NY, USA: ACM, 2015, pp. 4:1–4:10. [Online]. Available: http://doi.acm.org/10.1145/2834996.2835000

[9] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: bringing order to the web." 1999.

[10] W. C. Skamarock, J. B. Klemp, J. Dudhia, D. O. Gill, D. M. Barker, W. Wang, and J. G. Powers, "A description of the advanced research wrf version 2," DTIC Document, Tech. Rep., 2005.

[11] A. Henderson, J. Ahrens, C. Law *et al.*, *The ParaView Guide*. Kitware Clifton Park, NY, 2004.

[12] I. Foster and C. Kesselman, "Globus: a metacomputing infrastructure toolkit," *International Journal of High Performance Computing Applications*, vol. 11, no. 2, pp. 115–128, 1997. [Online]. Available: http://hpc.sagepub.com/content/11/2/115.abstract

[13] R. Raman, M. Livny, and M. Solomon, "Matchmaking: distributed resource management for high throughput computing," in *High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on*, Jul 1998, pp. 140–146.

[14] L. A. Wilson and J. M. Fonner, "Launcher: A shell-based framework for rapid development of parallel parametric studies," in *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, ser. XSEDE '14. New York, NY, USA: ACM, 2014, pp. 40:1–40:8. [Online]. Available: http://doi.acm.org/10.1145/2616498.2616534

[15] E. Roberts, M. Dahan, E. Walker, and J. Boisseau, "Mycluster ensemble manager: Ensemble jobs in the teragrid user portal," *Proc. of TeraGrid*, vol. 7, 2007.

[16] W. Gentzsch, "Sun grid engine: towards creating a compute power grid," in *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*, 2001, pp. 35–36.

[17] A. B. Yoo, M. A. Jette, and M. Grondona, *SLURM: Simple Linux Utility for Resource Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60. [Online]. Available: http://dx.doi.org/10.1007/10968987_3

[18] D. P. Anderson, "Boinc: a system for public-resource computing and storage," in *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, Nov 2004, pp. 4–10.

[19] O. Trott and A. J. Olson, "Autodock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading," *Journal of Computational Chemistry*, vol. 31, no. 2, pp. 455–461, 2010. [Online]. Available: http://dx.doi.org/10.1002/jcc.21334

[20] Texas Advanced Computing Center, "Diversity," https://www.tacc.utexas.edu/about/diversity, 2016.

[21] National Center for Supercomputer Applications, "NCSA diversity by the numbers," http://www.ncsa.illinois.edu/diversity, 2016.