J. Parallel Distrib. Comput. 113 (2018) 77–91

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Scaling of a Fast Fourier Transform and a pseudo-spectral fluid solver up to 196608 cores

Anando G. Chatterjee [a], Mahendra K. Verma [a], Abhishek Kumar [a,*], Ravi Samtaney [b], Bilel Hadri [c], Rooh Khurram [c]

[a] Department of Physics, Indian Institute of Technology Kanpur, Kanpur 208016, India
[b] Mechanical Engineering, Division of Physical Science and Engineering, King Abdullah University of Science and Technology, Thuwal, 23955-6900, Saudi Arabia
[c] KAUST Supercomputing Laboratory, King Abdullah University of Science and Technology, Thuwal, 23955-6900, Saudi Arabia

HIGHLIGHTS

- Scalability of Fast Fourier Transform on maximum of 196608 cores.
- Implementation and scaling of a scalable pseudo-spectral code Tarang.
- Comparative performance study of FFT and Tarang on Blue Gene/P and Cray XC40.
- Deduce Kolmogorov-like spectrum for Rayleigh–Bénard convection using simulation.

ARTICLE INFO

Article history:
Received 21 July 2017
Received in revised form 16 October 2017
Accepted 25 October 2017
Available online 3 November 2017

Keywords:
Extreme-resolution turbulence simulation
Pseudospectral method
Fast Fourier transform
Turbulence simulation

ABSTRACT

In this paper we present scaling results of a FFT library, FFTK, and a pseudospectral code, Tarang, on grid resolutions up to 8192^3 grid using 65536 cores of Blue Gene/P and 196608 cores of Cray XC40 supercomputers. We observe that communication dominates computation, more so on the Cray XC40. The computation time scales as $T_{\text{comp}} \sim p^{-1}$, and the communication time as $T_{\text{comm}} \sim n^{-\gamma_2}$ with γ_2 ranging from 0.7 to 0.9 for Blue Gene/P, and from 0.43 to 0.73 for Cray XC40. FFTK, and the fluid and convection solvers of Tarang exhibit weak as well as strong scaling nearly up to 196608 cores of Cray XC40. We perform a comparative study of the performance on the Blue Gene/P and Cray XC40 clusters.

1. Introduction

The Fast Fourier Transform (FFT), first discovered by Cooley and Tukey [7], is an important tool for image and signal processing, and radio astronomy. It is also used to solve partial differential equations, fluid flows, density functional theory, many-body theory, etc. For a three-dimensional N^3 grid, FFT has large time complexity $\mathcal{O}(N^3 \log N^3)$ for large N (e.g. 4096 or 8192). Hence, parallel algorithms have been devised to compute FFT of large grids.

One of the most popular opensource FFT libraries is FFTW (Fastest Fourier Transform in the West) [15,14] in which a three-dimensional (3D) array is divided into slabs (hence called *slab decomposition*) as shown in Fig. 1(a). Hence, we can employ a maximum of N cores in FFTW operations on an array of size N^3. This is a severe limitation since present-day supercomputers offer several hundreds of thousands of cores for use. To overcome this issue, Pekurovsky [24] employed a *pencil decomposition* in which the data is divided into pencils, as shown in Fig. 1(b). This method allows usage of a maximum of N^2 cores, equal to the maximum number of pencils.

In this paper we implement a pencil-based FFT using the algorithm of P3DFFT, and then construct a pseudo-spectral fluid solver. As described earlier, the most well-known pencil-based FFT library is P3DFFT, written by Pekurovsky [24] who reported that the total time T for a FFT operation is a sum of computation time a/p and communication time $b/p^{2/3}$, where p is the number of cores. This scaling was deduced based on runs using a grid of 8192^3 points on a Cray XT5 with a 3D torus interconnect, and 65 536 cores. In these tests, the communication time dominates the computation time due to the `MPI_Alltoall` data transfer. Chan et al. [6] studied scaling of P3DFFT on a 16 384 nodes of Blue Gene/L system; they

* Corresponding author.
E-mail addresses: anandogc@iitk.ac.in (A.G. Chatterjee), mkv@iitk.ac.in (M.K. Verma), abhkr@iitk.ac.in (A. Kumar), ravi.samtaney@kaust.edu.sa (R. Samtaney), bilel.hadri@kaust.edu.sa (B. Hadri), rooh.khurram@kaust.edu.sa (R. Khurram).

