Hierarchical Data Structure and Massively Parallel I/O

J. Werne,[a*] P. Adams[b] and D. Sanders[b]

[a]Colorado Research Associates (CoRA), NWRA, 3380 Mitchell Lane, Boulder, CO, USA

[b]U.S. Army Engineering Research and Development Center (ERDC) Major Shared Resource Center (MSRC), Vicksburg, MS

KEYWORDS: Input/Output, MPP Computing, High Performance Computing

We present an example of what we call Hierarchical Data Structuring (HDS) and associated performance gains for an MPP application with massively parallel data-transfer needs. The technique involves relatively little programing effort, yet overall speed-increase factors on 500 Cray T3E nodes range from 1.4 to 2.1 or more, depending on the number of data files created. We expect similar gains may also be realized on other MPP architectures.

1. INTRODUCTION

Current-generation massively parallel computing platforms allow opportunities for scientific inquiry that were impossible only a few years ago. Complex nonlinear problems that offer only weakly nonlinear solutions through traditional analytic methods are being directly attacked daily through high-performance computational technology at the DoD and NSF Supercomputing Centers. The opportunities for scientists are growing, but so too are logistical difficulties. One such of these problems is the subject of this brief note: Massively parallel Disk I/O, which refers to performance tuning associated with parallel data transfers to the MPP disks and then migrated off-line to a long-term data storage facility. In particular, we describe a technique we call "Hierarchical Data Structuring" (or HDS) which can speed up massively-parallel I/O-limited calculations. We present an example in which the speed-up can range from 1.4 to 2.1 or more, depending on the number of data files involved.

2. ABL TURBULENCE APPLICATION

As a vehicle for demonstrating our approach to massively parallel data handling, we describe high-resolution turbulence simulations on the ERDC MSRC T3E in support of the Air Force AirBorne Laser (ABL) Program. The scientists conducting the simulations are Joe Werne and Dave Fritts of Colorado Research Associates (CoRA). Their ERDC

*Supported by AFRL F19628-98-C-0030, AFOSR F49620-98-C-0029, and ARO DAAD191-99-C-0037. CoRA is a division of NorthWest Research Associates (NWRA), 14508 NE 20th St, Bellevue, WA 98007.

supercomputer work involves the highest-resolution pseudo-spectral calculations of atmospheric turbulence processes (such as wind shear and gravity-wave breaking) currently being conducted by anyone. See the March 2000 issue of the ERDC MSRC Resource Newsletter [3] or the Spring 1999 CEWES MSRC Journal [6] for further information on the ABL project. Scientific results from the project are reported in the publications listed with the references below [4,2,5,1].

3. ERDC T3E AND MASS STORAGE FACILITY

The T3E at ERDC has 544 Processing Elements (PE's). 512 are application PE's that are interconnected to form an $8 \times 8 \times 8$ wrap-around mesh. An additional 32 PE's are networked together in a partial plane and used as operating-system, command, or spare PE's. Each PE includes a 600-MHz DEC Alpha 21164 CPU, 256 MB of main memory, an 8 KB direct-mapped primary data cache, an 8 KB instruction cache, and a 96 KB three-way associative secondary cache for both data and instructions. 780 GB of file space is available for user (/tmp) workspace.

Long-term archival storage is handled by the ERDC Mass Storage Facility (MSF), consisting of an SGI Origin 2000 front end to a CRAY J16/4512. The J16 migrates data to three StorageTek robotic silos. Data is transferred between the T3E and the MSF across a High-Performance Parallel Interface (HiPPI) network, capable of 290 Mbps sustained data transfers with bursts as high as 800 Mbps. Actual transfer rates vary according to the channel load, as the HiPPI channel is shared; however, the bandwidth is sufficient such that throughput is typically limited by read/write speeds on the T3E and MSF. To facilitate the data movement, ERDC has developed a suite of utilities to enable users to store large files on the MSF and retrieve them as needed for computational jobs. The utilities are based on the SGI File Transfer Agent (FTA) software package, and provide fetch (msfget/msfmget), store (msfput/msfmput), status (msfstat), and file manipulation (msfchmod/msfls/msfrm/etc.) capabilities.

