317891
12 P.

Massive Parallelism in the Future of Science

Peter J. Denning

11 Nov 1988

RIACS Technical Report TR-88.33

NASA Cooperative Agreement Number NCC 2-387

(NASA-CR-185416) MASSIVE PARALLELISM IN THE FUTURE OF SCIENCE (Research Inst. for Advanced Computer Science) 12 p CSCL 09B

N89-26447

Unclas

G3/62 0217891

Research Institute for Advanced Computer Science

Massive Parallelism in the Future of Science

Peter J. Denning

11 Nov 1988

RIACS Technical Report TR-88.33

NASA Cooperative Agreement Number NCC 2-387

Massive Parallelism in the Future of Science

Peter J. Denning

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report TR-88.33
11 Nov 1988

Massive parallelism appears in three domains of action of concern to scientists, where it produces collective action that is not possible from any individual agent's behavior. In the domain of data parallelism we will design computers comprising very large numbers of processing agents, one for each data item in the result; these agents collectively can solve problems thousands of times faster than current supercomputers. In the domain of distributed parallelism we will design computations comprising large numbers of resource attached to the world network; the network will support computations far beyond the power of any one machine. In the domain of people parallelism we will design collaborations among large groups of scientists around the world who participate in projects that endure well past the sojourns of individuals within them; computing and telecommunications technology will support the large, long projects that will characterize "big science" by the turn of the century. Scientists must become masters in these three domains during the coming decade.

This is a preprint of the column *The Science of Computing* for *American Scientist* 77, No. 1 (January-February 1989).

Work reported herein was supported in part by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Association (USRA).

Massive Parallelism in the Future of Science

Peter J. Denning

Research Institute for Advanced Computer Science

11 Nov 1988

What will scientific computing be like at the turn of the century? Will we have computers capable of 1 teraflop -- 1 trillion floating point operations per second? Will we have an international network that can support data rates of 1 gigabit -- 1 billion bits -- per second? Will every personal computer support 3-D color animated graphics? Will computers see, hear, and speak? Will personal computers be fully portable and radio-linked to the world network? Will we be able to tap data streams produced by large shared facilities such as the supercollider, the space telescope, or the genome database? Will the entire scientific literature be accessible on-line or on optical disk? Will we conduct scientific collaborations over networks that hide the distance between us and our colleagues? If you're like me, you dream about these things, and you believe they are all possible by the turn of the century -- only eleven years from now.

Where must we direct our research energies to make these dreams come true? I have found a rich source of inspiration in two simple words, massive parallelism, as they apply to three domains of action that we must master by the year 2000. By domains of action, I mean sets of well-defined patterns of coordination that allow given agents to perform actions as a group that no one of them could perform alone.

Most of us are fully aware of the first domain; we are already able to perform actions in it effectively. We are dimly aware of the second, but we do not make use of its potential. We are generally unaware of the third, however, failing to recognize it as a domain of serious concern to scientists. I will argue that the realization of our dreams about the future of scientific computing will require us not only to become fully aware of all three domains but to attain mastery in each.

The first domain I call data parallelism. In this domain, we design computers that consist of large numbers of processing elements, one for each item of data in the result. I have discussed this domain in previous columns (*1,2*). The design of algorithms focuses on how to distribute a computation among processors so that most of it involves immediate neighbors and so that total communication time is independent of the size of the problem. The collective action that is possible in this domain is aggregate processing speeds far beyond those ultimately possible in any single processor. For example, the Connection Machine 2, when configured with its maximum of 65,536 processors, can deliver up to about 30 gigaflops, some 30 times faster than the Cray-2; the same design with each processor running 30 times faster would deliver the dreamt-of 1 teraflop. Machines of that speed would permit significant breakthroughs in a large variety of scientific problems, such as full digital simulation of an aircraft in flight in near real time or accurate prediction of weather weeks in advance.

Many of the fundamental algorithms that appear in scientific computing libraries are of the single-operator type: a single function is applied simultaneously to all the data.

Examples are searching, sorting, matrix operations, and the solution of linear and differential equations. For these problems, it is possible to load one single-function program into a control processor that broadcasts each instruction to all of the data processors. The resulting architecture is classified as single instruction stream, multiple data stream (SIMD). The Connection Machine 2 is an example of this type of architecture. It is straightforward to program because conventional sequential programming languages can be applied directly to the task. The many researchers around the world studying this architecture are learning how to program it efficiently.