https://doi.org/10.1016/j.jpdc.2017.10.014
0743-7315/

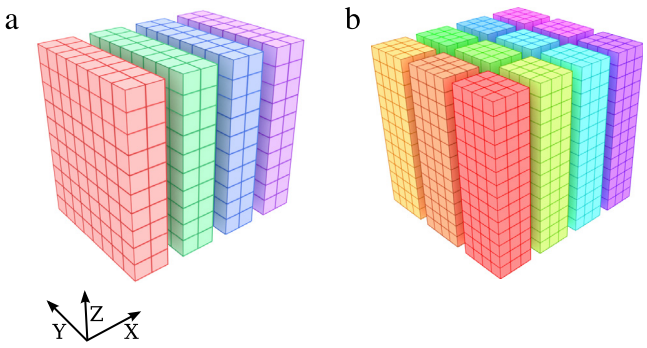


Fig. 1. (a) Slab decomposition of an array. (b) Pencil decomposition of an array.

reported that a combination of the network topology and the communication pattern of P3DFFT can affect performance.

Pippig and Potts [27] devised a similar FFT named PFFT, and ran it on a large number of cores; they observed that PFFT has a similar scaling as P3DFFT. Richards et al. [29] performed scalability analysis for their two-dimensional pencil FFT library on Blue Gene/P. Czechowski et al. [8] analysed the memory hierarchy traffic and network communication in GPU-based FFT, DiGPUFFT. Mininni et al. [20] employed hybrid scheme (MPI + OPENMP) to use large number of cores optimally; their FFT implementations scales well on 1536^3 and 3072^3 grids for approximately 20 000 cores with 6 and 12 threads on each socket.

We have devised another pencil-based FFT called FFTK (FFT Kanpur) and tested it on 65 536 cores of Blue Gene/P (Shaheen I) and 196 608 cores of Cray XC40 (Shaheen II) of KAUST. We computed separately the time required for the computation and communications during the FFT process; we showed that the computation time scales linearly, while the communication component approaches the ideal bisection bandwidth scaling for large arrays. In this paper we compare the performance of FFTK on Blue Gene/P and Cray XC40, and show that the relative speed of cores and switch matters for the efficiency. We show later that the per-core efficiency of Cray XC40 is lower than that of Blue Gene/P because the speed of the interconnect of Cray XC40 has not increased in commensurate with the speed of the processor. FFTK library is available for download at http://www.turbulencehub.org/codes/fftk.

FFT is used extensively in a pseudo-spectral method, which is one of the most accurate methods for solving differential equations due to the exponential convergence of derivative computations in this method [3,5]. In addition, a major advantage of the spectral method is that it allows for a convenient scale-by-scale analysis of the relevant quantities. Consequently, we can compute interactions and energy transfers among structures at different scales using the spectral method. Such scale-by-scale analysis is generally quite cumbersome in finite-difference and finite-volume solvers. Note that the convolution stemming from nonlinear terms in a partial differential equation is computed using the FFT. Though spectral method can be used to solve various types of partial differential equations (PDEs), in this paper we focus on the PDEs for fluid flows.

Turbulence remains one of the unsolved problems of classical physics, and no analytical solution of fluid equations in the turbulent limit is available at present. Hence numerical simulation is very handy for the analysis of turbulent flows. Unfortunately, the grid resolution and computational time required for turbulence simulations is very large. Hence such simulations are performed on large high performance computing clusters (HPC).

Many researchers [43,17,12,42,31,13,11,10,40,41,30,9] have performed high resolution turbulence simulations. Yokokawa et al. [43] performed first turbulence simulation on 4096^3 grid using the *Earth Simulator*. Donzis et al. [12] performed turbulence simulation on 4096^3 grid using P3DFFT library; they employed 32 768 cores of Blue Gene/L and Cray XT4, and reported that the effective performance of the FFT is approximately 5% of the peak performance due to the extensive communication and cache misses. Yeung et al. [42] performed pseudo spectral simulations of fluid turbulence on one of the highest resolution grids (8192^3) to study extreme events. Rosenberg et al. [31] simulated rotating stratified turbulence on a 4096^3 grid and studied its energy spectrum.

