Education and Research Challenges in Parallel Computing

L. Ridgway Scott[1], Terry Clark[2], and Babak Bagheri[3]

[1] The Institute for Biophysical Dynamics, the Computation Institute, and the Departments of Computer Science and Mathematics, The University of Chicago, Chicago IL 60637, USA
[2] Department of Electrical Engineering and Computer Science, and Information & Telecommunication Technology Center, The University of Kansas, Lawrence, KS 66045, USA
[3] PROS Revenue Management, 3100 Main Street, Houston, TX 77002, USA

Abstract. Over three decades of parallel computing, new computational requirements and systems have steadily evolved, yet parallel software remains notably more difficult relative to its sequential counterpart, especially for fine-grained parallel applications. We discuss the role of education to address challenges posed by applications such as informatics, scientific modeling, enterprise processing, and numerical computation. We outline new curricula both in computational science and in computer science. There appear to be new directions in which graduate education in parallel computing could be directed toward fulfilling needs in science and industry.

1 Introduction

High-performance computing today essentially means parallel computing. Vector processors have a significant role to play, but even these are often grouped to form a parallel processor with vector nodes. Parallel computing has matured both as a research field and a commercial field. A look at the list of top 500 supercomputers[1] shows that there are dozens with thousands of processors. Almost all of the machines on this list (November, 2004) have over one hundred processors. These machines represent only the tip of the iceberg of parallel computers, but the size of the tip gives a hint of what lies below the surface.

The most common type of parallel computer on university campuses is a cluster of (often low cost) workstations. Many of these workstations are themselves parallel computers, with multiple processors on a single board using shared memory. At the moment, dual processor machines are the most common, but this trend may lead to larger numbers of processors available at commodity prices in a single box. Network speeds have increased (and the cost of network interface

[1] `www.top500.org`

cards has decreased) to the point at which a conventional network of computers in a single department or larger organizational unit can be used easily and effectively as a parallel computer for some applications. In addition, smaller clusters using a few tens of computers in a single rack with dedicated networking hardware have become the norm as computational servers for single research groups.

The use of multiple computers working on unified tasks is only increasing. Grids [1, 7] of computers extend parallel computers to a global footprint. Indeed, the grid consisting of all the computers on the planet is an available resource that is being tapped by some. The model originally pioneered by the Search for Extra-Terrestrial Intelligence (SETI), has spawned the `X@home` computational paradigm, where `X` stands for SETI (`setiathome.ssl.berkeley.edu`), folding (`folding.stanford.edu`), fightAIDS (`www.fightaidsathome.org`), predictor (`predictor.scripps.edu`), etc.

The major applications of parallel computing have broadened from the traditional area of scientific, numerically-intensive simulation. Major web servers use parallel computing to answer search queries, and computational biology servers use parallel computing to compare biological sequences. Data-intensive computation has become a major target for parallel computation. Some applications such as biological sequence analysis involve both data-intensive and computation-intensive paradigms. Even the problem of web-page ranking requires the numeric-intensive task of eigenvalue computation on a massive scale. We discuss these topics more at length in section 3.

While the applications of parallel computing continue to broaden, the core challenges of parallel computing have remained substantial and stable for at least a decade. At one point, it was hoped that compilers would solve the problem of parallel programming by automatically converting sequential programs into efficient parallel codes. However, there appears to be no magic wand in sight that will make parallel computing challenges disappear; parallel computing will remain a discipline requiring substantial study for some time.

These developments imply that education in parallel computing has an increasingly important role to play. In the past, fairly simple approaches have been appropriate. But the field has reached a level where diverse and novel ideas are needed. We provide in section 4 some suggestions for new directions to take, with an emphasis on graduate education or advanced under-graduate instruction. In a recent book [10], we have attempted to support curricula of study for parallel computing. This book could be useful in pursuing some of these ideas, but in other cases additional material would be necessary.

Parallel computing has facets which make it of interest to diverse audiences. Anyone hoping to take advantage of high-end computing today must understand parallel computing, so advanced students in any technical field requiring extensive computing will be interested to learn how to harness parallel computation for their own application areas. But parallel computing can also play a role in computer science since it involves various systems issues that can complement traditional approaches. We describe some ideas for courses that might be developed in this direction in section 4.2. Finally, parallel computing can be used simply to challenge students in new ways of thinking. Parallel computing introduces some novel mathematics as well, so it can be used to develop logical reasoning skills. We describe some mathematics that arises in parallel computing in section 4.1.

