

The underbody of the X-43 serves as inlet and exhaust nozzle, so models for propulsion, aerodynamics, and structures couple into a meta-model of huge dimensionality.

Computing at the speed of thought

With rapidly advancing technology and greater use of multiple processors, billionfold increases in computing speed are now on the horizon

Designing complex aerospace systems such as aircraft, spacecraft, and launch vehicles proceeds as a series of “what if” questions. What if the wing sweep changes by 14 degrees? What if the solar panels change from configuration A to configuration B? What if a rocket design changes from liquid- to solid-fuel propulsion?

The need for radically faster computing

Despite all the computing speed available today, generating highly accurate answers to major questions such as these may require hundreds or even thousands of hours of computing time on a single processor. A recent structural analysis case involving 100 million unknowns required 12 hr of elapsed time, while a full aircraft aerodynamic analysis based on the non-linear Navier-Stokes equations consumed 50 hr. It appears certain that mathematical models of this magnitude will be needed more often as vehicle designs grow ever more sophisticated.

One of the factors fueling that trend is the “devil is in the details” syndrome. For example, the transonic drag may critically depend on small eddies embedded in the turbulent boundary layer, or the crack growth rate may critically affect the life and maintenance of a structure. Thus the models must resolve local detail while capturing the global design, because the micro- and macroscale behaviors may be strongly coupled.

The models representing various disciplines and subsystems may also interact strongly. A current example is the X-43 hypersonic aircraft, whose underbody performs the function of the inlet and exhaust nozzle so that the models for propulsion, aerodynamics, and structures all couple into a multiphysics meta-model of huge dimensionality.

Engineers must analyze large mathematical models multiple times to assess various alternatives in the search for a better design, whether the search is conducted via formal optimization or by trial and error.

The recent trend toward replacing customary, deterministic safety factors with the probability of failure analysis also demands multiple repetitions of analysis so that a statistically significant number of data points will be generated. With analysis alone, it could take more than 20 hr of elapsed time to conduct an automobile structure crash analysis whose finite-element model may require hundreds of thousands of unknowns for minimally acceptable accuracy.

Naturally, computing tasks such as these, especially when performed repetitively, could not become a routine part of the design process when predominantly single-processor-based computing is used. Still, the critical “what ifs” must be answered. This requires resorting to simplifying assumptions and reduced analysis fidelity at the price of uncertainty and a risk of

by **Jaroslav Sobieski**
and **Olaf Storaasli**
Senior research scientists,
NASA Langley

significant errors. These errors may not be uncovered until later in the design process (or even worse, at the manufacturing, prototype testing, or product operation phases), often resulting in staggering costs.

With radically accelerated computing speed available, engineers could revisit earlier decisions in response to new information, just as writers are enticed to improve their documents because word processors enable easy revisions.

The opportunity

Now imagine that today's typical processor computing speed of 1 billion FLOPS (floating point operations per second) is accelerated by another factor of a billion, to 10^{18} FLOPS (exa-FLOPS). This would reduce the elapsed time needed for solving a now-impossible aerospace application from 1,000 hr to less than 4 millisecon.

The benefits to design processes would be dramatic: Essentially, engineers would perceive the answers as instantaneous. The computer response to their queries would become analogous to the assistance modern word processors provide letter writers by highlighting spelling and grammatical mistakes on the screen almost instantaneously.

By the same token, the time elapsed to solution would depend on the engineers' ability to formulate "what if" questions and digest answers, rather than on the computer response rate. Many design options that might be superior, but could not be examined within today's time and budget constraints, could then be explored in depth without the current impediment of long waiting times.

In effect, engineers would be encouraged to revisit earlier decisions in response to any new information, just as writers are enticed to improve their work because word processors allow revisions to be made so easily. The resulting improvement in the design's quality, reliability, and timeliness cannot be overstated.

The potential

The prospect of computing a billion times faster than we do today is not a fantasy. Because of miniaturization manufacturing techniques, the density of the electronic components packed on a single chip has been doubling approximately every 18 months for decades. This doubling,

known as Moore's Law, has yielded a similar increase in computing speed in terms of operations per second. Even though chip miniaturization may slow down or even halt as it approaches the molecular or atomic scale in about a decade, by then it will already have accelerated computing speed by about a thousand times.