We have implemented a pseudospectral code named Tarang based on the FFTK library. Tarang is a parallel and C++ code written as a general PDE (partial-differential equation) solver. Using Tarang, we can compute incompressible flows involving pure fluid, magnetohydrodynamic flows [19], liquid metal flows [28], Rayleigh–Bénard convection [39,23,18], rotating convection [26], and rotating flows [32], etc. Simulation of convective flows involves rigid walls for which periodic boundary condition is not applicable. For such simulations, no-slip or free-slip boundary conditions are employed. Another major feature of Tarang is its rich library for computing the energy transfers in turbulence, e.g., energy flux, shell-to-shell and ring-to-ring energy transfers, etc. [18,35,21]. Tarang also enables us to probe any point in the Fourier space or in the real space [28]. In the present paper we empirically demonstrate that the flow solvers of Tarang scale well on HPC supercomputers.

The outline of the paper as follows: In Sections 2 and 3, we describe the numerical scheme and parallelization strategy. Section 4 contains a brief discussion about the HPC systems used for scaling analysis. We describe our scaling results for FFTK, the fluid solver, and the Rayleigh–Bénard solver in Sections 5, 6, and 7, respectively. We run FFTK on Blue Gene/P and benchmark this against the performance of P3DFFT on the same platform. We do not undertake a detailed comparison between the two FFT packages due to lack of space and comparison resources. We also deduce the Kolmogorov-like spectrum for Rayleigh–Bénard convection using analysis of data from a 4096^3 grid simulation. We conclude in Section 8.

2. Numerical scheme

In a pseudospectral code, approximately 70% to 80% of the total time is spent on the forward and inverse Fourier transforms. In this section we briefly explain the numerical schemes for FFT and the spectral solver Tarang.

2.1. Fast Fourier transform

The inverse Fourier transform is defined as

$$f(x,y,z) = \sum_{k_x,k_y,k_z} \hat{f}(k_x,k_y,k_z)\phi_{k_x}(x)\phi_{k_y}(y)\phi_{k_z}(z), \tag{1}$$

where $\hat{f}(k_x,k_y,k_z)$ is the Fourier transform of $f(x,y,z)$. Here we compute $f(x,y,z)$ from $\hat{f}(k_x,k_y,k_z)$. The functions $\phi_k(s)$ are the basis functions that appear in the following forms:

$$\text{Fourier}: \phi_k(s) = \exp(iks), \tag{2a}$$

$$\text{Sine}: \phi_k(s) = 2\sin(ks), \tag{2b}$$

$$\text{Cosine}: \phi_k(s) = 2\cos(ks), \tag{2c}$$

where k could be k_x, k_y, or k_z, and s could be x, y, or z. We use Fourier basis function for the periodic boundary condition, and employ the sine and cosine basis functions for the free-slip boundary condition. The FFTK library can perform the above transformations in different combinations. For example, for the periodic boundary

condition along the three directions, we employ Fourier basis function

$$\exp(ik_x x + ik_y y + ik_z z). \tag{3}$$

But for the free-slip boundary condition at all the three walls, u_z, the z-component of the velocity is expanded using the basis function

$$8\cos(k_x x)\cos(k_y y)\sin(k_z z). \tag{4}$$

Similar schemes are used for u_x and u_y. We term the above two basis functions as FFF and SSS respectively. In a similar fashion, we define other basis functions—SFF for one free-slip wall direction and two periodic directions, SSF for two free-slip wall directions and one periodic direction, and ChFF for one no-slip wall direction and two periodic directions. Note that the inverse of sine (or cosine) basis function is sine (or cosine) itself, but $\exp(ikx)$ and $\exp(-ikx)$ are mutual inverses of each other.

In Section 3 we will describe the FFT implementation in our library.

2.2. Spectral solver

As described in the introduction, the pseudospectral scheme is one of the most accurate methods to solve partial differential equations. In the following, we describe its implementation for computing fluid flows. The incompressible fluid equation is described using the celebrated Navier Stokes equation, which is

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u}, \tag{5}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{6}$$

where $\mathbf{u}$ is the velocity field, p is the pressure field, and ν is the kinematic viscosity. For simplicity we take the density of the fluid, ρ, to be unity. We rewrite the above equations in Fourier space:

$$(\partial_t + \nu k^2)\hat{u}_j(\mathbf{k}, t) = -ik_l\widehat{u_l u_j}(\mathbf{k}, t) - ik_j\hat{p}(\mathbf{k}, t), \tag{7}$$

$$k_j\hat{u}_j(\mathbf{k}) = 0, \tag{8}$$

where $i = \sqrt{-1}$. The above equations are time advanced using standard methods, e.g., Runge Kutta scheme. The $\widehat{u_l u_j}$ term of Eq. (7) becomes a convolution in spectral space that is very expensive to compute. Orszag [22] devised an efficient scheme in which $\widehat{\mathbf{u}}(\mathbf{k}, t)$ is transformed to real space, components of which are multiplied with each other, and the product is then transformed back to Fourier space. Due to the multiplication of arrays in real space, this method is called *pseudospectral method*. This multiplication however generates aliasing errors, which are mitigated by filling up only 2/3 of the array in each direction. See Boyd [3] and Canuto [5] for details.