Graduate education must challenge students with open problems in order to be effective. The field is continuing to be stimulated by new problems from the scientific, engineering and commercial sectors. Graduate students must be engaged in important research questions that can help them reach the forefront of the subject. Graduate education and academic research interact synergistically when a field is still developing. Since parallel computing has been actively studied for several decades, it is reasonable to ask whether there are major research challenges left. In section 5, we describe research questions that can be posed in a beginning graduate class on parallel computing.

2 Curricular Level

There are several levels of instruction that are important. Basic instruction in computing is moving from the undergraduate to the high school curriculum, and there are efforts to move it to the middle school level [5]. Parallelism is an important concept to introduce as early as possible, but we will not address the difficult question of how early one can do this. Instead, we focus on where parallel computing might fit in new ways into existing curricula at the undergraduate and graduate levels.

Many BS and MS students will go from university to a programming job. To what extent then is it appropriate for the university to train in the craft of parallel programming? This depends on two factors. One is the market for programmers and the other is purely pedagogical. We have indicated that parallel computing has become pervasive, so the market impetus seems sufficient to justify courses on parallel computing. On the pedagogical front, one seeks courses that cause students to grow in useful ways, independent of the subject matter. Again, parallelism challenges students to confront difficult issues with both quantifiable goals and simple (if they get it right) solutions.

Minimalist treatments of parallel computing are appropriate in many cases. This type of course only requires (1) an introduction to a basic parallel programming language environment, e.g., the sequential language C and the Message Passing Interface (MPI) library for data exchange, (2) some simple parallel algorithms, and (3) an overview of parallel computer architecture. This approach is sufficient for many applications where the bulk of computation is trivially parallel. Texts are available to support a “cook book” course using popular standards. However, such an approach is not sufficient for a proper graduate course on the subject. It lacks both the intellectual depth required as well as sufficient sophistication to allow students to master difficult issues.

In moving beyond the basics, there are different directions one can take. Emphasis can be placed on algorithms, architecture, programming languages and compilers, software engineering, and so forth. Excellent texts are available to support all of these, and faculty and students can combine different texts to achieve any desired balance. In addition, one can take an integrative approach [10] which highlights key issues in all of these areas.

3 Parallel Computing Paradigms

Diverse parallel computing paradigms make the pedagogical landscape more interesting and challenging. Much of the original impetus for parallel computing came from numeric-intensive simulation. The main issues required for success in this domain center on algorithms, programming languages and compilers, and computer architecture, especially the data-exchange network. Typically, low-latency and high bandwidth are both needed to be successful. However, data-intensive computing makes different demands.

Data-intensive computing refers to a paradigm where a large data set is a significant part of the computation. Interesting applications typically involve data sets so large that they have to be distributed across multiple processors to keep data in primary memory. Here parallelism is used to increase the memory system as much as the computational system. Software systems to support this are essential infrastructure. Often, demands on the communication system of a parallel machine are less critical than for numeric simulation. Data intensive computing may involve interaction with databases with numerous opportunities for parallelism [8]. Grid computing allows the databases in data-intensive computing to be distributed globally [4].

Data-intensive computation has been common in parts of geophysics for several decades. Companies doing data-intensive computation for oil exploration have been major consumers of parallel computers from the beginning. However, data-intensive computing is also on the rise with numerous new applications. Web servers are a simple example of data-intensive computation. However, these servers can also involve a substantial amount of numeric computation. For example, ranking the interactions of web pages, the key to a good search strategy, requires solution of an eigenvalue problem [10]. A linear system whose dimension is the number of web pages in the world (several billions are now being served) requires a good parallel solution. Biological and other sequence comparison algorithms require access to large databases as well as the usual parallel programming support required for numeric parallel computing.

Data-intensive computing can be purely integer-based, but we have shown that many applications involve a mixture of the data-intensive and the numeric-intensive paradigm. One might coin the term “data and numeric intensive computing” for this type of application. Our geophysics example falls in this class, as well as our other examples of web page ranking and biological sequence analysis.

4 New Curricula in Parallel Computing

Parallel computing introduces special challenges relative to sequential counterparts. Numerous traditional and modern areas of computer science and computational science take on new forms when parallel computing is injected as a central issue. One is rooted in parallel programming which provides a special challenge in all of these areas.

