Performance Tuning of a CFD Code on the Earth Simulator

By Ken’ichi ITAKURA,* Atsuya UNO,* Mitsuo YOKOKAWA,† Minoru SAITO,‡ Takashi ISHIHARA§ and Yukio KANEDA§

ABSTRACT High-resolution direct numerical simulations (DNSs) of incompressible turbulence with numbers of grid points up to $2048^3$ have been executed on the Earth Simulator (ES). The DNSs are based on the Fourier spectral method, so that the equation for mass conservation is accurately solved. In DNSs based on the spectral method, most of the computation time is consumed in calculating the three-dimensional (3D) Fast Fourier Transform (FFT). In this paper, we tuned the 3D-FFT algorithm for the Earth Simulator and have achieved DNS at 16.4TFLOPS on $2048^3$ grid points.

KEYWORDS Earth Simulator, Performance Tuning, CFD (Computational Fluid Dynamics), DNS (Direct Numerical Simulation), FFT (Fast Fourier Transform)

1. INTRODUCTION

Direct numerical simulation (DNS) of turbulence provides us with detailed data on turbulence that is free from experimental uncertainty. DNS is therefore not only a powerful means of finding directly applicable solutions to problems in practical application areas that involve turbulent phenomena, but also of advancing our understanding of turbulence itself — the last outstanding unsolved problem of classical physics, and a phenomenon that is seen in many areas which have societal impacts.

The Earth Simulator (ES)[1-3] provides a unique opportunity in these respects. On the ES, we have recently achieved DNS of incompressible turbulence under periodic boundary conditions by a spectral method on $2048^3$ grid points. Being based on the spectral method, our DNS accurately satisfies the law of mass conservation; this cannot be achieved by a conventional DNS based on a finite-difference scheme. Such accuracy is of crucial importance in the study of turbulence, and in particular for the resolution of small eddies.

The computational speed for DNSs was measured by using up to 512 processor nodes (PNs) of the ES to simulate runs with different numbers of grid points. The best sustained performance of 16.4TFLOPS was achieved in a DNS on $2048^3$ grid points.

2. SPECTRAL METHODS FOR DNS

In DNS by a spectral method[4], most of the CPU time is consumed in the evaluation of the nonlinear terms in the Navier-Stokes (NS) equations, which are expressed as convolutional sums in the wave vector space. As is well known, the sum can be efficiently evaluated by using an FFT. In order to achieve high levels of efficiency in terms of computational speed and numbers of modes retained in the calculation, we use the so-called phase-shift method, in which a convolutional sum is obtained for a shifted grid system as well as for the original grid system. This method allows us to keep all of the wave vector modes which satisfy $k < K_c = \sqrt{2} N/3$, where $N$ is the number of discretized grid points in each direction of the Cartesian coordinates, thus greatly increasing the number of retained modes. The nonlinear term can be evaluated without aliasing error by 18 real 3D-FFTs[4].

Our code, based on the standard 4-th order Runge-Kutta method for advancing time, was written in Fortran 90, and $25N^3$ main dependent variables are used in the program. The total number of lines of code, excluding comment lines, is about 3,000. The required amount of memory for $N = 2048$ and double-precision data was thus estimated to be about 1.56TB.

3. PARALLELIZATION

Since the 3D-FFT accounts for more than 90% of the computational cost of executing the code for a DNS of turbulence by the Fourier spectral method, the most crucial factor in maintaining high levels of performance in the DNS of turbulence is the efficient execution of this calculation. In particular, in a
parallel implementation, the 3D-FFT has a global data dependence because of its requirement for global summation over PNs, so data is frequently moved among the PNs during this computation.

Vector processing is capable of efficiently handling the 3D-FFT as decomposed along any of the three axes. Applying the domain decomposition method to assign the calculations for respective sets of several 2D planes to individual PNs, and then having each PN execute the 2D-FFT for its assigned slab by vector processing and automatic parallelization is an effective approach. The FFT in the remaining, direction should then be performed after transposition of the 3D array data. Domain decomposition in the k3 direction in wave-vector space and in the y direction in physical space was implemented in the code.

We achieved a high-performance FFT by implementing the following ideas/methods on the ES.