A spectral transform is general, and it can involve basis functions from Fourier series, sines and cosines, Chebyshev polynomials, spherical harmonics, or a combination of these functions. The FFTK library uses Fourier, sines, and cosine functions only. We plan to incorporate Chebyshev polynomials and spherical harmonics in the future. In this paper, we solve Eqs. (7), (8) in a periodic box (FFF basis) using Tarang. Therefore we illustrate the implementation and usage of FFF basis, and then describe scaling analysis of FFT, and the fluid and convection solvers.

We have computed fluid and magnetohydrodynamic flows, liquid metal flows, Rayleigh–Bénard convection (RBC), rotating convection, and rotating flows using Tarang. In the following we will illustrate one of the above modules, the RBC solver, whose governing equations are

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla\sigma + \alpha g\theta\hat{z} + \nu\nabla^2\mathbf{u}, \tag{9}$$

$$\frac{\partial \theta}{\partial t} + (\mathbf{u}\cdot\nabla)\theta = \frac{\Delta}{d}u_z + \kappa\nabla^2\theta, \tag{10}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{11}$$

where θ is the temperature fluctuation from the steady conduction state,

$$T(x, y, z) = T_c(z) + \theta(x, y, z), \tag{12}$$

with T_c as the conduction temperature profile), σ is the pressure fluctuation, $\hat{z}$ is the buoyancy direction, Δ is the temperature difference between the two plates that are separated by a distance d, ν is the kinematic viscosity, and κ is the thermal diffusivity.

We solve a nondimensionalized version of the RBC equations, which are obtained using d as the length scale, $(\alpha g\Delta d)^{1/2}$ as the velocity scale, and Δ as the temperature scale:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla\sigma + \theta\hat{z} + \sqrt{\frac{\mathrm{Pr}}{\mathrm{Ra}}}\nabla^2\mathbf{u}, \tag{13}$$

$$\frac{\partial \theta}{\partial t} + (\mathbf{u}\cdot\nabla)\theta = u_z + \frac{1}{\sqrt{\mathrm{RaPr}}}\nabla^2\theta. \tag{14}$$

Here the two important nondimensional parameters are the Rayleigh number $\mathrm{Ra} = \alpha g\Delta d^3/\nu\kappa$, and the Prandtl number $\mathrm{Pr} = \nu/\kappa$. We perform our simulations in a cubic fluid domain of unit size in each direction.

For the scaling analysis, we solve the RBC equations (Eqs. (13), (14)) with FFF and SFF basis functions. We also perform a production run for computing the energy spectrum and energy flux during the statistical steady state. Note that the SFF basis function corresponds to the free-slip boundary condition for which the velocity field at the top and bottom plates ($z = 0, 1$):

$$u_z = \partial_z u_x = \partial_z u_y = 0, \tag{15}$$

and periodic boundary conditions imposed on the vertical side walls. For the temperature field, we employ isothermal boundary condition ($\theta = 0$) at the top and bottom plates, and periodic boundary conditions at the side walls.

Many fluid flow simulations, especially Rayleigh–Bénard convection, employ no-slip boundary condition that requires Chebyshev basis functions (ChFF), which are somewhat complex to implement. Note however that the ChFF basis has its own limitations. For example, the energy spectrum and flux computations performed in the Fourier space requires a uniform grid. The collocation points in ChFF are nonuniform, and hence we need to interpolate the data to a uniform mesh that induces errors. Therefore, the free-slip basis functions that involves uniform mesh are convenient for studying the energy spectrum and flux. We remark the no-slip boundary condition captures the boundary layers near the walls. The flow near the walls contributes to the energy spectrum at small scales or large wavenumbers, therefore the boundary layers at the top and bottom walls do not significantly impact the inertial-range energy spectrum and flux (see Section 7). For such studies, the free-slip basis suffices.

