Message-Passing Environments - Closing Notes[*]

Charles Severance
Kevin Dowd

In this chapter we have looked at the "assembly language" of parallel programming. While it can seem daunting to rethink your application, there are often some simple changes you can make to port your code to message passing. Depending on the application, a master-slave, broadcast-gather, or decomposed data approach might be most appropriate.

It's important to realize that some applications just don't decompose into message passing very well. You may be working with just such an application. Once you have some experience with message passing, it becomes easier to identify the critical points where data must be communicated between processes.

While HPF, PVM, and MPI are all mature and popular technologies, it's not clear whether any of these technologies will be the long-term solution that we will use 10 years from now. One possibility is that we will use FORTRAN 90 (or FORTRAN 95) without any data layout directives or that the directives will be optional. Another interesting possibility is simply to keep using FORTRAN 77. As scalable, cache-coherent, non-uniform memory systems become more popular, they will evolve their own data allocation primitives. For example, the HP/Exemplar supports the following data storage attributes: shared, node-private, and thread-private. As dynamic data structures are allocated, they can be placed in any one of these classes. Node-private memory is shared across all the threads on a single node but not shared beyond those threads. Perhaps we will only have to declare the storage class of the data but not the data layout for these new machines.

PVM and MPI still need the capability of supporting a fault-tolerant style of computing that allows an application to complete as resources fail or otherwise become available. The amount of compute power that will be available to applications that can tolerate some unreliability in the resources will be very large. There have been a number of moderately successful attempts in this area such as Condor, but none have really caught on in the mainstream.

To run the most powerful computers in the world at their absolute maximum performance levels, the need to be portable is somewhat reduced. Making your particular application go ever faster and scale to ever higher numbers of processors is a fascinating activity. May the FLOPS be with you!