4. TWO I/O STRATEGIES

The ABL turbulence solutions require massive computational resources and long-term archival storage. Tens of Terrabytes of numerical data will be generated by the project by the end of this year. Runs currently being conducted employ 500 PE's and generate 410 GB of data during one 20-hour job. For optimal performance, I/O between T3E memory and the T3E data disks must be considered carefully. A poor ABL I/O strategy can ruin an otherwise scalable algorithm, especially as large numbers of processors are employed and massive amounts of data are generated. Two approaches to massively parallel disk I/O were considered:

1. have each processor perform I/O to its own set of files, or
2. stripe one large data file across multiple data disks and instruct participating PE's to write to their portion of the global file.

Physically, the two solutions are the same, with separate processors writing data to separate disks. The main difference involves specifics of the implementations and the manner in which file names are handled.

To implement solution (2), the user can stripe a file across multiple disks using the *-p* and *-q* options of the Cray *assign* command. The *-p* option indicates which partitions to use in the filesystem. The *-q* option indicates how much data will be placed on a single disk before moving on to the next disk. Users must be careful to preserve the intended file layout and use the Cray *fdcp* command when copying files from another machine. It is easy to see that parallel I/O via file striping immediately involves machine-specific commands, and portability can become an issue.

Solution (1) presents a simpler alternative. Assurance that a processor's output files reside on its disk is accomplished trivially by instructing each processor to create its own files. Though restart data from a previous run will not be locally resident if file pre-fetching is done by another processor, presumably restart data is read only once, and poor I/O performance for the initial read is tolerable. To insure that subsequent writes to the restart files are not hindered by the files' initial pre-fetched locations, the user should make sure that a new set of restart files is created for each run and that each processor creates its own restart file.

5. ABL'S I/O IMPLEMENTATION

We implemented solution (1) with direct-access, fixed-record-length binary files for the ABL turbulence simulations. Output files are opened immediately before writing, and then closed again after writing is complete. This approach minimizes the number of simultaneously open files, and avoids problems with unflushed file buffers when a run is unexpectedly stopped.

5.1. Write-behind data buffering and HDS

Despite the promised performance, the scalability of the T3E code suffered from the massive amounts of I/O demanded by the project. Two improvements restored the code's scaling with the number of processors: 1) write-behind data buffering,[2] and 2) Hierarchical Data Structuring (or HDS). The first improvement involves buffering output-data writes so that computation and I/O can be overlapped for as long as the array being written is not needed for subsequent computation. On the T3E this is accomplished with an option to the Cray *assign* command: *-F bufa:num1:num2*, where *num2* is the number of buffers used, and *num1* is the size in 4096-byte blocks of each buffer layer. We use *-F bufa:2000:4*. See *man intro_ffio* on the T3E for more information on this and other *assign* options. The second performance improvement, HDS, is a technique we recently discovered. It involves dividing and subdividing the output data directory into multiple subdirectories and sub-subdirectories, ending with tails that contain no more than NCPU files (NCPU is the number of processors used for a given calculation). For example, for 3D volume files written at timelevels 00, 01, etc., instead of a directory named "data" with the following layout:

[2]We learned about this technique from David Cole and Mike Patterson at the Naval Oceanographic Office (NAVO) MSRC. Our specific implementation grew out of valuable correspondance with another DoD user, Alan Wallcraft, who conducts global and basin-scale ocean modeling on the NAVO T3E and the new ERDC IBM SMP.

data/: file1.00.0000, file1.00.0001, ... file1.00.0499,
file2.00.0000, file2.00.0001, ... file2.00.0499, ...
file1.01.0000, file1.01.0001, ... file1.01.0499,
file2.01.0000, file2.01.0001, ... file2.01.0499, ...

we now use the new HDS layout:

data/00/file1/: file1.00.0000, file1.00.0001, ... file1.00.0499
data/00/file2/: file2.00.0000, file2.00.0001, ... file2.00.0499
data/01/file1/: file1.01.0000, file1.01.0001, ... file1.01.0499
data/01/file2/: file2.01.0000, file2.01.0001, ... file2.01.0499, ...