There are many important problems whose solution cannot be expressed as a single operation over a large data set. Examples include image processing, where the application of processing power among pixels depends heavily on the image, and finite element analysis of static stresses in nonhomogeneous structures, where each homogeneous component has its own set of governing equations. For these problems, there must be a separate program for each of the interconnected functions. The resulting architecture is classified as multiple instruction stream, multiple data stream (MIMD). Much less is known about the construction and programming of this type of architecture. However, it appears that significant improvements in performance are possible. For example, we know from experience that most user codes sustain only about 20% of the full power of a Cray-2. The prototype codes on the Connection Machine 2 sustain about 10% of its full power. Preliminary studies of a dataflow architecture (a form of MIMD) suggest that nearly 100% of the power can be sustained across a wide range of applications. There is clearly a payoff from mastering this architecture.

We can also expect payoffs from hybrid architectures that consist of SIMD machines interconnected in an MIMD structure. A multifunction algorithm can be implemented as a network of SIMD machines for each function.

Another important class of parallel-data problems involves large patterns of bits such as those that might arise in processing images, speech, and other encoded sensory data. The research problem is to design a machine that can be trained to respond to certain patterns; the machine must respond even if the sensory data do not exactly match any stored standard pattern. Many of these architectures are now studied under the rubric of neural networks.

There is already a high degree of awareness of data parallelism in the scientific community, and that the large amount of current research in algorithms and architectures will produce machines capable of teraflop speeds by 1995.

The second domain I call distributed parallelism. In this domain, we design computations that consist of large numbers of components, each of which is a resource attached to the national high-speed (gigabits per second) network. The resources available in this environment include instruments such as the space telescope, supercomputers, special purpose computers, graphics systems, special purpose servers, databases, and workstations. The design of algorithms focuses on how to select resources from among the huge numbers available in the network, how to specify their interconnections, how to start and stop them, how to authenicate data streams and messages among them, and how to control access. Collective action that is possible in this domain involves computations beyond the power of any one machine -- computations so massive that they require the simultaneous application of the computing power of many large systems around the world.

A recent example of an action in this domain is the factoring of a 100-digit number by Mark Manasse of Digital Equipment Corporation in Palo Alto and Arjen Lenstra of the University of Chicago (*3*). They used approximately 400 processors on the international network over a period of 26 days, consuming about one processor-year of computing power a day. The largest number that had been factored previously on a single machine was 92 digits, a problem that requires one-tenth the computational work.

I have already discussed some of the possibilities of this domain in columns on supernetworks and security in networks (*4,5*). In this domain, you could regularly collaborate with researchers around the world without having to know exactly where they are or what kind of computers they use. You could share any program, database, service, or facility with the entire community by registering it as a network-wide resource. You could find quickly the names of registered resources when all you know is their general function, and then you could access those resources without knowing exactly where they are or what computers they use. You could access resources by the same interface irrespective of whether they are local or remote. You could construct programs that simultaneously employ many resources around the network and do not malfunction if those resources are moved to new host computers.

Most of us are only dimly aware of the domain of distributed parallelism in the scientific community. It is easy, however, to appreciate what we could accomplish by investing research effort in mastering it, a goal that I believe we can attain, like mastery of data parallelism, by 1995.

The third domain I call people parallelism. This is the domain in which scientists around the world are able to engage in large collaborative projects that live on for many years despite constant turnovers of personnel. The importance of this domain is already beginning to be apparent: we are undertaking massive projects of long duration in ''big science'' -- projects like NASA's space station, particle physics with the supercollider, and the mapping of the human genome. This is destined to be the dominant paradigm of scientific research by the turn of the century. Our current inability to deal effectively with large, long projects is evident in the large sums of money spent on redundant efforts, the high cost of holding meetings attended by people from around the country, our inability :ntify and change design decisions made in remote parts of a project, and our difficulties integrating the efforts of various subgroups into working systems. Billions of dollars are lost annually because of our inability to maintain large projects effectively.

The design problems in this domain focus on how to provide technologies that support collaborations among scientists and on how to capture and preserve the corporate memories of large projects. In solving these problems, scientists have the opportunity to make a contribution well beyond the scientific community, becoming part of the structure that supports world peace.