3.1 Data Allocation
Let \( n_d \) be the number of PNs, and let us consider the 3D-FFT for \( N \) real-valued data, which we will call \( u \). In wave-vector space, the Fourier transform \( \hat{u} \) of the real \( u \) is divided into a real part \( \hat{u}_R \) and an imaginary part \( \hat{u}_I \), each of which is of size \( N^3/2 \). These data are divided into \( n_d \) data sets, each of which is allocated to the global memory region (GMR) of a PN, where the size of each data set is \((N-1) \times N \times (N^2/n_d)\). Similarly, the real data of \( u \) are divided into \( n_d \) data sets of size \((N-1) \times (Nn_d) \times N\), each of which is allocated to the GMR of the corresponding PN. Here the symbol \( n_i \) in \( n_1 \times n_2 \times n_3 \) denotes the data length along the \( i \)-th axis, and we set the length along the first axis to \((N-1)\), so as to speed up the memory throughput by avoiding memory-bank conflict.

3.2 Automatic Parallelization
For efficiently performing the \( N \times (N/2/n_d) \) 1D-FFTs of length \( N \) along the first axis, the data along the second axis are divided up equally among the 8 APs of the given PN. This division can be efficiently achieved by using the automatic parallelization provided by the ES. However, we decided to achieve this in practice by manual insertion of the some parallelization directives before target do-loops, which directs the compiler to apply automatic parallelization. We did this because we had found that the use of automatic parallelization by the compiler did not make use of the benefits of parallel execution.

3.3 Radix-4 FFT
Though the peak performance of an AP of the ES is 8GFLOPS, the bandwidth between an AP and the memory system is 32GB/s. This means that only one double-precision floating-point datum can be supplied for every two possible double-precision floating-point operations. The ratio of the number of times memory is accessed to the number of floating-point data operations for the radix-2 FFT is 1; the memory system is thus incapable of supplying sufficient data to the processors.

This bottleneck of memory access in the kernel loop of a radix-2 FFT function degrades the sustained levels of performance in the overall task. Thus, to obtain calculation of the 1D-FFT within the ES efficiently, the radix-4 FFT must replace the radix-2 FFT to the extent that this is possible. This is because of the lower ratio of the number of memory accesses to the number of floating-point data operations in the kernel loop of the radix-4 FFT, so the radix-4 FFT better fits the ES.

3.4 Remote Memory Access
Before performing the 1D-FFT along the 3rd axis, we need to transpose the data from the domain decomposition along the 2nd axis to the one along the 3rd axis. The remote memory access (RMA) function is capable of handling the transfer of data which is required for this transposition. RMA is a means for the direct transfer of data from the GMR of a PN of the ES to the GMR of pre-assigned PN. It is then not necessary to make copies of the data, i.e., data are copied neither to the MPI-library nor to the communications buffer region of the OS. In a single cycle of RMA transfer, \( N \times (Nn_d) \times (N^2/n_d) \) data are transferred from each of the \( n_d \) PNs to the other PNs. The data transposition can be completed with \((n_d - 1)\) such RMA-transfer operations, after which \( N \times (Nn_d) \times (N^2) \) data will have been stored at each target PN.

The 1D-FFT for the 3rd axis is then executed by dividing the do-loop along the 1st axis so as to apply automatic parallel processing.

4. PERFORMANCE OF PARALLEL COMPUTATION
We have measured the sustained performance for both the double-precision and single-precision versions by changing the number \( N \) of grid points, setting \( N \) values of 128, 256, 512, 1024, and 2048. The corresponding numbers of PNs taken up in the ES are 64, 128, 256, and 512; see Table I for the correspondences between number of PNs and number of grid points which we tested.

The calculation time for 100 time steps of the
Runge-Kutta integration was measured by using the MPI function MPI_wtime. The times taken in initialization and I/O processing are excluded from the measurement because these values are negligible in comparison with the cost of Runge-Kutta integration, which increases with the number of time steps.

The number of floating-point operations in the measurement range, which is needed in calculation of the sustained performance, is monitored by an analytical method for obtaining the number of operations. This is based on the fact that the number of operations in a 1D-FFT with both radix-2 and 4 against the number of grid points \( N \) is \( N(5p + 8.5q) \), where \( N \) is represented by \( 2^q 4^p \) and \( p = 0 \) or 1 according to the value of \( N \). Considering that the DNS code has 72 real 3D-FFTs per time step and summing up the number of operations over the measurement range by hand, we obtain the following analytical expressions of the number of operations:

\[
459N^p \log_2 N + (288 + 16\pi N)^q, \text{ if } p=0, \quad (1)
459N^q \log_2 N + (369 + 16\pi N)^p, \text{ if } p=1. \quad (2)
\]

The results of these expressions can be used as reference values on the number of operations in comparing the sustained performance of code.

Table I shows the effective performance results. \( N^p \) and \( n_d \) in the table indicate the numbers of grid points and PNs used in each run, respectively. The number \( n_p \) of APs in each PN is fixed to 8. In the measurement runs, the data allocated to the global memory region of each PN are transferred by MPI_put, and the tasks in each PN are divided up among the APs. The maximum sustained performance of 16.4 TFLOPS was for the problem size of 2048 with 512 PNs and the data are single precision format.

