1994

The Roles of Models, Software Tools, and Applications in High Performance Computing

Leah H. Jamieson

Susanne E. Hambrusch
Purdue University, seh@cs.purdue.edu

Ashfaq A. Khokhar

Edward J. Delp

Report Number:
94-032

THE ROLE OF MODELS, SOFTWARE TOOLS AND APPLICATIONS IN HIGH PERFORMANCE COMPUTING

Leah H. Jamieson
Susanne E. Hambrusch
Ashfaq A. Kohkhar
Edward J. Delp

CSD TR-94-032
May 1994

The Role of Models, Software Tools, and Applications in High Performance Computing *

Leah H. Jamieson† Susanne E. Hambrusch‡ Ashfaq A. Khokhar†‡
Edward J. Delp†

†School of Electrical Engineering
Purdue University, West Lafayette, IN 47907
{lhj,ace}@ecn.purdue.edu

‡Department of Computer Sciences
Purdue University, West Lafayette, IN 47907
{seh,ashfaq}@cs.purdue.edu

May 4, 1994

Abstract

In this paper we identify and discuss technical issues we consider crucial to the HPCC program. The focus is on the usefulness of scalable parallel computers for National Challenge problems. We identify three interrelated aspects of usefulness: performance, programmability, and the role of an application-driven design philosophy. We discuss the importance of algorithm design and computational model development and advocate the design of libraries and software environments to bridge the gap between algorithm designer and application programmer. Finally, we consider the role of applications for solving National Challenge problems.

*This work was supported by the Advanced Research Projects Agency under contract DABT63-92-C-0022. The content of the information does not necessarily reflect the position or policy of the United States Government and no official endorsement should be inferred.

1 Introduction

During the last several years significant progress has been made on the Grand Challenge problems [9]. Crucial to this progress has been the ability of recent architectures to deliver high performance on floating point operations [4, 14] and the development of substantial libraries for scientific computing problems [1, 2, 3]. At the same time, the beginnings of an emerging community of computational scientists are evident and the experience base for scientific computing is growing. Parallel to the advances in solving Grand Challenge problems, a new set of applications, known as the National Challenge problems, has emerged [10]. In contrast to the Grand Challenges, many of the National Challenge problems are in areas for which the experience with scalable parallel computers is minimal, but where the eventual size of the end-user community is immense. This makes issues related to usability of high performance computing even more crucial for National Challenge problems.

The focus of our paper is on the usefulness of scalable parallel computers, with particular emphasis on the National Challenge problems. We identify three interrelated aspects of usefulness: performance, programmability, and the role of an application-driven design philosophy. Because it is not valuable to provide programmability at the expense of performance that requires unreasonable programming effort, we discuss these two aspects together. An important requirement of almost all National Challenge problems is the need to process and manipulate large sets of spatial and/or imagery data [10]. This differs from a majority of the Grand Challenge problems, and leads us to consider the role of applications in hardware and software design. Many of the examples we use to substantiate our position are related to research we are conducting in using high performance computers to solve problems in image processing and computer vision.

The paramount conclusion of our paper is that high performance computing can achieve its goals only if software development is significantly accelerated. We maintain that useful advances in software must meet seemingly contradictory goals: that software development should be carried out in an architecture- and technology-independent environment, but that both algorithms and system software should take full advantage of the hardware features of potentially diverse architectures. We propose four areas of research as fundamental means of reconciling these contradictory demands:

– research on models as platforms for architecture-independent algorithm development;
– development of portable scalable communications libraries;

– further development of scalability measures that, when coupled with models, will give an accurate prediction of algorithm performance across a wide range of problem sizes, machine sizes, and architecture types;
– applications-driven library-based software for bridging the software gap between usability and high performance.

2 Performance and Programmability

If there were no need for high performance, there would be no need for scalable parallel computers. We therefore state the underlying premise: HPCC applications must make good use of a system's capabilities. Although computer vendors continue to demonstrate that machines can be made to be ever faster, it is not appropriate to use improved speed as an excuse for achieving performance that is significantly below a machine's potential performance. In particular, faster machines do not mean that we do not need good algorithms or good system software. In the following, we identify and discuss our position on some of the issues related to performance and programmability.

