Applications Analysis: Guest Editor's Introduction

DAVID F. SNELLING

Center for Novel Computing, Department of Computer Science, University of Manchester, Manchester M13 9PL, U.K.; e-mail: snelling@cs.man.ac.uk

The primary motivation behind this special issue stems from a desire to see how various scientists approach the task of scientific programming. In general, each scientist must formulate a problem and derive a solution. The steps in this process are not fixed or prescribed, however, they are nonetheless somewhat universal. In the Call for Papers for this issue, I asked each author to describe the entire process from problem formulation to the realization of a solution. Each step in this process can be characterized by a statement, describing the problem, in a notation or language suitable to the current *level of abstraction*. By way of guidance, the following levels of abstraction were suggested in the Call for Papers. These were not enforced, but all the articles roughly follow this outline.

Problem. This level is characterized by a description, in English, of the problem to be solved. It frequently includes a summary of the importance of the problem and some general background information.

Algorithm. An algorithm is a formal, mathematical statement of how the problem is to be solved. In many cases, several algorithms exist to solve the given problem. The authors have endeavored to provide their readers with not only a description of the algorithm used, but a summary of other variants or approaches and a justification of their choice.

Received May 1994
Revised December 1994

Scientific Programming, Vol. 4, pp. 123–125 (1995)
CCC 1058-9244/95/030123-03

Program. The program is the scientific programmer's "clay." He or she molds it to fit the constraints of both the algorithm and the computational platform. To say that there is no artistry in the other steps of the process would be a mistake; however, the process of programming seems to be the "craft" of the scientific programmer. At this level, the authors describe the techniques employed to create a balanced union between algorithm and machine.

Realization. The scientific results of the study and the extent to which the programming task was a success form the basis of this discussion. The success usually manifests itself as a statement of the program/machine performance relative to the ideal sought by the scientist: more on this to follow.

PERFORMANCE

Why it is that the quality of a program should be measured in terms of its performance, and not the elegance of the user interface, the clarity or bug-free nature of the code, or some other aspect, is not easy to explain. But, nonetheless, performance is universally cited as one of the most (if not the most) important issues in scientific programming. Possible reasons are easily enumerated and are not provided here. However, the way in which performance is reported is a secondary motivation behind this issue.

Performance reporting, particularly with respect to the parallel solutions so prevalent in scientific computation (see the PARKBENCH Report in *Scientific Programming*, Vol. 3, No. 2, 1994), has not been subjected to the same level of scrutiny that the science itself has. The worst offenses occur as a result of the misuse of the ill-defined

"Speedup" metric. Therefore, speedup has been discouraged in these articles and is not found in this issue. It can be argued that to ban speedup will not, by itself, improve the quality of performance reporting. However, by providing this focus, it has been possible to produce seven articles with exemplary reporting of performance results.

Performance results are generally reported in terms of "wall-clock" execution time. In some cases where a *performance* metric (rather than simply time) is desired, authors have used various forms of temporal performance, e.g., simulated-forecast-days per hour-of-computation or simply the reciprocal of the execution time.

ARTICLE SUMMARIES

G. Skinner describes a library module, PREMIX, which simulates chemical combustion in a dynamic system. Because of the complexity of the algorithms used and the library structure of the code, the parallelization of this code is far from straightforward. This is supported by the fact that the performance improvements begin to decline after only a small number of processors. This article contains a careful analysis and description of the parallelization process on an Aliant FX/80, shared memory MIMD system.

L. Wolters et al. investigate the complexities of implementing a general-purpose weather model on a SIMD system. The widely used HIRLAM numerical weather prediction model can be run in several modes, most notably employing either grid point or spectral algorithms. The authors describe the techniques necessary to port this model to the MASPAR MP-1 and MP-2. Their analysis includes a fairly convincing comparison of the performance differences between grid point and spectral techniques, a task not frequently attempted.

C. Korn et al. address environmental issues using a 3-D estuary model. Their algorithmic choice, which is a semi-implicit Lagrangian finite difference scheme, presents major challenges to parallel programming. Using this scheme, updating a single grid point requires information from grid points some distance away, the distance being determined by runtime velocities. Thus, simple "nearest neighbor" communication is not adequate. The shared memory KSR-1, used in this implementation, greatly simplifies this problem.

C. Anderson et al. address a real-world problem facing paralyzed individuals, namely controlling a wheelchair. Their aim is to interpret, using neural networks, EEG signals. In this way, intentional changes in mental state could be used to control the wheel chair. They are still some way from a complete working system, but the chronicle of issues involved is fascinating. One of the crucial aspects of this real-time application is the speed at which EEG signals can be interpreted. They report substantial performance improvements (over a workstation) using a 128 processor SIMD machine, the CNAPS server.

E. Glikman et al. use n-body simulation techniques to model the effect of radiation on crystal structures, in this case, copper. The critical algorithmic issue here has to do with the maintenance of "neighbor" lists, which identify those particles likely to influence a given particle. As the management of this list is critical to ensure load-balanced execution, three approaches are presented and implemented on a 28 node, i860 based MIMD system from Meiko.

D. Williams and L. Bauwens describe the use of a relatively standard, flux-based transport algorithm used in fluid dynamics. In this case, there is no specific application concerned, as the program is part of a widely used library. They port this code to an 8192 processor SIMD machine, the MASPAR MP-1. They provide extensive performance comparisons on a complex test problem.

R. Ford and I. Poll report on an application used to study techniques that delay the onset of turbulence over an aircraft wing. Success of one of these techniques (e.g., providing suction, through microscopic holes, on leading parts of the wing) could reduce drag by as much as 30%. According to the authors, a reduction of just 5% would provide a single manufacturer with total market dominance. Their article describes novel techniques for predicting when turbulence will start, a frequency-based algorithm for solving the problem, a collection of scalar optimizations, and a multilevel parallel implementation on a shared memory, MIMD KSR-1.

Although these articles draw from a wide range of scientific fields, the authors have found it straightforward to fit their work into this prescribed framework. This is my belief that is partially due to the existence of a relatively uniform practice of scientific programming. This is in spite of the absence of a universal terminology and the presence of a wide variety of programming

models. The emergence of uniformity in these processes (particularly with respect to use of parallelism) is a sign that scientific programming is evolving from an "art form," accessible to only a few, to a "scientific practice," available to many. Let us hope that this trend continues.

I hope that you, the reader, will enjoy reading this issue as much as I have enjoyed preparing it. A closing word of advice: Don't let a lack of interest in an application area prevent you from reading a particular article. All of these articles have something interesting to say.