# Performance evaluation of parallel direct numerical simulation code on supercomputer SX-Aurora TSUBASA

Yokokawa, Mitsuo
Takenaka, Yujiro
Ishihara, Takashi
Komatsu, Kazuhiko
Kobayashi, Hiroaki

(Citation)
Computers & Fluids,261:105913

(Issue Date)
2023-07-15

(Resource Type)
journal article

(Version)
Accepted Manuscript



(URL)
https://hdl.handle.net/20.500.14094/0100482625

# Performance evaluation of a parallel DNS code on the supercomputer SX-Aurora TSUBASA

Mitsuo YOKOKAWA[a,*], Yujiro TAKENAKA[a], Takashi ISHIHARA[b], Kazuhiko KOMATSU[c], Hiroaki KOBAYASHI[d]

[a] *Graduate School of System Informatics, Kobe University, Kobe, Hyogo 657-8501, Japan*
[b] *Graduate School of Environmental and Life Science, Okayama University, Okayama 700-8530, Japan*
[c] *Cybersicence Center, Tohoku University, Sendai, Miyagi 980-8579, Japan*
[d] *Graduate School of Information Sciences, Tohoku University, Sendai, Miyagi 980-8579, Japan*

**Abstract**

Direct numerical simulations (DNSs) of incompressible turbulence have been performed since the late 1960s, but simulations that reproduce strongly nonlinear turbulent flows as in the real-world have not been realized.

A DNS code was parallelized by the pencil decomposition and optimized for the vector-type supercomputer SX-Aurora TSUBASA. Applying a loop blocking technique to optimize the code, it is found that some loops of three-dimensional fast Fourier Transforms have the optimal loop blocking size with which the DNS code exhibits good performance. 3.3-fold speedup is obtained by the optimized DNS code compared to the time for the original code. A computation time estimation model for large-scale DNS is also constructed based on the measured throughput data of all-to-all communication on SX-Aurora TSUBASA. A step of the Runge-Kutta time integration needs a minute for the DNS with $18,432^3$ grid points on 16,384 VEs of SX-Aurora TSUBASA.

*Keywords:* Direct numerical simulation, Turbulent flow, Spectral method, Multi-core vector processors, loop blocking

*Corresponding author
*Email address:* yokokawa@port.kobe-u.ac.jp (Mitsuo YOKOKAWA)

## 1. Introduction

It is extremely important to elucidate real fluid flows because those exists everywhere around us in scientific and engineering field. In particular, properties and statistics of turbulent flows has been explored for a long time. The Navier-Stokes equation is well-known to represent fluid flows, but its analytical solution are not clarified except for very simple flows. Direct numerical simulations (DNSs) of the Navier-Stokes equation are only solutions to tackle this difficult problem. Among these simulations, DNS for incompressible isotropic homogeneous turbulent flows is considered as a canonical problem and many studies on these flows have been performed since the late 1960s [1, 2].

With the recent remarkable development of supercomputers in terms of capacities and capabilities, large-scale DNSs with appropriate implementation to supercomputers having special architecture are possible to carry out. The DNS with $12,288^3$ grid points of which the Taylor scale Reynolds number is approximately 2,300 was performed on the K computer at RIKEN Center for Computational Sciences [3]. The DNS using GPUs with $18,432^3$ grid points on the Summit at Oak Ridge National Laboratory was reported [4]. However, those DNSs that reproduce strongly nonlinear turbulent flows as in the real-world have not been realized even with such cutting-edge supercomputer systems. Recently announced supercomputer SX-Aurora TSUBASA inheriting vector operation architectures has many attractive functions and is expected to make larger scale DNSs.

We have developed a parallel Fourier spectral DNS code applying a two-dimensional domain decomposition for twenty years, in which an original three-dimensional Fast Fourier Transforms (3D-FFT) library written in Fortran is incorporated so that the code can be modified freely to achieve high performance on various supercomputer systems.

In this study, the DNS code are optimized on the SX-Aurora TSUBASA. Firstly, a parallel DNS code implementation is presented, followed by optimization of the code and its performance evaluation. A computation time estimation

model for larger size DNS constructed from measurement results for data communication on the SX-Aurora TSUBASA is also given.