• *Algorithm design for parallel computers will continue to be more difficult than for serial computers.*
A large number of parameters affect the performance of a parallel algorithm. Besides machine and problem size, these parameters include network and processor bandwidth, network topology, latency, and distribution of the data. We expect that these parameters will continue to play a significant role in future systems. Hence, algorithm design will remain important, even as machines continue to get faster. The challenge is to provide computational models and tools that allow a user to develop high performance algorithms without having intimate knowledge of parallel processing.

• *No architectural convergence is in sight.*
A standard parallel architecture is not likely to emerge in the near future. Besides distributed and shared memory machines, the development of new fine-grained as well as coarse-grained distributed memory machines will continue. In addition, design and development time for parallel machines has sharply decreased. For example, the T3D by Cray Inc. was designed and built in 26 months. A consequence of this architectural diversity is that software development

and compiler construction are lagging behind: compiler and software development have not experienced a comparable shortening of the design cycle. However difficult it may be, it is undesirable for compilers and software to ignore architecture differences if higher performance can be achieved by exploiting the architectural strengths of a particular architecture. In order to improve usability of current and future parallel machines, significant attention is required towards software development.

• *Libraries and software environments will bridge the gap between algorithm designer and application programmer.*
A parallel software environment will allow parallel machines to be used as general-purpose machines and will give applications programmers access to high performance computing. Providing efficient library routines can make programs scalable and portable across different machines. Hence, libraries are a fundamental building block for achieving high performance for application programmers. Libraries should be implemented with the specifics of a parallel machine in mind and implementations should take advantage of hardware as well software features of the underlying parallel machine.

So the question is: What will it take to achieve both ease of use and high performance? In Section 3 we address three aspects of the performance/programmability issue and discuss the approaches we are taking in our research projects. In Section 4 we consider how an applications focus can help in making high performance computers useful. We conclude by summarizing our recommendations.

3 Achieving Performance and Programmability

The simultaneous goals of high performance and programmability lead to immediate contradictions. Programmability is most easily achieved by assuming architecture independence and programmer naivete. High performance is best achieved by assuming architecture expertise and programmer sophistication. We identify three areas where research can help bridge the gap between these opposing points of view, describe the work that we have been doing in these areas, and point to directions for further research.

1. A computational model as a platform for algorithm development.

Coarse-grained machines are emerging as general-purpose parallel machines, while fine-grained parallel machines will continue playing the role of special purpose machines. A successful general purpose parallel machine needs a sound computational model. Such a model should bridge software and hardware and be robust with respect to technological and architectural changes, much in the same way as the von Neumann model for sequential computation. Such a model should also serve as a platform for the design of portable, scalable, coarse-grained algorithms.

In [8], we propose an architecture-independent computational model, the C^3-model. The motivation for this work is the recognized need for a model that (i) accurately reflects the constraints of a coarse-grained parallel machine, (ii) has broad applicability with respect to existing machines, and (iii) allows accurate prediction of performance. A novel feature of the C^3-model is that it evaluates not only the complexity of *computation* and the pattern of *communication*, but also estimates the effect of the potential *congestion*. This is accomplished by a new metric which estimates the effect of link and processor congestion on the performance. The metric can be used without having to specify fine scheduling details and it allows the evaluation of arbitrary communication operations. This high-level abstraction of communication, together with ability to estimate link and processor congestion, distinguishes the C^3-model from other proposed models [6, 19].

No matter how fast HPCC systems become, performance will be limited by the quality of algorithms executed. The development of appropriate models is critical to the ability of users to design quality algorithms for complex HPCC architectures.