I have attempted to provide a glimpse of the world of massive scientific collaboration in a previous column (*6*). An impressive array of supporting technologies already exists (*7*), such as electronic boardrooms that allow a single meeting to span several separated locations, teleconferences that permit two or more people to work on a specific task by simultaneously manipulating objects in identical windows appearing on their individual workstation screens, group networks that enable people with common interests to exchange information and consult with one another, collaboration laboratories that permit scientists to brainstorm together, and project coordinators that track requests and promises among coworkers.

It is an interesting fact that little of this technology has been used outside the projects that developed the prototypes. What blocks its acceptance within the scientific community? I speculate that we are not used to working in teams and we don't know how to use technology effectively to support collaboration. We don't know how to integrate computing technology into our working environments so that it is ''invisible'' or how to tailor a base system rapidly to meet the needs of a particular discipline. We don't know how to capture a corporate memory in a useful database or how to cross cultural and international barriers with computer-based tools.

Even more interesting than these speculations is an idea proposed recently by Michael Dertouzos of MIT, who suggests that much computing (and technology) research has been ''supply-side''. Scientists and engineers select problems to work on and design solutions, which they then make available to the world to use as it sees fit. Dertouzos calls this ''throwing the goodies over the fence.'' Although many of the technological marvels produced this way have been put to good use, the process is inadequate for improving the productivity of people working alone and in groups. A new strategy should be demand-side, taking users' concerns into account. Dertouzos illustrates the demand side with an analogy from architecture: an architect spends a lot of time trying to find out what the occupants of a building will be doing, seeking a design that facilitates their work. If architects followed the supply-side strategy of computer research, they would deliver truckloads of building materials to the site and leave it to the occupants to put up the building. Dertouzos suggests that scientists and engineers could learn a lot from architects.

To the extent that we have been operating on the supply side in our research, we will be concerned only with the question of proving that a system meets its specifications, rather than with the relevance of the system to actions performed in the working environment (*8*). Most of us are unaware of the importance of the domain of people parallelism for the effective conduct of scientific research (*9*). We do not see it as meriting attention by scientists. Given the nature of scientific computing by the turn of the century -- oriented around large, long projects with many individual contributions -- it is important that we focus research effort in this domain and master it.

The scientific computing environment at the turn of the century will support our active participation in all three domains. It will unleash a new era of creativity and innovation that will bring forth the Einstein and Edison in each of us. We will design, test, trace, and use algorithms that solve very large problems on massively parallel processors running thousands of times faster than current supercomputers. We will be able to construct massive computations that simultaneously use many large resources attached to a gigabit bandwidth world network, computations beyond the power of any single machine. We will be able to form massive groups of scientists collaborating in ''big science'' with researchers from around the world and recording their corporate memories in useful databases. It is clear that mastery of the three domains offers an exciting future for scientific research and a new opportunity to share our results with the rest of the world.

References

1. P. J. Denning. 1985. ''Parallel computation.'' *American Scientist* 73, 4 (July-August). 322-323.

2. P. J. Denning. 1985. ''The evolution of parallel processing.'' *American Scientist* 73, 5 (September-October). 414-416.

3. B. Cipra. 1988. ''Mathematicians reach factoring milestone.'' *Science* 242 (21 October). 374-375.

4. P. J. Denning. 1985. ''Supernetworks.'' *American Scientist* 73, 3 (May-June). 225-227.

5. P. J. Denning. 1987. ''Security of data in networks.'' *American Scientist* 75, 1 (January-February). 12-14.

6. P. J. Denning. 1987. ''A New Paradigm for Science.'' *American Scientist* 75, 6 (November-December). 572-573.

7. K. L. Kraemer and J. L. King. 1988. ''Computer-based systems for cooperative work and group decision making.'' *Computing Surveys* 2, 2 (June). 115-146.

8. C. Floyd. 1988. ''Outline of a paradigm change in software engineering.'' in *Computers and Democracy: A Scandinavian Challenge* (G. Bjerknes, P. Ehn, and M. Kyng, eds.). Gower.

9. P. J. Denning. 1988. ''Blindness in designing intelligent systems.'' *American Scientist* 76, 2 (March-April). 118-120.