## 2. Basic Equations and Their Discretization

We consider homogeneous isotropic turbulent flows that obey the incompressible Navier-Stokes equation and the continuum equation

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = -\nabla p + \nu\nabla^2\boldsymbol{u} + \boldsymbol{f}, \quad \text{and} \tag{1}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{2}$$

in a periodic box of the side length $2\pi$. Here $\boldsymbol{u} = (u_1, u_2, u_3)$, $p$, $\nu$, and $\boldsymbol{f} = (f_1, f_2, f_3)$ denote velocity, pressure, kinematic viscosity, and external force, respectively (see [5] for details of the external force). Since the boundary condition is periodic, the Fourier spectral method can be applied for discretization.

Then, equations (1) are written as

$$\left(\frac{d}{dt} + \nu|\boldsymbol{k}|^2\right)\hat{u_i}(\boldsymbol{k}) = \left(\delta_{ij} - \frac{k_i k_j}{|\boldsymbol{k}|^2}\right)\hat{h_j}(\boldsymbol{k}) + \hat{f}_i(\boldsymbol{k}), \tag{3}$$

where equation (2) is used to eliminate the pressure term. In equation (3), $\boldsymbol{k} = (k_1, k_2, k_3)(-\frac{N}{2} \le k_i \le \frac{N}{2} - 1, i = 1, 2, 3)$ is the wavenumber vector, $N$ is the number of grid points in each of the Cartesian coordinates in the physical space, $\hat{h_j}(\boldsymbol{k}) = \sqrt{-1}\, k_l\, \widehat{u_j u_l}$ is the nonlinear term, and the hat symbol $\hat{}$ denotes the Fourier coefficients. The ordinary differential equations (ODEs) in terms of $\hat{u_i}(\boldsymbol{k})$ are integrated by the fourth-order Runge-Kutta-Gill (RKG) method.

In calculating the nonlinear term $\hat{h_j}(\boldsymbol{k})$ which is expressed as a convolution sum in the wave vector space, the fast Fourier transform (FFT) can be efficiently used to reduce computational time. Aliasing errors induced by the spectral transform method are removed by a phase shift method (Fig. 1) and a cutoff of the modes $\boldsymbol{k}$ satisfying $|\boldsymbol{k}| > (\sqrt{2}/3)N$.

### *2.1. Code Parallelization*

In DNS by the spectral method, most of the computation time is associated with evaluating the alias-free nonlinear terms, and the three-dimensional fast

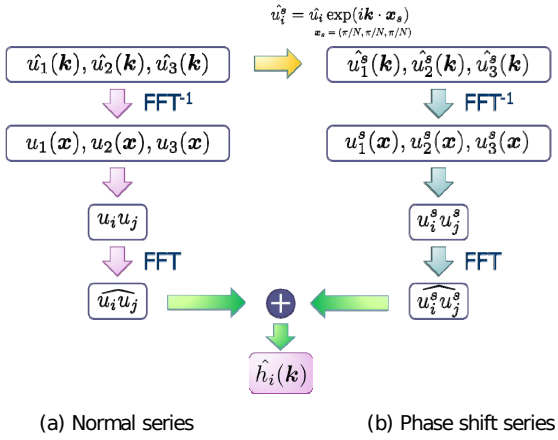


Figure 1: Phase shift method

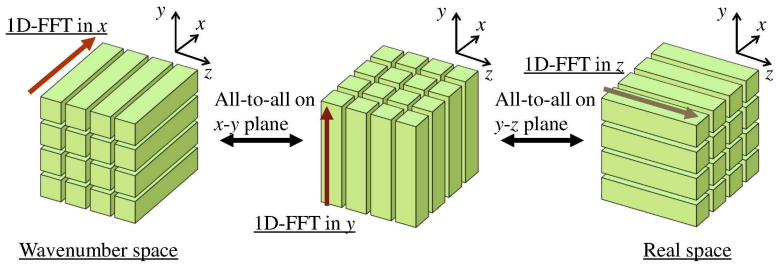


Figure 2: Schematic diagram of 3D-FFT procedure parallelized by the pencil decomposition.

