SUPERMICROCOMPUTERS

Proceedings of

The Workshop on the Applications of Supermicrocomputer Workstations in Physics and Astronomy, January 20-22, 1984, University of North Carolina at Chapel Hill.

Edited by Steven M. Christensen

SUPERMICROCOMPUTERS

Proceedings of

The Workshop on the Applications of Supermicrocomputer Workstations in Physics and Astronomy, January 20-22, 1984, University of North Carolina at Chapel Hill.

Sponsored by the National Science Foundation.

Edited by Steven M. Christensen

TABLE OF CONTENTS

COMPUTING: A NEW TOOL FOR FUNDAMENTAL PHYSICS

Stephen Wolfram
The Institute for Advanced Study

It used to be said in theoretical physics that one could either "think" or one could "compute". For certainly most believed there was no creative thinking involved in computing, and no computing that could be of value in thinking. Computers were just for finishing off details after all conceptual work was done. Very few fundamental discoveries were made with computers (notable exceptions were in non-linear dynamics and astrophysics). Their use was best delegated to students, or left to the inferior few percent of professional theoretical physicists who computed. When computing had to be done, it was usually quite grungy. Most universities had just one major computer, located in an inefficient computer center. Computer time was expensive. Essentially all computing was done in batch mode. Programs were edited and input on punched cards, then processed with turnaround time between 10 minutes and 10 days. Output was produced on a line printer.

About two years ago, as computers became more affordable, all this began to change. Experimental physics groups bought their own VAX-like computers, and theoretical physicists could often use them without paying exorbitant rates. Fundamental progress seemed to be made in quantum field theory (particularly lattice gauge theories), statistical mechanics and dynamical systems theory using computers. And many older theoretical physicists learned BASIC for their home computers. Computing suddenly became fashionable in theoretical physics. Many theoretical physics groups acquired their own computers, so there was no explicit charge for computer time. Many theoretical physicists learned to use screen editors, test processors, vector (line-drawing) graphics, and electronic mail. In fact, in say the last year, probably about half of all theoretical physicists used a computer at least once: an increase by at least an order of magnitude over a few years earlier.

Despite this, most theoretical physicists still use computers in a rather primitive way. Most turn to the computer only for "big problems": writing a program is still considered a separate part of research from creative thinking. Straightforward computing techniques are used. Most programs are written in FORTRAN. Most programs are quite short: if there is complication, it is to mimic complicated structure in the systems studied, and not because of advanced programming methods. Most theoretical physicists learn only enough about computing to find some way to do their calculations. Computer science, dilute though it may be, has brought up many useful techniques, such as linked lists, recursion, and hash tables. The techniques are straightforward once explained, but very few theoretical physicists have ever encountered them. Computing is still not considered part of theoretical physics education, even though so many trained in physics go to computer-related jobs in industry.

Computer hardware and software just now becoming widely available make it possible for computing to be fully integrated into fundamental theoretical physics research. But for this to happen, the attitudes of theoretical physicists towards computers must change some more. In the end, computing must become a tool like mathematics. It must be used continually, and in a refined way. Theoretical physicists should be able and willing to use powerful computers in the same casual way as they now use hand-held calculators. The multitude of everday problems, not just the few month-long projects, should be treated by computer. Theoretical physicists should learn and use advanced computing techniques as they now do advanced mathematics. They should learn about dynamically-allocated memory and use it instead of fixed-length arrays just as they now learn about contour integration, and use it insead of simple direct integration techniques. And when this has happened, computing will contribute to fundamental physics research just as mathematics now does. In some cases, one may be able to identify major advances specifically made using computers, but usually computing will be so interwoven into the research process that its specific contributions cannot effectively be disentangled. And denying theoretical physicists access to computers will be like denying them access to tables of integrals.

Interactive computing is the key to these advances. It must be possible to change and run programs as ideas develop, rather than having to finish off with the ideas, then write the program, and then run it. The computer must store and organize previously-written program modules, so that they can be accessed and used later like notes or textbooks. Modern microprocessor-based workstations provide the necessary computer hardware. Software is their most critical element. Many of the components for the necessary software environment are already available; some have still to be constructed. In the immediate future, it seems that the software system should be built on top of the UNIX operating system, should allow a high level of graphical output and interaction, and should include the interactive mathematics langauge SMP and the C programming language.

