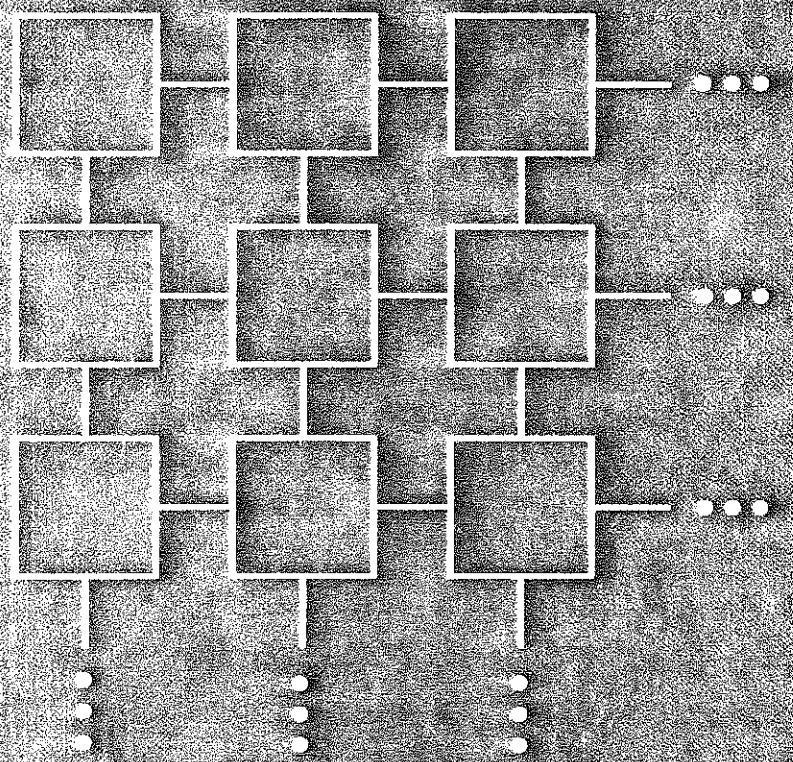
Jorge L. C. Sanz
Editor

Opportunities and Constraints of Parallel Computing



Springer-Verlag

The Physics of the Parallel Machines

Tony F. Chan
University of California at Los Angeles

Abstract

I argue that architectures for massively parallel computers must be designed to go beyond supporting a particular class of algorithms to supporting the underlying physical processes being modelled. I shall discuss specifically physical processes modelled by partial differential equations (PDEs) and argue that an efficient architecture must go beyond nearest neighbor mesh interconnections and support global and hierarchical communications.

1 Architecture Must Support Physical Processes

Massively parallel computers are viewed by many people as a cost effective way of providing the increasing performance required by many areas of scientific computing. Many of these computers have a distributed memory architecture, in which each individual processor is an independent computer with its own memory. To allow the processors to work on the same problem, they must be connected in some fashion which makes it possible to share information efficiently. To be able to use these machines efficiently, new

*Dept. of Mathematics, Univ. of Calif. at Los Angeles, CA 90024. The author has been supported in part by the Dept of Energy under contract DE-FG03-87ER25037. Part of this work was done while the author was visiting RIACS, NASA Ames. This paper is prepared for the Workshop on Opportunities and Constraints of Parallel Computing, December 5-6, 1988, IBM Almaden Research Center.

algorithms have to be designed to take advantage of the underlying interconnection architecture. Very often, the architecture is chosen first and the algorithm designs follow. Many people realize that this is not the optimal design process. Some suggest that the machines should be designed with the algorithms in mind. I believe that we must go beyond even this: the machines should be designed with the physical problems in mind. The reason is simple. Algorithms change but physical laws are constant. The best algorithms are those that capture efficiently the underlying physical processes being modelled and therefore so must the machines if they were to support the best algorithms. Only in this way are we guaranteed that the machines will support a wide variety of algorithms and that they will not become obsolete when better and different algorithms are developed in the future.

2 Parallel PDE $\neq$ Nearest Neighbor Mesh

I shall illustrate my point by concentrating on the case of physical processes modelled by the PDEs of mathematical physics. Surely, this is one of the largest classes of problems whose insatiable demand for performance has provided the main impetus for the new parallel machines. The PDE typically describes the time evolution of certain physical quantities which interact in space. It is usually discretized on a computational grid and the discrete problem is then solved on the computer. On the computational grid, the PDE is often replaced by a spatially local computational stencil. This spatial locality is a direct consequence of the differential nature of the PDE (differentiation is the limit of a spatial differencing process).

Because of this property of locality of the discrete model, many PDE algorithms are also local in nature: the value of the variable at a grid point is updated by using information only from nearby neighbors. It is therefore not surprising that many parallel computers, designed specifically with PDE algorithms in mind, also have a local interconnection architecture. The most common is a nearest neighbor mesh (NN-mesh), e.g. the ILLIAC IV, the ICL DAP, the Goodyear MPP. Even some of the more recent "general purpose" parallel machines have the NN-mesh as a basic architecture (e.g. the hypercubes and the Connection Machine). The new Ametek Series 2010 machine has even abandoned the more global hypercube connections used in earlier versions and rely only on a NN-mesh architecture. The wisdom seems to be

that "Parallel PDEs = NN-mesh." I claim that this is not the case.

