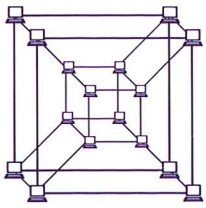
Why there isn't a parallel MATLAB

Our experience has made us skeptical

by Cleve Moler

There actually have been a few experimental versions of MATLAB for parallel computers. None of them has been effective enough to justify development beyond the experimental prototype. But we have learned enough from these experiences to make us skeptical about the viability of a fully functional MATLAB running on today's parallel machines. There are three basic difficulties:

- Memory model
- Granularity
- Business situation



A 16-node hypercube parallel computer. Each node can send messages directly to its nearest neighbors and indirectly to all other nodes.

Memory model

The most important attribute of a parallel computer is its memory model. Large-scale, massively parallel computers have potentially thousands of processors and *distributed memory*, that is, each processor has its own memory. Smaller scale machines, including some high-end workstations, have only a few processors and *shared memory*.

A good example of a distributed memory parallel computer is one of the first commercially available parallel computers, the *Intel iPSC*, where we tried to make our first parallel MATLAB almost ten years ago. It had up to 128 nodes—each a separate single board computer with an Intel microprocessor and maybe half a megabyte of memory. In principle, each node could execute a different program, but we usually ran the same program on all of them. Each node could send messages directly to its nearest neighbors and indirectly to all the other nodes. The whole machine was controlled by a front-end host, which initiated tasks, collected results, and handled all I/O.

We ran MATLAB on the host and gave names with capital letters to the functions in the parallel math library. So `INV(A)` or `FFT(X)` would start with a matrix in the host memory, split it into equally sized submatrices, send each of the submatrices to a node, invoke the parallel routine, and then collect the results back on the host. It took far longer to distribute the data than it did to do the computation. Any matrix that would fit into memory on the host was too small to make effective use of the parallel computer itself.

The situation hasn't changed very much in ten years. MATLAB is a lot bigger, and parallel computers are a lot faster. But distributed memory is still a fundamental difficulty. One of MATLAB's most attractive features is its memory model. There are no declarations or allocations—it is all handled automatically. The key question is: *Where are the matrices stored?* It is still true today that any matrix that fits into the host memory should probably stay there.

Granularity

A little over five years ago, we had a parallel MATLAB on a shared memory multiprocessor, the Ardent Titan, but we didn't tell the world about it. The most effective use of this machine, as well as today's multiprocessor workstations, is already done automatically by the operating system. MATLAB should run on only one processor, while other tasks, like the X-Windows server, use the other processors. In typical use, MATLAB spends only a small portion of its time in routines that can be parallelized, like the ones in the math library. It spends much more time in places like the parser, the interpreter, and the graphics routines, where any parallelism is difficult to find.

There are some special situations where parallel computation within MATLAB would be effective. For example, suppose I want to find what fraction of a large number of matrices have eigenvalues in the left half plane. The obvious place to parallelize this is on the outer loop. It's not necessary to use more than one processor to generate a single matrix or to compute its eigenvalues. The only place the processors would need to cooperate is in merging their final counts. However, to get MATLAB to handle this kind of parallelism would require fundamental changes to its architecture.

Business situation

It doesn't make good business sense for us to undertake fundamental changes in MATLAB's architecture. There are not enough potential customers with parallel machines. Most of the MATLAB community would rather see us devote our efforts to improving our conventional, uniprocessor software. So, we will continue to track developments in parallel computing, but we don't expect to get seriously involved again in the near future. ■

Cleve Moler is chairman and co-founder of The MathWorks. His e-mail address is moler@mathworks.com