Fourier Transforms (3D-FFTs) are used in the evaluation that require data transfer among all processes in the parallel computation.

We have developed a parallel DNS code in Fortran 90 using the Message Passing Interface (MPI) and OpenMP for parallel processing. Two-directional domain decomposition or pencil decomposition for MPI parallelization is applied for the data distribution.

Figure 2 shows a parallel 3D-FFT procedure used in the code. The complex valuable data in the wavenumber space are partitioned among MPI processes in $y$- and $z$-directions first. After executing the one-dimensional FFTs (1D-FFTs) in the $x$-direction, the domain decomposition changes to the $z$- and $x$-directions

by performing all-to-all communications among the $x$-$y$ plane processes. After executing the 1D-FFTs in the $y$-direction, the domain decomposition changes to the $x$- and $y$-directions by performing all-to-all communications among the $y$-$z$ plane processes in the same manner. Finally, the 1D-FFTs are executed for the data in the $z$-direction.

### *2.2. Data transposition in 3D-FFT*

Figure 3 describes a schematic calculation sequence of 3D-FFT in detail. After 1D-FFTs are applied to multiple $x$-directional data that are located in memory of a process, the data in the $y$- and $z$-directions are distributed over processes. Therefore, the data on a process should be rearranged so that the data in the $y$-direction are gathered in a process and 1D-FFTs are applied to the data in $y$-direction. Firstly, the data are re-ordered on memory in a process so that a bunch of data to be transferred to an appropriate destination process by an MPI function become contiguous on memory, then data transposition in $x$-$y$ slab among processes are carried out, and then the transferred data are re-ordered in a process so that data in $y$-direction to be contiguous on memory. The same operation should be taken in applying 1D-FFTs to data in the $z$-direction.

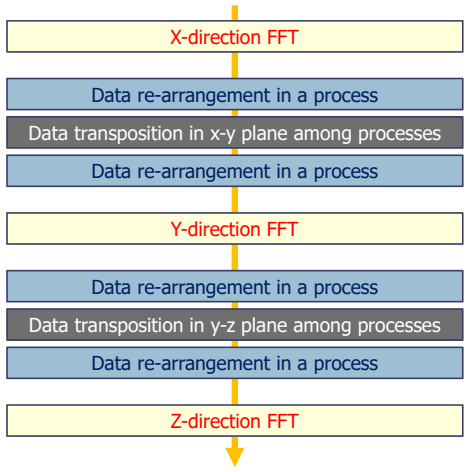


Figure 3: Flowchart of parallel 3D-FFT calculation

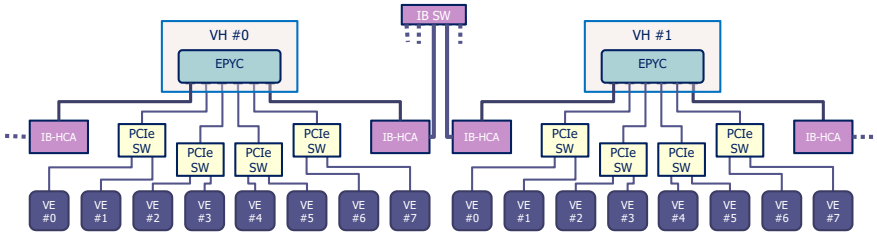


Figure 4: Configuration of SX-Aurora TSUBASA B401-8 model

In the implementation, we created two sub-communicators, `com1` and `com2`, each of which contains several $x$-$y$ or $y$-$z$ planes or slabs, using the MPI function `MPI_Comm_split` so that all-to-all communication can be confined in each slab in each sub-communicator. Note that an allocation of MPI processes to the compute nodes is quite important, because, in some cases, processes in the same communicator are assigned to distant nodes from the viewpoint of intra-network configuration and all-to-all data transposition time differs between two communicators.