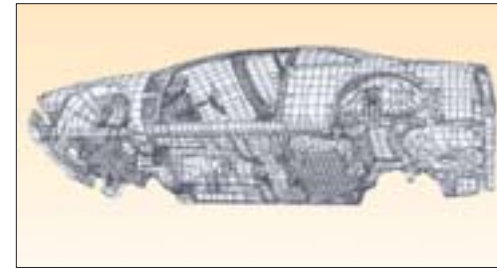
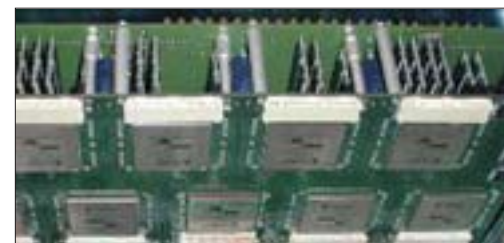
Paralleling such single-processor acceleration is the aggregate acceleration achieved by harnessing thousands of processors to operate concurrently. As one data point, IBM's Blue Gene computer will have 1 million processors. The multiplicative effect of faster processors operating concurrently makes plausible our prediction of increasing computing speed by a factor of a billion.

Multiprocessor computer technology, unlike that of single processors, is open ended, with no natural limit on the number of processors that may run concurrently. Computer technology has already implemented alternatives for many processors working simultaneously on a problem. These range from clusters of individual processors packaged in single boxes to reconfigurable computers such as those based on FPGAs (field programmable gate arrays).

FPGAs provide millions of electronic building blocks (gates) and special "gateway" that lets users configure them for solving applications using thousands of inherently parallel, "virtual" processors. Reconfigurable computers also are open ended (they can incorporate multiple FPGAs), and are limited only by the size of the gates and FPGA modules. (NASA has initiated a program to develop FPGA technology for aerospace applications, funded for a total of \$15 million over the next four years.)

FPGA gate capacity has recently been doubling at a rate exceeding even Moore's Law. Finally, computational grid technology assembles computers of all these types into a geographically distributed network serving many users. Each user may view the grid as a single, virtual computer enabling the operation of many computers, different in type and number as required by the problem at hand.

For the more distant future, the concepts of quantum computers, chemical computers, and DNA-inspired computers are being postu-



lated. Although it is too early to speculate on their potential, it should be noted that all of them are also parallel computers.

Examples of engineering applications expected to benefit immediately from such revolutionary computing speed-ups include multidisciplinary aircraft optimization using high-fidelity finite element analysis; computational fluid dynamics (Navier-Stokes) with unsteady aerodynamics and multimission performance; aerothermodynamics of the reentry vehicle, including chemistry effects of superheated and rarefied atmosphere; and analysis of various cases of an automobile crash that must be accounted for as constraints in car body design.

The problem

Expectations of great elapsed time reductions via concurrent computing must be tempered by the realization that one key ingredient is missing. That ingredient is the development of parallel solution algorithms that harness the collective power of the parallel computers on which they will operate.

Most current solution methods have evolved over the last 300 years from sequential pencil-and-paper procedures that reflect the way humans deal with numerical or logical problems. In most cases, these solution methods simply cannot exploit the full potential of a large number of concurrently operating processors.

Engaging many processors is relatively straightforward when data, however voluminous, are not coupled by equations, as with multiple telemetry channels and pixel-by-pixel imagery. However, in engineering calculations supporting design, the data are both voluminous and coupled. Hence, the solution of simultaneous equations must be embedded in the data processing.

There are at least two reasons why distribution of such a solution over many processors is not straightforward. Most of the matrices involved in engineering applications are sparse and, if matrix factoring is the solution method used, the null entries often become non-zero in a pattern that is difficult to predict, thus increasing the memory requirements.

Furthermore, some of the data, input or interim, must be shared among the processors. The resulting interprocessor data traffic slows down the effective solution rate and increases with the volume of data in the equations so that the law of diminishing returns sets in.

Structural analysis using substructuring illustrates this point. Dividing a structure into more and more substructures engages more processors; but at the same time it introduces new computations (and data traffic) at the substructure interfaces, detracting from the gain. At the limit, when each substructure becomes a single finite element, paradoxically, the substructure approach returns to where it started.

Consequently, the law of diminishing returns limits the number of processors that can be used productively in any substructuring approach. Furthermore, the relative merits of various solution methods for the same problem

In most cases the traditional sequential solution methods simply cannot exploit the full potential of a large number of concurrently operating processors

also change as one moves into a parallel computing environment.

For example, consider again a large set of sparse equations, which are encountered in many engineering applications. Matrix factoring is a solution algorithm favored in a conventional, single-processor implementation, because it completes the task in a finite number of operations. However, it may not distribute well over many processors.