2. Development of portable communication libraries.

Efficient and portable communication routines are crucial to many National Challenge problems. Our work on the Intel Delta and Paragon reported in [7] has demonstrated that the performance of communication operations is influenced by relationships among the parameters of a parallel machine, as well as by the relationship of the machine parameters to the amount of data involved. Clearly, the number of processors and the length of the messages influence which algorithms give the best performance. In addition, we found that the set-up cost, the ratio between send and receive time, the bandwidth of the processors and the network, the latency and the bisection width, and the type of synchronization used also influence the performance. Our

work has demonstrated that for a given operation different algorithms perform well for different ranges of input and machine sizes. We are currently building a general communications library which includes, for every communication operation, a set of implementations based on different approaches.

The existence of possible communications primitives will provide users with an important tool for designing parallel algorithms. Such libraries have the potential to enlarge the user community and will improve the performance of user's algorithms.

3. Scalability measures that give an accurate prediction of algorithm performance.

The performance of an algorithm on a serial computer can be accurately characterized by the problem size and processor speed. On the other hand, similar characterization of a parallel algorithm on a parallel computer entails consideration of a far greater number of parameters. These additional factors include machine size, interconnection topology, interprocessor communication speed and bandwidth, and data distribution. This characterization of parallel algorithm performance is generally referred to as the *scalability* of the algorithm. Several metrics for analyzing the scalability of a parallel algorithm-machine pair have been proposed in the recent past [13]. However, there is no precise, commonly accepted definition of scalability. Comparing different scalability measures is difficult and risky. In fact, applying different scalability measures to the same parallel scenario can lead to contradictory results. A better understanding of these issues is essential if we are to harness the computing power of current and future parallel machines.

We address the issues related to scalability by applying different scalability measures to various computer vision and image processing algorithms, and implementing these algorithms on currently available parallel machines. We have compared analytical and experimental results of the scalability of these algorithms, varying the basic algorithm approach as well as the machine size, problem size, and interconnection topology [11, 16]. We have chosen the FFT and list ranking as representative problems whose algorithms typically have highly regular and highly irregular communication patterns, respectively. We observed that, similar to algorithms for implementing communication operations, no single algorithm would give maximum speedup over the entire range of problem and machine sizes. This suggests the use of multiple algorithms to cover the entire range of scalability, where each algorithm scales best only on a fixed range of problem and machine size.

Work on scalability and models should proceed hand-in-hand. A more comprehensive the-

ory of scalability, coupled with architecture-independent models, will give an accurate prediction of algorithm performance across a wide range of problem sizes, machine sizes, and architecture types.

4 The Role of Applications

The ability to process, manipulate, and store/retrieve imagery data in a high speed networked environment is a key issue in several National Challenge problems. For example, in the area of health care the need to store, process, and display medical images from distributed databases is paramount. Such requirements have direct implications on the hardware and software design philosophies for future HPCC systems.

In this section, we consider two aspects of the how applications can affect HPCC research. First, we identify areas in which current systems most clearly fail to meet the needs of National Challenge problems. Second, we argue that application-driven approaches can make progress in some areas where general-purpose methods have not yet successfully provided both programmability and high performance.

1. Architectural and software support for I/O intensive applications.

For many applications in the National Challenges domain, the I/O subsystem of HPCC systems is one of the major bottlenecks. Dedicated disk file systems are often very slow to load or unload. This has become more crucial with the insurgence of high speed networks. Current HPCC systems do not easily support real-time frame grabbers or other image and video digitizers. Similarly, most of the applications require that massive databases be searched, e.g. digital libraries and government records. Also, in applications areas such as intelligent highway systems, real-time processing, and fast I/O is very much desired. Special attention is needed in the design of algorithms, systems software, and in the hardware components of an I/O sub-system to achieve high performance computing goals.

We have designed parallel techniques to implement the JPEG compression algorithm on a massively parallel SIMD computer [5]. Implementing the algorithm in parallel was not difficult; the performance bottleneck arose in reading data into the processor array and writing data out of the processor array in such a way that these communication times did not overwhelm the gains obtained by parallel processing. We have developed a data-independent input re-alignment

algorithm and two data-dependent output re-alignment algorithms. The results show near real-time performance. Using these data re-alignment techniques, the I/O task execution time is decreased by a factor of 10000 on a 16K processor MasPar machine.