The finer resolution via a directory tree insures that the application program does not spend time wading through an overwhelming number of filenames. For example, when using NCPU=500, ABL simulations generate in excess of 72,000 individual files during the course of a 20-hour run. When real-time data migration to the MSF is accomplished according to plan, all 72,000 files do not reside on the T3E /tmp disks at the same time. They are collected into tar files and moved off-line immediately after they are closed. Nevertheless, a minimum of 9,000 files simultaneously reside on the T3E /tmp space during the run, and this is sufficient to impact performance with the old data layout. For instance, when running with the old directory structure, ABL turbulence simulations required 82 minutes to complete 50 timesteps and do the associated disk I/O. When data from previous runs also occupied the directory, as much as 2 hours or more could be required. In fact, when performing tests to diagnose the source of the variable execution time, we intentionly allowed the data directory to fill with files that were not transfered when previous tests failed. Eventually the directory became so full with O(100,000) files that 50 timesteps were not completed in as many as 6 hours. Often an error "_sma_deadlock_wait" was encountered on the T3E. In contrast, now that we have implemented our new HDS strategy, execution time no longer varies. 50-timestep tests finish in 59 minutes ±1 minute, and 20-hour dedicated runs demonstrate the same predictable performance regardless of the number of files in the directory tree.

6. HDS PERFORMANCE

Figure 1 shows a plot of the ABL turbulence-code performance as a function of NCPU. The wallclock time t_c required per processor-gridpoint ($\mathrm{N_g}$/NCPU) per timestep is reported. $\mathrm{N_g} = \mathrm{NX} \times \mathrm{NY} \times \mathrm{NZ}$ is the total number of gridpoints. The per-processor problem size ($\mathrm{N_g}$/NCPU) is held fixed as NCPU is varied. Deviation from exactly linear scaling on NCPU (i.e., t_c = constant) results from A) the nonlinear dependence of the FFT cost on NX, NY and NZ and B) the differing cache utilization as the problem shape is varied to keep $\mathrm{N_g}$/NCPU fixed. To illustrate (A), consider that radix-2, 3, 4 and 5 Fast Fourier Transforms (FFTs) are used for the spectral calculations. The cost of an FFT of size M is proportional to $\mathrm{M}\Pi_\mathrm{i}\mathrm{N_i}$ for $\mathrm{i} = [2, 3, 4, 5]$, where Π_i indicates multiplication by the various i factors, and $\mathrm{N_i}$ is a radix-exponent of the ith factor of M; i.e., $\mathrm{M} = \Pi_\mathrm{i}\,\mathrm{i}^{\mathrm{N_i}}$. For $\mathrm{M} = 2000 = 4^2 \times 5^3$, the exponents are $\mathrm{N_2} = 0$, $\mathrm{N_3} = 0$, $\mathrm{N_4} = 2$ and $\mathrm{N_5} = 3$. The reduction of t_c as NCPU $\rightarrow 0$ results from the dependence of the FFT cost on $\mathrm{N_i}$ and the smaller FFT sizes used to fix $\mathrm{N_g}$/NCPU. The second effect, (B), is due to the relatively better