FORTRAN has been the most popular language for theoretical physics for nearly 20 years. It is not inadequate for many reasons. First, as a programming language, it has grown amorphous and illogical, and it can no longer support modern programming techniques and methodologies. And second, it must be used in a rigid non-interactive cycle: a source program is created using a text editor; then the FORTRAN compiler translates it into machine code; and finally the resulting program is executed, input is given, and output is produced. Typically the cycle must be followed many times to debug each program. And most importantly, the program must be written to read in all necessary parameters, and print exactly the required results. No direct interaction with the running program is usually possible, and a complete program must be set up to perform even the smallest problem. Moreover, the FORTRAN language itself has almost no built-in "knowledge" of theoretical physics or mathematics. A library of utilities must be assembled separately for each application.

BASIC is an example of an interactive system that uses an interpreter to execute commands directly and interactively, without prior compilation. But BASIC, like FORTRAN, has almost no built-in knowledge. What is needed is a higher-level language, that already includes a substantial body of mathematical knowledge, and allows mathematical objects to be manipulated and defined directly in mathematical terms. I developed the SMP language to meet this need. A crucial feature of SMP is the ability to manipulate symbolic as well as purely numerical data, so that analytical as well as numerical mathematical operations can be performed. The SMP language was designed to be close to mathematics, making it easy to learn and use, and possible to add new mathematical knowledge in a direct manner. The fundamental functions of SMP incorporate many mathematical skills, such as matrix manipulation, special function evaluation, and symbolic integration. (Precursors of SMP, such as REDUCE and MACSYMA, concentrated on symbolic college-level algebra, rather than attempting a general coverage of mathematics.) And with time, more and more of the mathematical knowledge conventionally found in text books will be accessible in SMP. SMP can then become the medium for almost all mathematical calculations: calculations conventionally done by hand will be done more quickly and carried further, and whole new classes of calculations can be tackled.

But not all calculations in theoretical physics are of the conventional mathematical kind. Many involve explicit simulation of physical systems, or some mathematical idealization of them. Sometimes the calculations involve Monte Carlo numerical evaluation of an integral; sometimes solution of differential equations; sometimes simulation of a Markovian cascade process; and sometimes direct simulation of many components, each behaving according to simple "programs". It is an "experimental mathematics" that is required for these calculations. C appears to be the most suitable current programming language to implement such calculations. Through the UNIX operating system, C is rapidly becoming almost universally available, and almost completely standardized. In fact, on many new computers, it is the first programming language to be implemented. C is a comparatively modern language (about 15 years old), and supports most modern programming techniques (exceptions include coroutines). Such techniques lead to greater efficiency in the writing of programs and in the running of programs. The methodology of "structured programming" consists in such apparent trivia as conventions for indenting lines of code, or, more substantially, making sure that different functionalities are represented by different functions, rather than pieces of a single block of code. Sadly many who preach about structured programming insist on such unfortunate crusades as the complete elimination of "goto" statements. But this should not detract from the fundamental utility of the approach. C allows modern programming techniques such as recursion (functions calling themselves), abstract data types (being able to define composite data types beyond the usual integers, real numbers, etc.) and dynamic memory allocation (building structures as they are needed, rather than specifying their size from the start). Using these techniques, the programs one writes can often follow much more closely the mathematical conception of a calculation. One can build and manipulate data structures—such as linked lists, binary trees and directed graphs. And one can easily apply algorithms such as a quicksort (sorting n-element lists in O(n log n) rather than n^2 time).

A major current drawback of C is that, like FORTRAN, it is not interactive: programs must be written, then compiled, and only then run. What is needed is an interactive version of the C language, in which programs can be created, accessed and executed interactively. I have been developing a system called IXIS to address this need. At the heart of IXIS is an interpreter, that executes C language source code directly and interactively, without compilation. Programs can be executed in a controlled manner, and changed and rerun immediately as necessary. IXIS uses portions of the screen or "windows" to give a graphical interface to different tasks.