In the above example, the I/O problem was handled at the algorithm design level. However, a comprehensive effort that addresses algorithm, system software, and architecture issues is needed in order to reach the point where I/O capability matches current computing power.

2. Support for non-floating point operations.

It is likely that today's systems that deliver highest performance for Grand Challenge problems would also deliver the highest performance for National Challenge problems (analogous to the observation made in 1980's that supercomputers were the fastest word processors), by virtue of their raw speed. However, they may not well be the fastest possible systems for solving National Challenge problems. Many image processing and computer vision problems could benefit from high speed fixed point operations that are not currently available. Many such application problems could be implemented more easily and would run faster if true fixed point support were available. Many of the National Challenge problems are also characterized by the need to do symbolic operations, particularly in database problems.

3. Application-specific software.

General-purpose parallel programming tools have not yet succeeded in providing both ease of use and high speed for applications researchers attempting to use parallel machines [17, 18]. One factor that contributes to the difficulty of this task is that current parallel machines often exhibit anomalous behavior that can have a major impact on performance. Coping with such behavior requires a detailed knowledge of the system that cannot be expected from a typical applications user. High performance computing will not be successful if it requires application researchers to become parallel processing experts in order to reap the benefits that parallel systems can provide.

We do not believe that compilers that are both able to perform machine specific optimization and fast to build (commensurate with the time to do architecture design) will exist in the near future. Application characteristics can provide additional leverage at a number of levels. Application-driven modification of a language (e.g., FORTRAN-P [12]) can facilitate compiler

construction, and can allow attention to be paid to optimizations that will likely have significant impact for that application. Application-specific programming environments such as the Image Understanding Environment [15] can define formats and objects that are recurrent in an application, so that implementations can concentrate on supporting these constructs. Libraries and library tools can play a critical role in achieving high performance on applications, as has been demonstrated for the Grand Challenge problems.

In our work, we are concentrating on the development of application-specific software tools that incorporate machine-specific performance optimizations into a user's program. Library primitives and kernels form the basis of this approach. Cloner is a library-based program development environment for computer vision and image processing (CVIP) that allows users from different backgrounds to take advantage of the computing power provided by multiprocessor machines and the algorithmic techniques designed by parallel algorithms experts [20]. Cloner focuses on how information about the CVIP problem domain can make the high performance algorithms and the sophisticated algorithm techniques being designed by algorithms experts more readily available to CVIP researchers. Algorithm and architecture-related scalability information as well as appropriate machine-specific expertise are embedded within this environment.

Cloner is also a software reuse tool that helps a user design parallel algorithms by building on and modifying algorithms that already exist in an overall system library. It takes advantage of the fact that CVIP algorithms are often highly structured and that many image-, vector-, and array-based algorithms have the same or similar structure. Cloner allows the user to identify similarities between the new algorithm and library algorithms (e.g., median filtering has the same data dependency pattern as image smoothing), queries the user about the new algorithm's principal execution characteristics, and provides the user with code templates from the library that can be modified for the new algorithm. Since these library codes and mappings contain machine-dependent optimizations, the user can build high performance algorithms without intimate knowledge of the target architecture or a high level of parallel programming expertise.

Software tools that take advantage of applications characteristics can make parallel systems more accessible to users. This is a practical approach that has unrealized potential.

5 Conclusions

We focus on usefulness as the fundamental issue that must be addressed in the coming years of HPCC research, and identify accelerated software development as the key to achieving useful HPCC systems. Although compiler technology is obviously critical to making high performance computing accessible to a broad user community, we do not believe that we are close to seeing compilers that are both fast to build and able to take full advantage of hardware features of potentially diverse architectures. We therefore identify other areas in which research can make progress towards meeting the dual goals of performance and programmability. Research in the theoretical areas of models and scalability measures will improve the ability of users to write algorithms and predict their performance. Research in the areas of communications libraries and applications-driven software will provide practical tools for users of HPCC systems. It is only through attention to this full spectrum of theoretical and practical approaches that HPCC systems will be able to fulfill their potential.