## 3. Code Optimization to SX-Aurora TSUBASA

### *3.1. Overview of SX-Aurora TSUBASA*

The SX-Aurora TSUBASA is a vector-type supercomputer that has a distinguished system architecture compared to previous SX-series supercomputers[6, 7]. The base unit called the vector host island (VH island) consists primarily of a vector host (VH) and one or more vector engines (VEs). The VH is a standard server with one or two x86 architecture processors on which a standard Linux operating system (OS) is operated. A VE is implemented as a Peripheral Component Interconnect Express (PCIe) card, on which a vector processor is mounted with six high-bandwidth memory (HBM2) modules. The vector processor consists of eight vector cores, a 16 MB last-level-cache (LLC), and a VE direct memory access (DMA) engine. The LLC is connected to each core through a two-dimensional (2D) intra-network with a total cache bandwidth of

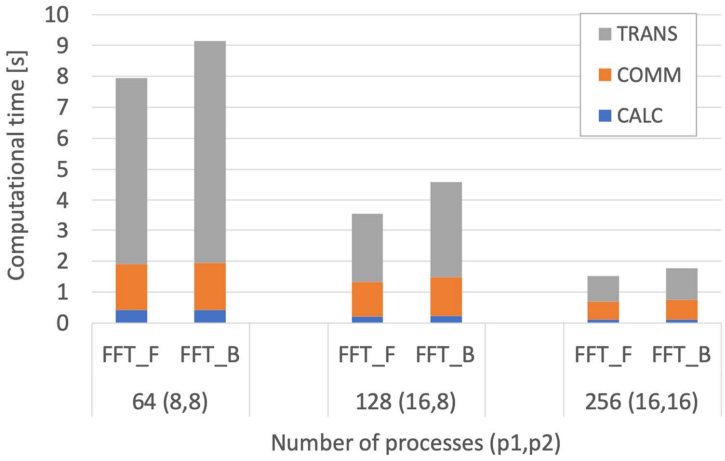


Figure 5: Calculation time of 3D-FFT

3.0 TB/s. The system consisting of multiple VH islands can be built by connecting them with the InfiniBand (IB) network via IB host channel adapters (IB-HCA) that are attached to the VH through PCIe Gen4 and has 12GB/s bandwidth. Legacy programs developed on the conventional vector system can be utilized as they are and can be carried out with high performance. In this study, SX-Aurora TSUBASA model B401-8 is taken as the VH island for performance evaluation into consideration (Fig. 4).

### *3.2. Performance of the original code*

Most of the calculation time of the spectral DNS code is consumed in the 3D-FFT to compute non-linear terms of the Navier-Stokes equation. It is crucial to reduce 3D-FFT computation time to make long time steps RKG integration so that flow fields reach stationary states statistically.

We measured the parallel 3D-FFT calculation time by dividing it into three parts that are so-called butterfly operations time (referred to as **CALC**), data re-arrangement time on memory in a process (referred to as **TRANS**), and all-to-all data transpose time among parallel processes (referred to as **COMM**) for $2048^3$ grid data. Computation time of 3D-FFT by changing the total number of processes of 64, 128, and 256 is shown in Fig. 5. Each color denotes the

Table 1: Averaged vector length and vectorization ratio of **CALC** and LLC hit ratio of **TRANS** for $n = 2048^3$

| Number of processes (p1, p2) | **CALC** | | **TRANS** |
|---|---|---|---|
| | Averaged vector length | Vectorization ratio (%) | LLC hit ratio (%) |
| 64 (8, 8) | 256 | 99.49 | 29.9 |
| 128 (16, 8) | 256 | 99.48 | 25.3 |
| 256 (16, 16) | 256 | 99.48 | 16.8 |

time for one of the three measurement parts. The numbers (p1, p2) in the bottom of the figure stands for the numbers of parallel processes in $y$- and $z$-directions, respectively, i.e. each process in the sub-communicator `com1` has the rank number from 0 to p1$-$1, and so on. `FFT-F` and `FFT-B` stands for the forward FFT and backward FFT, respectively. It is found that the computational time is decreased linearly as the number of processes increases, and the **TRANS** part is the most time-consuming part of the three.

Execution profiles for the different number of processes obtained by the performance analyzer "ftrace" are summarized in Table 1. Averaged vector length and vectorization ratio of **CALC** is 256 and approximately 99.5 %, respectively, in all cases and that appears to be well computation. It is found, however, that LLC hit ratio of **TRANS** is pretty low, i.e. less than 20 % for 256 processes. And the LLC hit ratio decreases as the number of processes increases and this is possible to cause the computation time of TRANS large.

