Distributed Object Network I/O Library (DONIO) on Intel Paragon

Problem: DONIO is designed to speed up *non-sequential* I/O for Paragon applications in which all processors open a large multi-megabyte shared file for concurrent direct access. This mode of access is common in finite element codes that are based on subdomain decomposition. The data for material properties or boundary conditions can be stored in shared files. This arrangement provides flexibility in solving the same problem with varying numbers or configurations of processors without rearranging the data files. On the Paragon, this mode of parallel access incurs a high overhead since the Parallel File System (PFS) strictly adheres to the OSF/1 standard, which in effect serializes the I/O to prevent any anomalous behavior.

Approach: DONIO offers a UNIX-like interface consisting of the 'C' callable primitives do_open, do_read, do_write, do_lseek, do_lsize, do_flush and do_close, which are similar to the corresponding UNIX and NX routines. DONIO uses the simple idea of caching the *entire* disk file into the aggregate memory on the multiprocessor. do_read and do_write access the cached copy in memory instead of the disk file. Actual disk operations in DONIO are performed only during do_open for read-only and read-write files, and do_close for read-write and write-only files. DONIO operates on large blocks of contiguous data to take full advantage of RAID 0 striping across multiple disks.

The major advantage provided by DONIO is that I/O operations on the cached file exploit the entire aggregate bandwidth of the network of processors. For certain types of concurrent disk access patterns, this improves I/O performance over the standard Intel NX calls by an order of magnitude. The greatest difference between the performance of DONIO and standard PFS is exhibited when a large number of relatively small read operations is performed concurrently. Such file access patterns are common in finite element codes that are based on subdomain decomposition.

Progress and Results: The data for the graphs are obtained from an example code that generates, writes and later rereads the element-to-vertex list for a three dimensional grid. Three problems were used for testing: a small $41 \times 41 \times 31$ (48,000 elements), a medium $81 \times 81 \times 61$ (384,000 elements), and a large $121 \times 121 \times 91$ (1,296,000 elements) problem. Timings for NX native routines on the largest problem were over 1,000 seconds.

Future Work: Currently DONIO can handle files up to 2gigabytes (based on 32bit integers). We are exploring enhancing DONIO to handle access to much larger hundred gigabytes files.

Sponsor: DONIO was supported by the PICS (Partnership in Computational Sciences) Computational Techniques and Software Tools projects.

For additional information contact:

Ed F. D'Azevedo or Charles H. Romine
Mathematical Sciences Section
Oak Ridge National Laboratory
P.O. Box 2008, Bldg. 6012
Oak Ridge, TN 37831-6367
E-mail: Ed: efdazedo@msr.epm.ornl.gov, Charles: rominech@ornl.gov
Phone: Ed: (615) 576-7925, Charles: (615) 574-3141
FAX: (615) 574-0680