In a parallel context, topics in core computer science, including operating systems, compilers, languages, architecture, algorithms, and complexity theory, acquire characteristics unique relative to their serial form. Numerical mathematics changes focus as well when parallel algorithms become an issue; some of the issues are traditional (such as stability of new parallel variants of standard algorithms) and others are novel, since new algorithms become of interest.

Curricula can combine and focus areas to create courses in, for example, scientific and numerical algorithms, computer and network architecture, compilers and runtime systems, advanced courses in theory, and enterprise web services.

To illustrate how new curricula can be developed based on parallel computing, we give details regarding two extremes. One is mathematical and the other is about systems. One can imagine many other alternatives as well.

4.1 Math in Parallel Computing

Parallel computing introduces novel mathematical issues. These can be emphasized in a parallel computing course to challenge students in a mathematical direction. For example, a key issue in automatic parallelizing compilers is dependence analysis. This introduces some simple, but novel, issues requiring set theoretic arguments, the use of multi-index notation, and some simple number theory and algebraic geometry. It is even possible to find an application of Fermat's Last Theorem in dependence analysis [10].

The need to develop new parallel algorithms introduces mathematical challenges. However, it also offers pedagogical opportunities to introduce some advanced concepts in a simple context. For example, we develop in [10] the multigrid algorithm in one dimension. In a sequential world, this would not make any sense, since direct solution methods (e.g., Gaussian elimination) are optimal-order for one dimensional differential equation problems, and much easier to program.

Parallelizing direct methods is itself an excellent mathematical topic. It is fascinating how many different algorithms are available to handle the most important problem of sparse triangular system solution (among all the tasks in numerical linear algebra, this has the *least* inherent parallelism). Moreover, there seems to be a need for more than one algorithm to handle the full range of possible sparsity patterns [10]. Given the intrinsic difficulty of parallel direct methods, it then seems interesting to look at multi-grid as part of a parallel solution strategy. Fortunately, in one-dimension, the data structures required for multi-grid are greatly simplified and suitable for students who have not seen anything about partial differential equations.

Floating-point computation requires an understanding of stability, and establishing this for new parallel algorithms can be a source of good mathematical problems. At the introductory level, parallelization can introduce some fairly simple variants of standard algorithms, and this can lead to some useful exercises to confirm knowledge acquired in basic numerical analysis courses. At a more advanced level, there is the opportunity to be much more challenging, providing advanced numerical issues, and even research problems.

4.2 Parallel Computing as a Systems Subject

In the past, the systems curriculum in computer science was quite simple: compilers, operating systems, and databases. But now these subjects are relatively mature, and new areas are important. A key requirement of a systems course is that there be a system large and complex enough to be challenging to design and build. Parallel computing offers exactly this type of challenge.

Similarities between parallel computing and operating systems could be exploited. A curriculum could extend the classic elements typically covered throughly in operating systems (semaphores, shared-memory segments, monitors, and so on), but which are relevant also in parallel computing and programming languages. Parallel runtime systems rate more attention, and are critical to how a parallel language performs. A systems topic like this often gets limited treatment in an applications oriented course on parallel computing.

Computer science as a discipline is realizing the importance of making connections with other fields in addition to continuing to develop core areas. One way to educate students to be better prepared to do this is to present core material in a broader context. What we are suggesting here is that parallel computing provides just such an opportunity. One can imagine other areas where this could be done: a bioinformatics course that introduces key algorithms and basic learning theory, a data-mining course that covers basics in learning theory and databases, and so on.

5 Research Challenges

It is not easy to pose research questions in a beginning graduate class. However, graduate classes should provide directions that lead to research areas. In some cases, open research questions can be presented as we now illustrate.

5.1 A Long-Standing Challenge

IBM's Blue Gene project has stimulated new interest in parallel molecular dynamics [11]. Solving non-linear ordinary differential equations, such as is done in simulating planetary motion or molecular dynamics, presents a challenge to parallel computation. It has always been hard to simulate molecular dynamics on time scales that are biologically significant. Although it has been demonstrated that parallelism (using a spatial decomposition) can allow very large problems to be solved efficiently [12], the efficient use of parallelism to extend the time of simulation for a fixed-size problem has been elusive. Decomposition of the time domain provides one possible option [2, 10].

5.2 Latency Tolerant Algorithms

The rate limiting factor in many parallel computer systems is the latency of communication. Improvements in latency have been much slower than improvements in bandwidth and computational power. Thus algorithms which are more latency tolerant are of significant interest, cf. [9].