### *3.3. Optimization of 3D-FFT*

Cache blocking technique is applied to the **TRANS** part to improve the LLC hit ratio. Figure 6 denotes the loop blocking scheme whose size is expressed by the variable `bl`. By this modification, LLC hit ratio is expected to improve, but in general the vector length becomes shorter and the performance degrades in some situation.

Computation time of the modified **TRANS** part was measured by changing the blocking size `bl` as 2, 4, 8, 16, 32, 64, 128, and 256. Figure 7 is the compu-

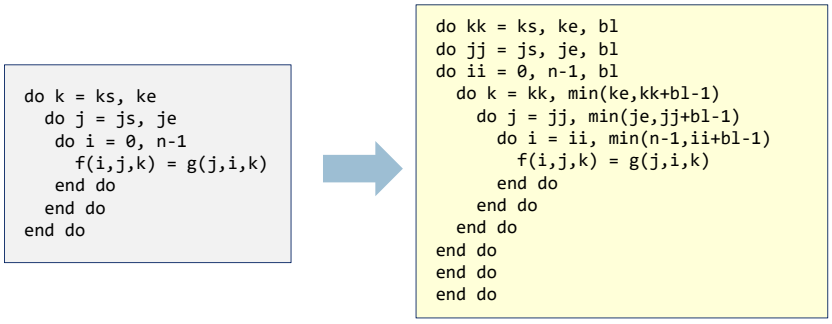


Figure 6: Code modification by loop blocking in **TRANS**

tation time of different eight portions named TRANS**n**[FB] of TRANS, where "F" and "B" stand for the forward FFT and the backward FFT, respectively, and **n** stands for the different part of data re-ordering part on memory corresponding to the blue shaded boxes in Fig. 3. Averaged vector length and LLC hit ratio according to the blocking size `bl` of the modified **TRANS** are shown in Table 2. The shortest computation time is obtained as `bl` is 16. LLC hit ratio is high for the smaller blocking sizes of 2, 4, and 8, but computation time is not shorter than that at `bl` of 16, because latency of vector operations with short averaged vector length affects. The optimal blocking size for each portion is different for eight different portions. It is found that the optimal blocking size for **TRANS1F** and **TRANS3B** is 16. As for the other potions, it is optimal for `bl` to be 256 rather than the LLC hit ratio is high.

The computation time of the revised 3D-FFT with the data size $2048^3$ is shown in Fig. 8. Calculation time of **TRANS** was 9.2 times faster than that of the original **TRANS** and 3.3 times faster in the 3D-FFT calculations. It is found that the loop blocking works well.

### *3.4. An Estimation Model of Computation Time*

It is important to carry out quite larger scale DNS of turbulent flows to make statistics of turbulence clear as long as a computer system can manage large DNS size in terms of resource capacity. Here, an estimation model of computation time of DNS with grid point more than $10,000^3$ is constructed

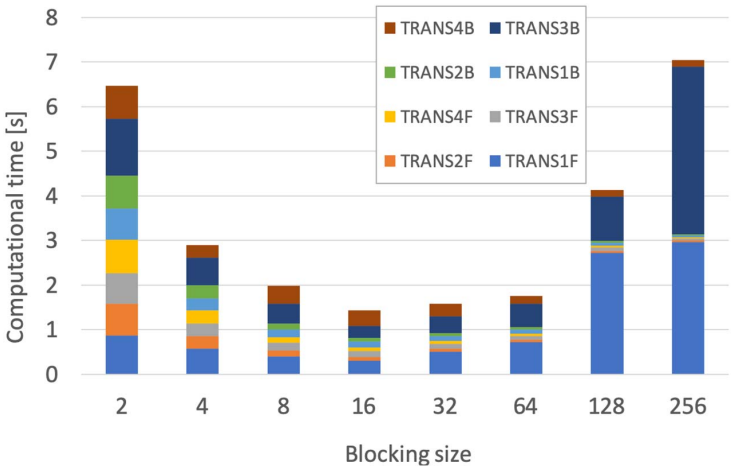


Figure 7: Calculation time of **TRANS** part. Eight different parts of **TRANS** are measured for the size of $2048^3$