References

[1] E. Anderson et al. "LAPACK: A Portable Linear Algebra Library for High performance Computers," *Proceedings of Supercomputing*, 1990.

[2] Richard Barrett et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994.

[3] Z. Bai, J. Demmel, and A. McKenney, "On Computing Condition Numbers for the Nonsymmetric Eigenproblem," *ACM Transactions on Mathematical Software*, Vol. 19, No. 2, pp. 202-223, 1993.

[4] Connection Machine CM-5, Technical Summary, Thinking Machine Corporation, 1992.

[5] G. W. Cook and E. J. Delp, "An investigation of JPEG image and video compression using parallel processing," *Proceedings of the 1994 International Conference on Acoustics, Speech, and Signal Processing*, Vol. V, pp. 437-440, April 1994.

[6] D. E. Culler et al., "LogP: Towards a Realistic Model of Parallel Computation," *Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, May 1993.

[7] S.E. Hambrusch, F. Hameed, A. Khokhar, "Communication Operations on Coarse-Grained Mesh Architectures," Technical Report, Purdue University, West Lafayette, IN, April 1994.

[8] S.E. Hambrusch, A. Khokhar, "C^3: An Architecture-independent Model for Coarse-Grained Parallel Machines", Technical Report, Purdue University, West Lafayette, IN, December 1993.

[9] *High Performance Computing and Communications: Towards a National Information Infrastructure*, A Report by the Committee on Physical, Mathematical and Engineering Sciences, Federal Coordinating Council for Science, Engineering and Technology, Office of Science and Technology Policy, Washington DC, 1994.

[10] *Information Infrastructure Technology and Applications*, Report of the IITA Task Group, National Coordination Office of HPCC, Office of Science and Technology Policy, Washington DC, February 1994.

[11] L. H. Jamieson, et al., "A Library-Based Program Development Environment for Parallel Image Processing," *Proceedings of the Scalable Parallel Libraries Conference*, pp. 187-194, October 1993.

[12] M. T. O'Keefe et al., "The Fortran-P Translator: Towards Automatic Translation of Fortran 77 Programs for Massively Parallel Processors," AHPCRC Preprint #93-021, Minneapolis, MN, to appear in *The Journal of Scientific Programming*, 1994.

[13] V. Kumar and A. Gupta, "Analyzing Scalability of Parallel Algorithms and Architectures," Technical Report TR 91-18, Department of Computer Science, University of Minnesota, Minneapolis, MN 1991.

[14] S. Lillevik, "The Touchstone 30 Gigaflop DELTA Prototype," *Proceedings of 6-th Distributed Memory Computing Conference*, pp. 671-677, 1991.

[15] J. Mundy et al., *IUE Overview Document*, Image Understanding Environment (IUE) Program, October 1992.

[16] J. N. Patel and L. H. Jamieson, "Evaluating Scalability of the 2-D FFT on Parallel Computers," *Computer Architectures for Machine Perception*, pp. 109-116, December 1993.

[17] C. D. Polychronopoulos and U. Banerjee, "Speedup Bounds and Processor Allocation for Parallel Programs on Multiprocessors," *1986 Int'l Conf. Parallel Processing*, pp. 961-968, August 1986.

[18] Q. Stout, "Mapping Vision Algorithms to Parallel Architectures," *Proc. IEEE*, Vol. 76, pp. 982-995, August 1988.

[19] L. G. Valiant, "A Bridging Model for Parallel Computation," *Communications of the ACM*, Vol. 33, No. 8., August 1990.

[20] C.-C. Wang and L. H. Jamieson, "Xcloner: An Interactive Multiplatform Parallel Image Processing Development Environment," *Proceedings of the 1992 IEEE Signal Processing Society Workshop on VLSI Signal Processing*, pp. 287-296, October 1992.