An Approach to Parallel MxN Communication

Felipe Bertrand Yongquan Yuan Kenneth Chiu
Randall Bramley*

July 26, 2003

High-performance scientific computing now faces problems that span multiple scales, domains, and disciplines. The resulting multiphysics simulations are composites of highly-specialized codes developed by large, diverse, and sometimes geographically distributed research teams. As the complexity, size, and specialization of the applications increase, so do the benefits of dividing the application into independent components that can be developed and tested separately. This requires a fast and efficient mechanism to share the large parallel data structures that are used in scientific applications. The "MxN problem" is the transfer of data between two scientific parallel programs with different numbers of processes on each side.

We have developed a lightweight solution to the MxN problem based on an MPI-I/O interface. This approach builds on existing technology, emphasizes the easy migration of current applications, and simplifies the unit testing of the components while maintaining parallel high performance throughout the application. The system is based on defining a new device for the ROMIO implementation of MPI-I/O. A key element of the design of ROMIO is an abstract-device layer (ADIO). This layer consists of a relatively small set of basic functions for parallel I/O. Adding a new backend for ROMIO consists simply of implementing the ADIO interface for a new hardware device. Current devices for ROMIO include a default device for the UNIX file-system, a device for NFS file-systems, and a device for the parallel virtual file-system (PVFS). We created a new backend, the MxN device, which allows the application to transfer data using the regular MPI-I/O interface, as if writing to a file.

Data can be exchanged between components written in different languages, started a different times, and using different MPI implementations, provided they use ROMIO for the underlying I/O library (this includes MPICH, LAM-MPI, HP MPI, SGI MPI, and NEC MPI). Neither communicating component need be aware of the number of processes in the other component.

The performance of this MxN system was compared on Thor to the two other general methods for transfering MxN data: serializing the I/O through MPI process 0, and using a parallel file system and exchanging the data through files. Figure 2 shows that MxN provides higher performance than either of those, even with the same degree of parallelism as the underlying file system does.

*Work supported by National Science Foundation Grants 0116050 and EIA-0202048, and Department of Energy's Office of Science SciDAC grants.

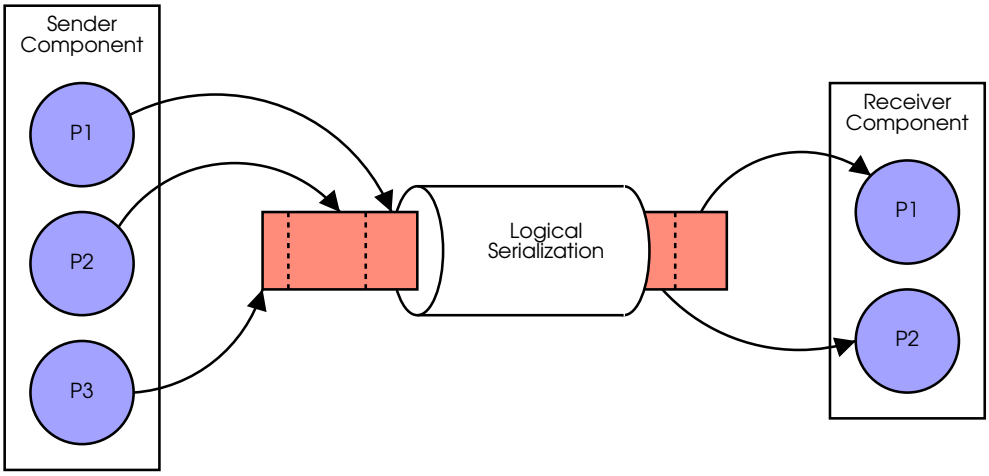

Figure 1: Data is logically ordered, but transfered in parallel.

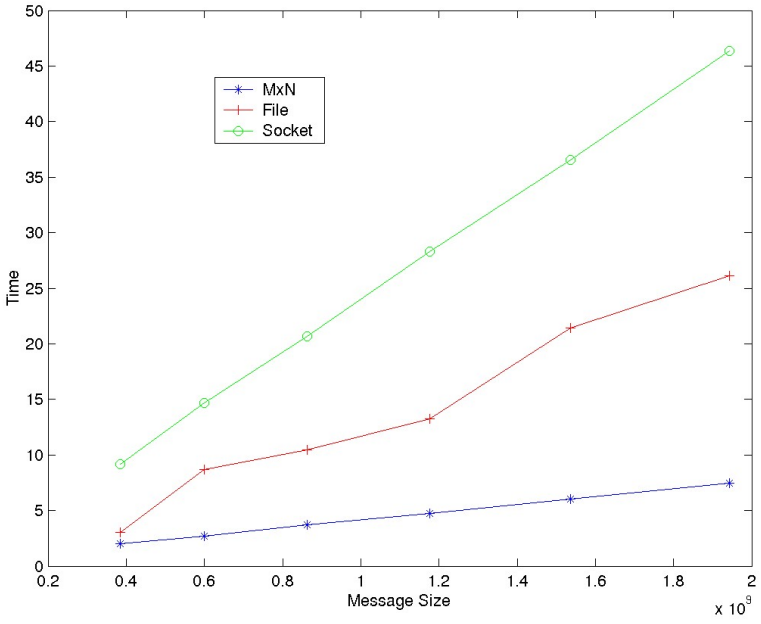

Figure 2: Discretizer-Solver Communication Times