Table 2: Averaged vector length and LLC hit ratio for $n = 2048^3$ for the modified TRANS

| blocking size $bl$ | Averaged vector length | LLC hit ratio (%) |
|---|---|---|
| 2 | 2.0 | 66.0 |
| 4 | 4.0 | 76.1 |
| 8 | 8.0 | 66.5 |
| 16 | 16.1 | 34.6 |
| 32 | 32.2 | 19.1 |
| 64 | 64.0 | 20.9 |
| 128 | 123.7 | 32.6 |
| 256 | 213.2 | 36.7 |

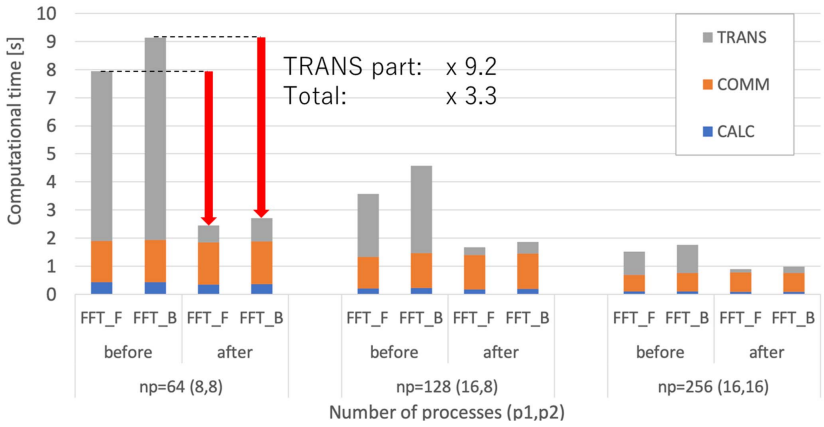


Figure 8: Improvement of calculation time of 3D-FFT for the size of $2048^3$

based on measured communication performance of MPI functions among the nodes of SX-Aurora TSUBASA.

There are four kinds of routes for MPI communication on SX-Aurora TSUBASA model B401-8 (Fig. 4); (a) MPI communication route by memory copy operations between processes launched on cores on a VE, (b) MPI communication route between two VEs that are connected to the same PCIe switch, .e.g. VE `#0` and VE `#1`, (c) MPI communication between two VEs that are connected to the different PCIe switches but those PCIe switches are connected to the same VH, .e.g. VE `#0` and VE `#2`, and (d) MPI communication between two VEs, each of which is under the different VH and data are passed through IB switches, .e.g. VE `#0` of VH `#0` and VE `#0` of the VH `#1`. These four routes of data communication have different latencies and throughputs, and note that parallel 3D-FFT computation time differs according to assignment of parallel processes to VEs.

Communication time for all-to-all communication functions on these routes is measured by changing message sizes from 8 bytes (B) to 64 mega bytes (MiB). Each MPI process is assigned to the core of SX-Aurora TSUBASA. Sustained data throughputs are shown in Fig. 9, where `np` stands for the number of MPI processes. The sustained throughputs gradually decreased as the number of

processes increases, and sustained throughputs in the case that $np$ is larger than 128 appears to saturate to approximately 0.5 GiB/s for more than 1 MiB message data.