The above example illustrates that we can easily perform simulations with different boundary conditions using Tarang solvers just by changing the basis function in the input file. We report our results for the RBC solvers in Section 7.

3. Parallelization strategy

In the following discussion we illustrate our implementation for the FFF basis. We divide the data equally among all the cores for load balancing. For the pencil decomposition, we divide the data into rows and columns, and p cores into a core grid $p_{\mathrm{row}} \times p_{\mathrm{col}} = p$ as shown in Fig. 2. The cores are divided into two MPI communicators: MPI_COM_ROW and MPI_COM_COL (see Fig. 2). In

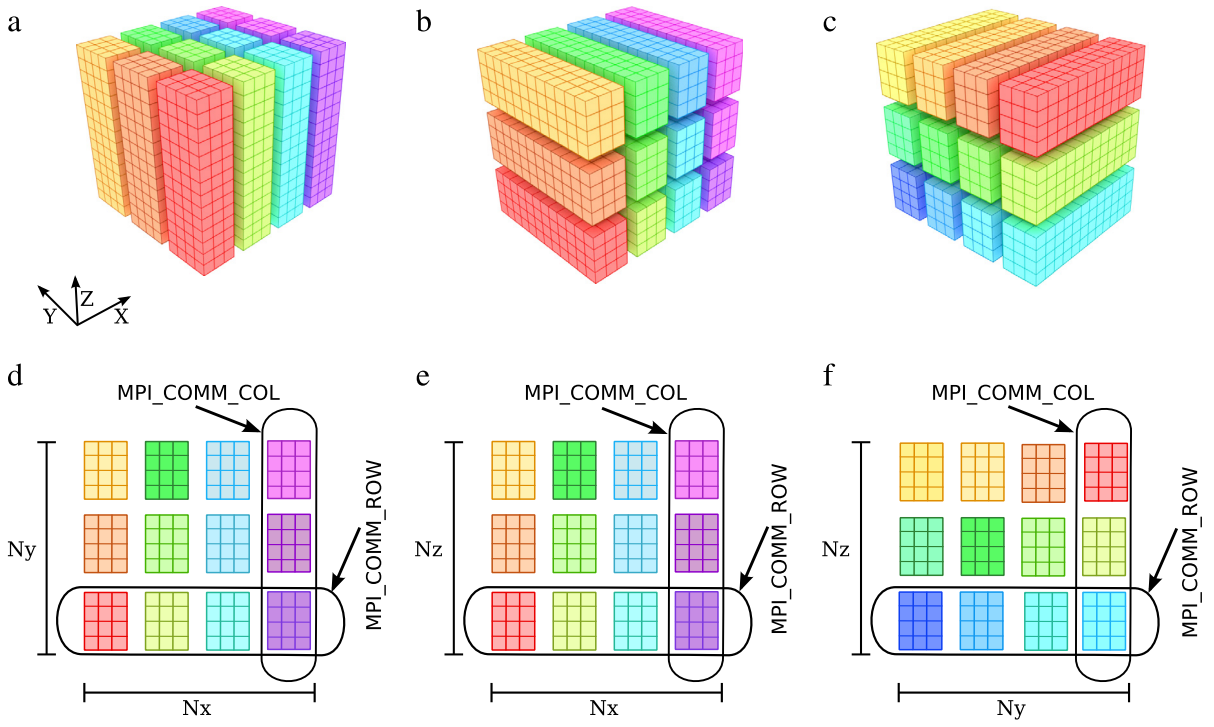


Fig. 2. Pencil decomposition: (a) real space data, (b) intermediate configuration, (c) data in Fourier space. (d, e, f) Division of cores into p_{row} and p_{col} such that $p = p_{\text{row}} \times p_{\text{col}}$ as seen in XY, XZ, and YZ projections respectively. Here $N_x = N_y = N_z = 12$. In the subfigures (a, d), $p_{\text{row}} = 3, p_{\text{col}} = 4$, thus each core contains $N_x/p_{\text{col}} \times N_y/p_{\text{row}} \times N_z = 3 \times 4 \times 12$ data points.

real space [Fig. 2(a)] each core has $N_x/p_{\text{col}} \times N_y/p_{\text{row}} \times N_z$ data points.