Table I shows that the sustained performance is better for single precision than for double precision code for larger values of \( N \) and/or smaller values of \( n_d \). Although the numbers of floating-point operations in the code are the same regardless of the precision, only half as much data is transferred in the transposition for 3D-FFT in the former case.

Table II shows the ratio of data transfer time to whole time by changing the number \( N^q \) of grid points and the number \( n_d \) of PNs. The ratio becomes larger as the number of \( n_d \) becomes larger in the case where \( N \) is constant. When \( n_d \) is constant, the ratio also becomes smaller as \( N \) becomes larger. The MPI_put function shows good performance for the larger size of data, and the latency should account for the small amount of data transfer time on the ES. On the other hand, the latency of MPI_put function dominates data transfer time when the message size is small. Therefore, the smaller the message size residing in each PN becomes, the larger the ratio of data transfer time becomes compared to the computation time.

Figure 1 shows the computation time of one-time step of the Runge-Kutta integration in double precision for 1024\(^3\) grid points by divided into the data transfer time and others, when the number of APs in PN is changed as 1, 2, 4, and 8. The computation time is reduced almost linearly with respect to the number of APs and the number of PNs, because the ratio of data transfer time to the computation time is not very crucial for these cases.

Figure 2 shows the computation time for three versions of Trans7 as the number of PNs and the precision of floating point data are changed. The difference of the versions is a 3D-FFT implementation; one uses a radix-2 FFT with transposition on global memory area (Case 1), the other uses a radix-4 FFT with transposition on local memory area (Case 2), and the rest use a radix-4 FFT with transposition on global memory area (Case 3). Case 3 shows the best performance of the three. Although the use of global memory area reduces the computation time, it is found that the implementation by using radix-4 FFT is much more effective in reducing the time. It should be mentioned that users pay attention to the ratio of the number of times memory is accessed to the number of floating point data operations in kernel loops in developing the program for the ES. It is also clear that the data transfer time for single precision is smaller than that for double precision, as stated.
5. SUMMARY

For the DNS of turbulence, the Fourier spectral method has the advantage of accuracy, particularly in terms of solving the Poisson equation, which represents mass conservation and needs to be solved accurately for a good resolution of small eddies. However, the method requires frequent execution of the 3D-FFT, the computation of which requires global data transfer. In order to achieve high performance in the DNS of turbulence on the basis of the spectral method, efficient execution of the 3D-FFT is therefore of crucial importance.

By implementing new methods for the 3D-FFT on the ES, we have accomplished high-performance DNS of turbulence on the basis of the Fourier spectral method. This is probably the world’s first DNS of incompressible turbulence with $2048^3$ grid points. The DNS provides valuable data for the study of the universal features of turbulence at large Reynolds number. The numbers of degrees of freedom $4 \times N^3$ (4 is the number of degrees of freedom $(u_1, u_2, u_3, p)$ in terms of grid points, and $N^3$ is the number of grid points) are about $3.4 \times 10^{10}$ in the DNS with $N = 2048$. 
The sustained speed is 16.4TFLOPS. To the authors' knowledge, these values are the highest in any simulation so far carried out on the basis of spectral methods in any field of science and technology.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Tetsuya Sato, director-general of the Earth Simulator Center, for his warm encouragement of this study.

REFERENCES


Takashi ISHIHARA received his B.Sc. degree in physics and M.E. degree in applied physics from Nagoya University in 1989 and 1991, respectively. He received his D.E degree in applied physics from Nagoya University in 1994. He was a research associate at the Department of Mathematics in Toyama University from 1994 to 1997. He has been an assistant professor at the Graduate School of Engineering in Nagoya University since 1997.

Dr. Ishihara is a member of the Physical Society of Japan and of the Japan Society of Fluid Mechanics.

Yukio KANEDA received his B.Sc. and M.Sc. degrees in physics from the University of Tokyo in 1971 and 1973, respectively. He received his D.Sc. degree in physics, also from the University of Tokyo, in 1976. He has worked at the Faculty of Science as a research associate, and at the Faculty of Engineering, Nagoya University as a research associate, an associate professor and a professor. He has also worked at the Graduate School of Poly-Mathematics, Nagoya University as a professor, and is currently a professor in the Department of Computational Science and Engineering, at the Graduate School of Engineering, Nagoya University. His main interest is in fluid mechanics, including computational fluid mechanics and turbulence.

Dr. Kaneda is a member of the Physical Society of Japan, the Japan Society of Fluid Mechanics, and the Japan Society for Industrial and Applied Mathematics.