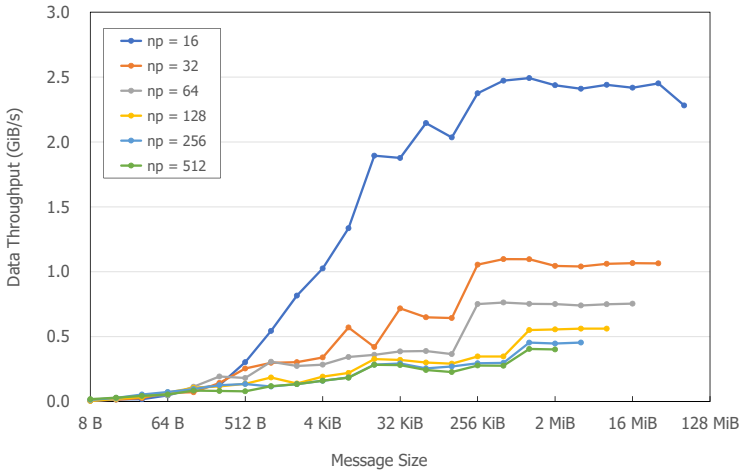


Figure 9: Data throughputs for all-to-all communications

As explained in section 2.2, all-to-all communications in $y$- and $z$- directions are carried out using all MPI processes, but those communications in each direction are grouped into several bunches by sub-communicators `com1` and `com2` and are restricted on some MPI processes that have the same sub-communicator. Distributions of MPI processes with the same communicator for $y$-direction (`com1`) and $z$-direction (`com2`) are different. For example, in the case that the total number of processes `np` = 256 is divided into 16 (p1) $\times$ 16 (p2) processes in $y$- and $z$-directions and the processes are assigned to all cores of four VH islands of SX-Aurora TSUBASA in order, 16 groups of adjacent two VEs, e.g. VEs `#0` and `#1`, each of which has 16 processes and is bundled with the same sub-communicator `com1`, and carry out all-to-all communications concurrently. On the other hand, the 16 MPI processes with the same value of a fomula

$$(\texttt{rank number}\ \textit{in}\ \texttt{MPI\_COMM\_WORLD}) \bmod 16 \tag{4}$$

are grouped into 16 bunches in the sub-communicator `com2` and execute all-to-all

185 communications in $z$-direction.

Sustained throughput $B_{\texttt{eff}}$ can be calculated by the equation

$$B_{\texttt{eff}} = \frac{m \times n_p}{T} \tag{5}$$

where $m$ message size, $n_p$ the number of MPI processes, and $T$ measured communication time. Sustained throughputs for two communicators are measured on the four VH islands and are drawn by solid line in Fig. 10. The throughput for the communicator `com1` depicted blue solid line is approximately the same
190 as the throughput of `np` = 16, because the communication are restricted in two adjacent VEs connected to the same PCIe switch and are not affected from other all-to-all communications at all. On the other hand, the throughput for the communicator `com2` that is drawn by red solid line is quite low compared to that for `com1` and appears to saturate to 0.45 GiB/s on more than 16 VEs
195 as the message size becomes larger.

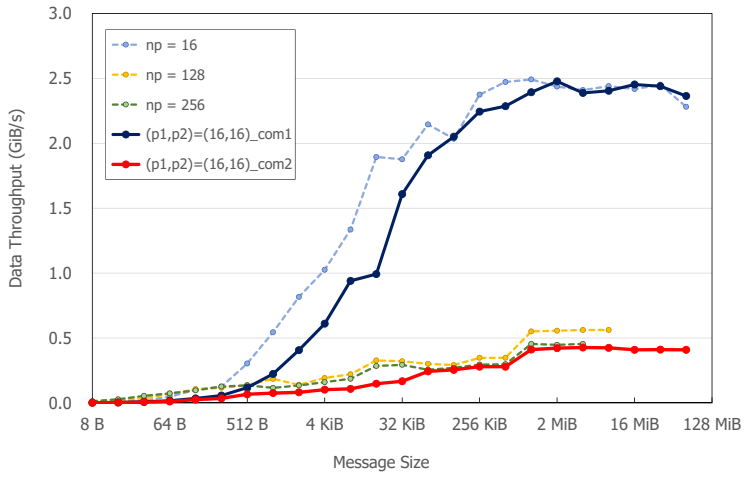


Figure 10: Data throughputs for all-to-all communications in different communicators

It is stated before that the communication time for 3D-FFT is dominant on large-scale supercomputers, because recent supercomputers are constructed as interconnect network performance is low compared to CPU performance. The

communication time for 3D-FFT is estimated by the equations

$$t_{\texttt{comm}}\ [\text{sec.}] = \frac{8 \times n^3}{B_{\texttt{eff}} \cdot \text{p1} \cdot \text{p2}} \tag{6}$$

where $n$ is the number of grid points in each Cartesian direction, p1 and p2 are the number of processes in two-axes parallelization, and $B_{\texttt{eff}}$ is the sustained bandwidth calculated by the equation 5. If the DNS with $18,432^3$ grid points is carried out with 131,072 processes or (p1,p2)=(512, 256) on 16,384 VEs of SX-Aurora TSUBASA, the size of which is estimated in terms of memory capacity and throughputs are kept for this large configuration, $t_{\texttt{comm}}$ is estimated as 0.76 seconds for one 3D-FFT calculation and time for an Runge-Kutta integration step is approximately 1 minute.