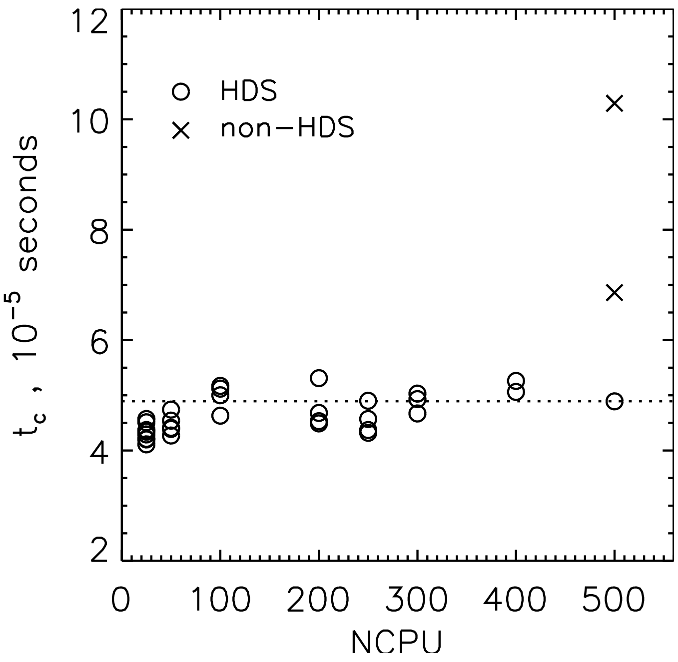

Figure 1. Wallclock time t_c per processor-gridpoint (N_g/NCPU) per timestep for ABL spectral turbulence code. N_g/NCPU is held fixed for all tests. Exact linear scaling on NCPU would result in t_c = constant. Circles indicate wallclock time when HDS is used to conduct I/O. Crosses indicate the best-case and typical times required when HDS is not used. They are 1.4 and 2.1 times slower than the same run with HDS. The difference between the two non-HDS timings result from larger numbers of files in the data directory for the slower case. Tests with even more files in the data directory did not finish in six hours (i.e., $t_c > 31 \times 10^{-5}$). The dotted line shows the time for the 500-PE HDS case.

cache reuse exhibited by the higher radix routines, but also the poorer performance of the communication-based transpose when array shapes deviate greatly from NX = NY = NZ. For example, in changing the number of spectral modes from $1000 \times 360 \times 2000$ on 500 PE's to $1000 \times 360 \times 1000$ on 250 PE's, the FFT algorithm used for the last index changes from two nested radix-4 calls to a radix-4 and a radix-2 call. Because the radix-4 routine has better cache reuse, we expect better FFT peformance for the 500-PE test; however, this gain is offset by the slower data-reordering portion of the transpose routine for the 500-PE run. In an attempt to compensate for these effects, multiple tests are reported in Figure 1 with different problem shapes. For the case with NCPU=500, we also include execution times without HDS (indicated by crosses). Table 1 presents all of the tests plotted in Figure 1 along with their execution times.

Though the execution times quoted here and reported in Figure 1 involve a specific code on the ERDC T3E, we expect the performance enhancement exhibited by HDS will be universally applicable to other architectures. Given its simplicity, we believe it should be implemented for all MPP applications that generate large amounts of data and large numbers of individual files. An additional benefit of HDS is the user can easily navigate the directory tree and inspect data as it is being generated by the application program. This is not feasible with a non-HDS directory containing O(10,000) or more files.

Joe Werne is a research scientist at the CoRA division of NWRA. He has 11 years of experience in high-performance computing on shared-memory vector and distributed-memory parallel platforms. He has written highly optimized Fortran programs, including the ABL turbulence code, for both stably and unstably stratified turbulence simulations as part of NSF and DoD Grand Challenge applications since 1993. Paul Adams recently moved from Applications Support Analyst to User Services Manager at the ERDC MSRC. He has 12 years of experience in the computer industry, providing programming and user support on Cray, SGI, and IBM platforms. David Sanders is an Applications Support Analyst at the ERDC MSRC and a Systems Support Engineer with Silicon Graphics, Inc. He has 16 years of programming and user-support experience on shared-memory, distributed-memory parallel, and distributed-shared-memory platforms. He wrote the suite of MSF data-handling utilities described above. David joins Cray, Inc. in April 2000.