6 Industrial Challenges

Although parallel computing has become the norm in technical computing in laboratories and academe, fine-grained parallelism is not yet widely used in commercial and corporate applications. There are many cases where fine-grained parallelism would be of great benefit. For example, pricing and revenue optimization applications process large numbers of transactions per day to generate forecasts and optimize inventory controls and prices. These computations currently need to be done in a nightly time window. Future applications would benefit from on-line analysis of trends. Whether batch or real-time, the calculations require storage of results (including intermediate results) to relational databases.

Currently, developers in industry have to use tools that are not well matched to this kind of processing. A critical requirement is standardization and broad adoption of tools. Many software developers cannot dictate the kind of hardware or operating systems that customers use. Thus a common choice for inter-process middle-ware is CORBA, which is not designed for high performance. Low-level multi-threading is often done using Java threads because of the universal acceptance of Java.

Even though MPI is a well accepted standard for technical computing, it is not yet practical in many commercial settings. The complexity of using MPI requires extensive training beyond the standard software engineering curriculum. Furthermore, debuggers and other tools would need to be available for commercial software development support. One possible improvement to the current situation might involve adoption of high-level parallel programming language constructs. Although Java provides appropriate mechanisms for a shared-memory approach, distributed-memory languages would allow the use of low-cost distributed memory machines, while still being compatible with shared-memory machines. In the past, different approaches such as High Performance Fortran [6] and the $\mathbb{P}$-languages [3] have been studied extensively. Adoption of the appropriate parallel constructs in popular languages might lead to a broader use of fine-grained parallelism in industry.

Graduate education can address this situation through research into more appropriate tools and systems for fine-grained parallelism. Educational programs can also transfer knowledge about existing techniques to parallel computing infrastructure and tool vendors. To be more effective in these respects in the future, parallel-computing researchers and educators may need to address the concerns of commercial and corporate computing more directly. Some of these are (1) parallel I/O in general and parallel database access in particular, (2) standard in-memory data structures that are appropriate for both fine-grained parallel computation and database access, and (3) portability and robustness of tools.

7 Conclusions

We have indicated some directions in which graduate education could be changed in ways involving parallel computing. We have explained why this would be a good idea both for educating computationally literate people and also for solving important problems in science and industry. We outlined new curricula both in computational science and in computer science.

References

1. Abbas, A. *Grid Computing: A Practical Guide to Technology and Applications.* Charles River Media, 2004.
2. Baffico, L., Bernard, S., Maday, Y., Turinici, G., and Zérah, G. Parallel-in-time molecular-dynamics simulations. *Phys. Rev. E 66* (2002), 057701.
3. Bagheri, B., Clark, T. W., and Scott, L. R. IPfortran: a parallel dialect of Fortran. *Fortran Forum 11* (Sept. 1992), 20–31.
4. Bunn, J., and Newman, H. Data intensive grids for high energy physics. In *Grid Computing: Making the Global Infrastructure a Reality* (2003), F. Berman, G. Fox, and T. Hey, Eds., Wiley, pp. 859–906.
5. Chen, N. High school computing: The inside story. *The Computing Teacher 19*, 8 (1992), 51–52.
6. Clark, T. W., v. Hanxleden, R., and Kennedy, K. Experiences in data-parallel programming. *Scientific Programming 6* (1997), 153–158.
7. Foster, I., and Kesselman, C. *The Grid 2: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 2003.
8. Garcia-Molina, H., Labio, W. J., Wiener, J. L., and Zhuge, Y. Distributed and parallel computing issues in data warehousing. In *Proceedings of ACM Principles of Distributed Computing Conference* (1998), vol. 17, p. 7.
9. Kalé, L. V., Skeel, R., Bhandarkar, M., Brunner, R., Gursoy, A., Krawetz, N., Phillips, J., Shinozaki, A., Varadarajan, K., and Schulten, K. NAMD2: Greater scalability for parallel molecular dynamics. *Journal of Computational Physics 151* (1999), 283–312.
10. Scott, L. R., Clark, T. W., and Bagheri, B. *Scientific Parlallel Computing.* Princeton University Press, 2005.
11. Snir, M. A note on n-body computations with cutoffs. *Theory of Computing Systems 37* (2004), 295–318.
12. Wlodek, S. T., Clark, T. W., Scott, L. R., and McCammon, J. A. Molecular dynamics of acetylcholinesterase dimer complexed with tacrine. *J. Am. Chem. Soc. 119* (1997), 9513–9522.