Are Parallel Computers "Special Purpose?"

By John Gustafson
Computational Scientist
Ames Laboratory-DOE

"The development of massively-parallel computers has led some people to confuse these computers with supercomputers. It can be shown, however, that massively-parallel computers are a more specialized type of computer because their range of application decreases as the number of processors increases." —Jack Worlton

The wagons are circling around traditional vector supercomputing. Worlton's definitions show a desire to keep the term "supercomputer" consistent with beliefs of the late 1970's and early 1980's about what a supercomputer is. This is a view all too familiar to those of us who have been using massively parallel computers: that only serial processors are "true" supercomputers, and only serial processors are general-purpose.

The reasoning behind this opinion usually makes use of one or more of the following ideas:

- *If the application isn't highly parallel, then a parallel computer wastes a lot of hardware.*
- *If the application isn't highly parallel, the slow sections destroy performance.*
- *If it doesn't easily run a dusty Fortran deck, it's not a supercomputer.*

Certainly there is precedent for massively parallel machines being specialized… collections of one-bit processors such as the ICL DAP, Goodyear MPP, and Connection Machine, come to mind. What makes these machines special-purpose is more the shared instruction stream and the ultra-simplicity of each processor than the fact that there are many processors.

The more recent ensemble machines, with autonomous, robust processors, have proved so successful at so many applications that they hardly seem "special purpose." To pick a particular example, the massively parallel NCUBE/ten has been used in the following "special purpose" functions:

Fluid Dynamics	*Database Management*
Structural Analysis	*Semi-Empirical Chemistry*
Image Processing	*Transaction Processing*
Radar Simulation	*Neural Networks*
Chess	*Seismic Data Processing*
Robotics	*Particle-In-Cell Methods*
Galaxy Modeling	*Factoring Large Integers*
Machine Vision	*Oil Reservoir Simulation*

Arguments against massively parallel computing claim that large ensembles are "inefficient," with hardware sitting idle for serial applications. This argument is seldom applied to the complicated hardware inside a traditional supercomputer. At any given time, how many of the pipeline stages in a serial vector supercomputer are busy? How much of the monolithic memory is working on the application? Unless the application happens to continuously and simultaneously require, say, an integer add, a floating-point add, a floating-point multiply, a reciprocal approximation, and four memory references, most of that very expensive hardware is just consuming electricity. Users of such machines have grown used to sustained performance less than 10% of the theoretical peak. In contrast, the 1024-processor NCUBE seldom gets less than 30% of its peak.

At Sandia, we compared a 30,000-line radar simulation program running on an NCUBE/ten and one processor of a CRAY Y-MP. The CRAY version had been optimized, using the SSD for faster I/O and tuning compute-intensive routines. The NCUBE/ten is currently running 6.2 times faster than the Y-MP. The reason was traced to the greater *flexibility* of the massively parallel system at branching, scalar operations, and irregular task sizes. Work done at Caltech points to the same conclusion: the more complex the application, the greater the advantage of machines like the NCUBE over vector supercomputers.

If an application is inherently serial, why assume most of the processors in an ensemble must sit idle while that application runs? These arrays can be, and in practice are, shared by multiple users. If a restaurant has many tables, it doesn't mean it's specialized to serving only banquets. On the contrary, it is easy to allocate just the right number of processors to a job. Can a vector machine adjust its vector lengths to suit the needs of each application?

While vector supercomputers are still faster at running dusty-deck Fortran than other machines, I wonder how much longer we can assert that those dusty Fortran decks belong at the leading edge of computing. Let me offer the flip sides of the arguments above:

- *If the application isn't highly serial, then a serial computer wastes a lot of human time.*
- *If the application isn't highly serial, the parallel sections result in discarded performance.*
- *If you can't or won't modify your dusty Fortran deck, then you're not doing supercomputing.*

"Easy to use" should not be confused with "general purpose." The concepts are independent. The idea that supercomputers should be easy to own, program, or access is contradictory. The reason for the "super" in the word "supercomputing" is that neither the computer nor the effort to use it are ordinary. To solve the biggest problems humanly possible, some mix of high hardware cost, complex facilities, extra programming effort, lower reliability, and limited access is tolerable. That is why phrases like "personal supercomputer," and "affordable supercomputer" abuse English. The fact that parallel machines trade ease-of-use for high performance is even more reason to consider them supercomputers.

If we return to the original meaning of "supercomputer," and resist the tendency of vendors to pull the word over to their product, it seems clear that the term *must* include massively parallel computers. ■