REFERENCES

1. D.E. Gibson-Wilde, J. Werne, D.C. Fritts and R.J. Hill, Radio Sci. (2000) in press.
2. R.J. Hill, D.E. Gibson-Wilde, J. Werne and D.C. Fritts, Earth Planets Space 51 (1999) 499.
3. R. Strelitz and J. Werne, CEWES MSRC Journal (Spring 1999) 4.
4. J. Werne and D.C. Fritts, Geophys. Res. Lett. 26 (1999) 439.
5. J. Werne and D.C. Fritts, Phy. Chem. Earth (1999) in press.
6. "Interview with Joseph Werne", ERDC Resource Newsletter (March 2000).

Table 1
Timing tests for ABL turbulence code.

NCPU	NX × NY × NZ	$NX_i \times NY_i \times NZ_i$	*time*
25	300 × 200 × 600	(0112) × (1012) × (1112)	2962
25	400 × 180 × 500	(0022) × (0211) × (0013)	3023
25	500 × 144 × 500	(0013) × (0220) × (0013)	3031
25	500 × 72 × 1000	(0013) × (1210) × (1013)	3086
25	400 × 100 × 900	(0022) × (0012) × (0212)	3127
25	600 × 100 × 600	(1112) × (0012) × (1112)	3146
25	600 × 50 × 1200	(1112) × (1002) × (0122)	3247
25	400 × 90 × 1000	(0022) × (1201) × (1013)	3293
50	400 × 200 × 900	(0022) × (1012) × (0212)	3077
50	400 × 180 × 1000	(0022) × (0211) × (1013)	3160
50	450 × 200 × 800	(1202) × (1012) × (1022)	3175
50	500 × 144 × 1000	(0013) × (0220) × (1013)	3267
50	600 × 100 × 1200	(1112) × (0012) × (0122)	3414
100	600 × 200 × 1200	(1112) × (1012) × (0122)	3335
100	800 × 100 × 1800	(1022) × (0012) × (1212)	3598
100	1000 × 72 × 2000	(1013) × (1210) × (0023)	3686
100	500 × 288 × 1000	(0013) × (1220) × (1013)	3724
200	800 × 300 × 1200	(1022) × (0112) × (0122)	3233
200	600 × 300 × 1600	(1112) × (0112) × (0032)	3265
200	600 × 400 × 1200	(1112) × (0022) × (0122)	3370
200	1000 × 144 × 2000	(1013) × (0220) × (0023)	3820
250	500 × 720 × 1000	(0013) × (0221) × (1013)	3113
250	1000 × 360 × 1000	(1013) × (1211) × (1013)	3147
250	500 × 360 × 2000	(0013) × (1211) × (0023)	3291
250	1000 × 180 × 2000	(1013) × (0211) × (0023)	3526
300	1200 × 300 × 1200	(0122) × (0112) × (0122)	3364
300	1200 × 200 × 1800	(0122) × (1012) × (1212)	3549
300	900 × 400 × 1200	(0212) × (0022) × (0122)	3619
400	1200 × 200 × 2400	(0122) × (1012) × (1122)	3641
400	800 × 450 × 1600	(1022) × (1202) × (0032)	3784
500	1000 × 360 × 2000	(1013) × (1211) × (0023)	3524
500	1000 × 360 × 2000	(1013) × (1211) × (0023)	4940
500	1000 × 360 × 2000	(1013) × (1211) × (0023)	7410
500	1000 × 360 × 2000	(1013) × (1211) × (0023)	****

NCPU is the number of processors used. $N_g = NX \times NY \times NZ$ is the number of spectral modes. $N_g/NCPU$ is held constant for all tests. $NX_i \times NY_i \times NZ_i$ lists the radix exponents for $NX \times NY \times NZ$ in the order $2^i 3^j 4^k 5^l$. For example, $300 = 2^0 3^1 4^1 5^2$ is denoted (0112). The wallclock time in seconds is indicated in the last column. The final three entries are for non-HDS cases, the last of which failed to finish in 6 hours. See text for discussion.