## 4. Concluding Remarks

It is quite important to elucidate properly statistics of turbulent flows based on the famous Navier-Stokes equation that has been a great concern for a long time and still be the unsolved problem in physics. It is considered that direct numerical simulations (DNS) of turbulent flows using cutting-edge supercomputers is only a solution to solve the problem. Authors have been developed DNS codes optimized for the newly developed supercomputers at that time for over twenty years.

In this paper, the DNS code was parallelized by the pencil decomposition and optimized for the vector-type supercomputer SX-Aurora TSUBASA. Applying a loop blocking technique to optimized the code, it is found that some part of 3D-FFT have the optimal loop blocking size is determined to reduce the computational time by giving up long vector length, with which the system exhibits good performance. 3.3-fold speedup is obtained by the modified 3D-FFT code compared to the time for the original code.

A computation time estimation model for large-scale DNS was also constructed by using the measured throughput data of all-to-all communication on SX-Aurora TSUBASA. A step of the RKG time integration needs a minute for

the DNS with $18,432^3$ grid points, that is, to our best knowledge, the largest grid points so far, on 16,384 VEs of SX-Aurora TSUBASA. However, a configuration of such numbers of VEs is extremely difficult to construct. We can, of course, use the largest supercomputer FUGAKU in japan, but even using the system, the DNS with $18,432^3$ grid points is difficult to execute. Therefore, a DNS code with reduced memory capacity should be developed for the future.

## Acknowledgments

This work is supported partially by MEXT Next Generation High-Performance Computing Infrastructure and Applications R&D Program, entitled R&D of a Quantum-Annealing-Assisted Next Generation HPC Infrastructure and Its Applications.

## References

[1] S. A. Orszag, Numerical methods for the simulation of turbulence, The Physics of Fluids 12 (12) (1969) II–250–II–257. doi:10.1063/1.1692445.

[2] M. Yokokawa, K. Itakura, A. Uno, T. Ishihara, Y. Kaneda, 16.4-Tflops direct numerical simulation of turbulence by a Fourier spectral method on the Earth Simulator, in: SC '02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing, 2002, pp. 1–17. doi:10.1109/SC.2002.10052.

[3] T. Ishihara, K. Morishita, M. Yokokawa, A. Uno, Y. Kaneda, Energy spectrum in high-resolution direct numerical simulations of turbulence, Phys. Rev. Fluids 1 (2016) 082403. doi:10.1103/PhysRevFluids.1.082403.

[4] K. Ravikumar, D. Appelhans, P. K. Yeung, GPU acceleration of extreme scale pseudo-spectral simulations of turbulence using asynchronism, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '19, Association for Computing Machinery, New York, NY, USA, 2019. doi:10.1145/3295500.3356209.

[5] T. Ishihara, Y. Kaneda, M. Yokokawa, K. Itakura, A. Uno, Small-scale statistics in high-resolution direct numerical simulation of turbulence: Reynolds number dependence of one-point velocity gradient statistics, J. Fluid Mech. 592 (2007) 335–366. doi:10.1017/S0022112007008531.

[6] R. Egawa, K. Komatsu, S. Momose, Y. Isobe, A. Musa, H. Takizawa, H. Kobayashi, Potential of a modern vector supercomputer for practical applications: performance evaluation of SX-ACE, The Journal of Supercomputing 73 (9) (2017) 3948–3976. doi:10.1007/s11227-017-1993-y.

[7] T. Soga, A. Musa, Y. Shimomura, R. Egawa, K. Itakura, H. Takizawa, K. Okabe, H. Kobayashi, Performance evaluation of NEC SX-9 using real science and engineering applications, in: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09, Tohoku University, 2009, pp. 28:1–28:12. doi:10.1145/1654059.1654088.