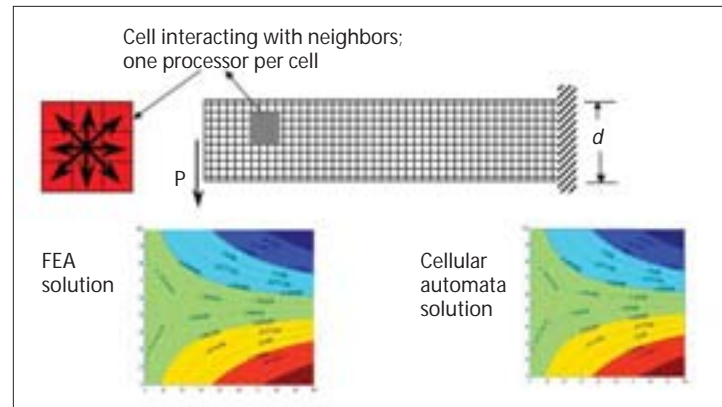
By contrast, an iterative method may engage, at the limit, as many processors as there are individual equation terms, leaving the sparsity undisturbed. Its drawback is in the number of iterations to converge, which may be large and problem dependent. Moreover, convergence to an accuracy comparable to that of factorization methods may not be guaranteed for some problems. Nevertheless, iterative methods may outperform matrix-factoring methods for a favorable combination of the problem and computer architecture, because of its inherent scalability.

In considering scalability as a criterion to qualify a method for use in parallel computing, one should realize that perfect scalability (that is, reduction of a single processor time from 1 to $1/N$ for N processors) is seldom, if ever, attainable. The obstacle to perfection is in that part of

A detailed automobile structure is rendered via a high-fidelity finite element model.

FPGA hardware enables the creation of a multiprocessor computer tailored to the problem at hand.

Finite element analysis is simulated by cellular automata as interactions of adjacent cells; the stress distribution results agree with the conventional approach.



the algorithm that cannot be distributed. Indeed, if the elapsed time on a single processor is a sum of the times for the perfectly distributable part (T_D), and for the nondistributable part (T_{ND}), the latter limits asymptotically the elapsed time reduction.

This limitation is known as Amdahl's Law. It suggests that the greatest gain from parallel computing is achievable in methods whose ratio T_{ND}/T_D is as small as possible.

The remedy

Fortunately, many encouraging prospects for replacing conventional solutions with new, intrinsically parallel ones are emerging. One such approach for aerodynamic analyses replaces the differential equations of a continuum with a direct simulation of the molecular collisions. Similar application to structural analysis directly models the elemental force interactions and dimensional compatibilities.

These simulations, known as the cellular automata methods, inherently engage large numbers of processors simultaneously. They implement, in effect, the idea that a phenomenon's complexity is the result of a very large number of very simple interacting events. A recent book by Stephen Wolfram, *A New Science*, examines hundreds of cases illustrating this notion across many branches of science.

Such new developments require considerable investment, creativity, and a long lead time. It is easy to predict that if these developments are not vigorously pursued, a vicious cycle will set in: Demand for parallel computing technology will not develop as long as potential users perceive the utility of such technology as limited, and the low demand will make software developers less eager to invest in methods that would make that utility greater.

A look into the future

Fortunately, gearing up for parallel computing need not be an "all or nothing" proposition. A

gradual, three-pronged development approach suggests itself.

The first avenue of development would be to "harvest the low-hanging fruit" by aggressively implementing opportunities for the use of legacy codes, unchanged, in parallel computing. Applications for this approach are abundant. One example would be Monte Carlo methods, which are now gaining

importance because of growing interest in designing to a probability of failure instead of the traditional deterministic safety factors. Other examples are optimization guided by gradients computed by finite differences; optimization by genetic algorithm; and incremental nonlinear analysis. All these applications rely on analyses of many different designs. Given enough processors, all these analyses may be done in the time it takes to perform only one.

The second avenue would be rebuilding the legacy codes to tailor them to parallel architectures without changing their underlying algorithms. Consider, for example, a typical finite-element code. In an existing code, it is likely that the load-deflection portion of the code implements one of the matrix-factoring techniques that, for the reasons discussed above, is a candidate for replacement.

The third approach would be a radical revision of the underlying solution algorithms to abandon sequential thinking in favor of inherently parallel approaches. An example would be a cellular automata solution to problems in continuum mechanics.

Opportunities for the first avenue of development are ready to be exploited now at low cost; they may produce spectacular results while buying time. The second avenue occupies a middle ground on the time and cost scale. Innovative research in the third has truly revolutionary potential, at a proportionally higher cost and longer lead time.

The time to initiate the phased, three-pronged development is now. If sufficient resources are committed, the notion of computing at the speed of thought will start to benefit the engineering community soon, and its full impact may begin to be felt within this decade. The attendant gain for the computer builders will be in breaking through the "chicken vs. egg" syndrome, and in a resulting radical expansion of customer demand for and acceptance of parallel computer hardware. \blacktriangle