When computer programs are used in an exploratory fashion for fundamental research, the real time spent in their development and debugging is usually a thousand times larger than is taken for their final execution. Improvements in the software environment for program development are therefore crucial, and much more significant than small increases in raw computational speed. And probably more significant still are better planning and better knowledge of computational techniques.

Computers powerful enough for theoretical physics are getting cheaper all the time. For several years past, the speed of the computers available to most theoretical physicists has remained roughly the same, but their price has progressively decreased. One will soon have reached the point where every theoretical physicist could and should have a workstation of his own, presumably allocated much as laboratory space is to experimental scientists. Of course, as computers become more closely integrated into theoretical physics research, it will become almost impossible for theoretical physicists to work without a computer at hand. The lowest-level approach is to use a terminal with a telephone connections whenever one is away from one's office. But telephone lines have a limited bandwidth, so modern graphical software environments and high-resolution displays cannot be used through them. A more satifactory approach uses an intelligent terminal (such as the BLIT), with it own central processing unit, to which programs and data are downloaded and uploaded by telephone. However, with increasing software standardization around UNIX, it will not be long before completely autonomous computers compatible with one's workstation will be available at personal computer prices. Finally, there are beginning to be battery-operated fully-portable computers, such as the TRS80 model 100. With flat screen television and bubble memory technologies, computers powerful enough for theoretical physics research should become almost as portable as calculators.

It seems that the raw computational speed of a single-user VAX 11/780 (or equivalent workstation) is now quite adequate for interactive use in theoretical physics research. Future software systems will make computers easier to use, but will introduce more layers of software, requiring more computational power to maintain the same response speed. Graphical programming techniques, with windows, menus and mice (analogue positioning devices) will probably be the most significant near-term advances. For many purposes, typing commands to a computer will be superceded by graphical selection of options. Complementary to this, higher-level computer languages, such as SMP, allow more sophisticated commands to be made succinctly. Moreover, expert systems technology should allow computer command languages to be brought closer in form and usage to natural languages. Increases in computer speed will be most significant in making more sophisticated software systems usuable, and not, for the most part, in allowing slightly longer or larger calculations to be done. The actual running of the final program will almost always take a tiny fraction of the computer time for a whole project.

Fundamentally new classes of theoretical physics computations require increases in computer speed not by factors of ten, but rather, factors of thousands. Current serial-processing computer technology will not achieve such increases in the forseeable future. Parallel processing is essential. There have been many projects to develop parallel processing computers; most have been quite ill-conceived. One promising current project (at Columbia University) consists of connecting together 64 powerful single-board computers. Another (at Thinking Machines Corporation), is building a machine with a 10^5 very simple processors, made with many processors on each custom very-large-scale integrated circuit chip. For suitably parallelizable problems both machines will run much faster than current Cray-class computers. Yet in both cases, one can envisage production models costing about the same as current VAX-like computers. Parallel processing should be of particular value in theoretical physics because it is closer to the mechanisms of information processing in the physical systems to be simulated. But the necessary programming remains a challenge. It is still just possible to make use of vectorized or pipelined computer hardware by compiling FORTRAN or C code; but for truly parallel machines one must use fundamentally new programming languages and techniques.

Computers are now advancing to the point where they can be used throughout fundamental physics research. Computing should become as central a tool in theoretical physics as conventional mathematical techniques such as calculus are now. Theoretical physicists should treat computing much as they now treat mathematics: They should learn it, then if necessary refine and advance it to solve their problems. Of course, using computers, unlike using mathematics, costs money: but it is essential to the advancement of theoretical physics that almost every theoretical physicist get a computer to use interactively as an integral part of his research. (Funding instead a few centralized computer facilities is sheer folly.) Then, just as when other new techniques and methodologies have entered physics, there is little doubt that many of the fundamental discoveries of the next few years will be made with computers.

This work was supported in part by the U.S. Office of Naval Research under contract number N00014-80-C-0657.