The reason for my statement is that while the differential nature of PDEs may seem to be purely local, the PDEs together with boundary conditions often describe global processes as well. This is especially true for steady state problems, i.e. the equilibrium configuration of the physical system after an initial transient phase. Mathematically, the physical systems are often described by elliptic PDEs. Elliptic problems can be characterized as having an infinite domain of dependence: the solution at any point depends on the solution (and the boundary conditions) at every other point. Thus, for example, changing the load at a single location on the span of a bridge will change the deflection at every other location.

Global dependence occur for even time dependent problems. For example, changing the velocity field locally in an incompressible fluid will instantaneously affect the pressure field globally. The presence of different length scales in many physical systems also account for global dependence. In the modelling of the atmospheric circulation, the key issue is to model the large scale and global features such as fronts without having to resolve the finest scale of the molecules in the air. In fact the interaction of the different scales is often the fundamental physical process that physical scientists try to understand. A most notable example is turbulence modelling.

The main point is that while the PDE itself may be local in nature, the solution of a PDE often has global dependence. Moreover, it is often the global features of the solution that we are interested in, not the small scale, local interactions. This is typical of many physical processes.

3 Good PDE Algorithms Must Capture Global Coupling Efficiently

Let us now look at PDE algorithms. Obviously, the simplest ones are local in nature, exploiting the local nature of the discrete model. However, as we have shown above, they do not necessarily capture the underlying physical processes efficiently. For problems with global dependencies, the best algorithms are often also global in nature. An example is the iterative solution of elliptic problems. The simplest, and often the slowest, algorithms are the local relaxation methods such as the Jacobi, the Gauss Seidel methods and

the SOR methods. Faster algorithms employ more global couplings. An example is the class of ADI methods which couples implicitly the unknowns in each coordinate directions alternately. For the same reasons, block versions of these methods (in which a block of points or lines are solved together implicitly) are also faster. The same situation holds for preconditioned conjugate gradient methods, with more global preconditioners usually having faster convergence rates. Of course, the faster convergence comes at a price: more global algorithms often require the solution of coupled systems of equations at every step as opposed to the much simpler local averaging used in the local relaxation methods.

A similar tradeoff also occurs for many time dependent problems, for example, in the choice between explicit and implicit time marching algorithms. Explicit methods are local and inexpensive per step but one must necessarily take many steps to transmit global information. By having only local interactions, one limits the time scale to the smallest spatial scale represented on the grid. This is reflected in the stability limit for the time step. Implicit algorithms are exactly opposite. By allowing a global transfer of information, much larger time steps can usually be taken, as the cost of solving the implicit equations at each time step. Explicit methods were very popular in the early days of scientific computing, due to its simplicity and the limited computing resources available then. However, implicit methods are starting to play an increasingly dominant role, especially in fields such as computational fluid dynamics.

The key in the design and choice of algorithms is to find the most efficient method for transmitting global information. In fact, the essence of most research in the design of PDE algorithms can be described as trying to balance this fundamental tradeoff of global dependency and inexpensive iteration steps. The most efficient algorithms are precisely those that strike the optimal balance between these two conflicting goals. The multigrid algorithm is an example. It captures global information but utilizing coarser grids on which global coupling can be accounted for inexpensively by simple iteration steps. New types of domain decomposition algorithms are being developed for solving elliptic problems which can be viewed as reducing the global coupling of the unknowns to only those on the interfaces between the subdomains. The nested dissection algorithm in sparse factorization is based on a very similar principle. The very recent Rokhlin/Greengard fast multipole algorithm (L. Greengard, " The Rapid Evaluation of Potential Fields

in Particle Systems," ACM Distinguished Dissertation Series, MIT Press, 1988) for particle simulations is another example. In this algorithm, global interaction is accounted for by "lumping" collections of far away particles and using a simplied and less expensive center-of-mass approximation.

These new algorithms have several properties in common: they are almost all *hierarchical* in nature, are based on the divide-and-conquer principle, are nearly optimal in computational complexity, and most importantly, capture the global coupling inherent in the physical problem in an efficient way. The trend in PDE algorithm research is towards hierarchical and implicit algorithms. One can expect that these global algorithms will see increasing use in the near future.

4 Parallel PDE Architecture Must Support Global Communication

Assuming that a "general purpose" parallel PDE computer must be prepared to handle efficiently problems with global dependency in addition to problems with only local dependency, it follows from my previous argument that they must support algorithms that share global information and therefore the architecture must support global communication. They must provide a way for the global information required in these algorithms to be transmitted efficiently.

A simple analogy is the national telephone network: the need for global information is as important as the local ones. The current system is a hierarchical one, with local exchanges, long distance trunk lines and international satellite transmissions. The system works well because it supports the demands of the physical system. I cannot imagine the performance of a purely local architecture for the system, such as having my phone hardwired only to my neighbors'!

Implementing the global algorithms on parallel architectures are often tricky and may not make use of the available processors as efficiently as the local algorithms, but they more often than not make up for this loss of "parallel efficiency" in achieving better "problem efficiency." Incidentally, the optimal hierarchical algorithms are also highly parallel in nature. I suppose this is a reflection of the intrinsic spatial locality of even the global features

of the underlying physical processes.

The hierarchical nature of the optimal algorithms suggests that the optimal architecture should also be hierarchical. The hypercube seems to be a good candidate but other hierarchical architectures may also work well. Further research is needed to design architectures which reflect the physics of the problems to be modelled on them. But certainly a NN-mesh architecture does not seem to be the best in any intrinsic way. For a given technology (number of processor, computational speed and communication speed) and size of problem, the NN-mesh architecture may perform respectably well. But ultimately, ignoring the physics of the problem will not lead to the most efficient use of the parallel machines, especially when the number of processors becomes large.