The forward FFT transform (from real to complex) follows the following set of steps:

1. We perform forward FFT, `r2c` real-to-complex, along the Z-axis for each data column.
2. We perform `MPI_Alltoall` operation among the cores in `MPI_COM_COL` to transform the data from the real configuration [Fig. 2(a)] to the intermediate configuration [Fig. 2(b)].
3. After interprocess communication, we perform forward `c2c` (complex-to-complex) transform along the Y-axis for each pencil of the array.
4. We perform `MPI_Alltoall` operation among the cores in `MPI_COM_ROW` to transform the data from the intermediate configuration [Fig. 2(b)] to the Fourier configuration [Fig. 2(c)].
5. Now we perform forward `c2c` transform along the X-axis for each pencil [see Fig. 2(c)].

For one-dimensional FFT operations, we use the FFTW transforms. During steps 2,4 of the above, we employ transpose-free data transfer among the communicators, as described in Appendix. This scheme avoids local transpose during these processes. However, after the `MPI_Alltoall`, the data along a column are not contagious. Hence, we need to employ strided FFT, which is efficient, and is provided in the FFTW library [14]. Note, however, this is prone to cache misses since the columnar/row data are not contagious. See Appendix for details. We also remark that FFTK and Tarang use a meta-template C++ library, `Blitz++` [2,34] for array manipulation; this library provides efficient operations for arrays.

This completes the forward transform. The inverse transform is reverse of the above operation. Note that the above strategy is general, and it works for all the basis functions. Our library also works for two-dimensional (2D) data, for which we set $N_y = 1$. The intermediate state is avoided for 2D Fourier transforms. Also note that the slab FFT can be performed by setting $p_{\text{row}} = 1$, and again, the configuration (b) of Fig. 2 is avoided.

The other functions of a spectral solver that require parallelization are multiplication of arrays elements and input/output (I/O). A multiplication of array elements is trivial to parallelize. Since the data-size involved in high-resolution turbulence simulation is very large, of the order of several terabytes, it is more efficient to use parallel I/O. In our spectral code, we use the `HDF5` library to perform parallel I/O.

Our code has important sets of functions to compute energy flux, shell-to-shell energy transfers, and ring-to-ring energy transfers. These quantities are computed using FFT [35,21]. For brevity, we omit discussion on these implementations in the present paper. In Section 7, we will briefly describe the computation of the energy flux for RBC. We have exploited this feature to implement `FFF`, `SFF`, `SSF`, `SSS` basis functions in two and three dimensions in a single code. We also make use of efficient libraries such as `Blitz++` and `HDF5`. These are some of the unique features of FFTK and Tarang.

4. About the HPC systems

We performed scaling tests of our FFT library and pseudospectral code on Shaheen I, a Blue Gene/P supercomputer, and Shaheen II, a Cray XC40 supercomputer, of King Abdullah University of Science and Technology (KAUST). First we provide some details of these systems.

4.1. Blue Gene/P

The Blue Gene/P supercomputer consists of 16 racks with each rack containing 1024 quad-core, 32-bit, 850 MHz PowerPC compute nodes. Hence the total number of cores in the system is 65 536.

It also has 65 536 GB of RAM. The Blue Gene/P nodes are interconnected by a three-dimensional point-to-point *torus* network. The theoretical peak speed of the Blue Gene/P supercomputer is 222 Tera FLOP/s (Floating point operations per second).

4.2. Cray XC40

The Cray XC40 supercomputer has 6174 dual-socket compute nodes each containing two Intel Haswell processors with 16 cores, with each core running at a clock speed of 2.3 GHz. In aggregate, the system has a total of 197 568 cores and 790 TB of memory. The compute nodes, contained in 36 water-cooled XC40 cabinets, are connected via the Aries High Speed Network. Cray XC40 adopts a dragonfly topology that yields 57% of the maximum global bandwidth between the 18 groups of two cabinets [16]. Shaheen II delivers a theoretical peak performance of 7.2 Peta FLOP/s and a sustained LINPACK performance of 5.53 Peta FLOP/s.

A parallel program involves computation time T_{comp} and communication time across nodes T_{comm} [33,4]. Thus the total time T for a parallel program is

$$T = T_{\text{comp}} + T_{\text{comm}}. \tag{16}$$

We report T_{comp} and T_{comm} for the execution of the FFTK library on the Blue Gene/P and Cray XC40 supercomputers. Since the data is divided equally among all the cores, we expect $T_{\text{comp}} \sim p^{-1}$, where p is the number of cores; we observe the above scaling in all our tests. Note that FFT, which is a dominant operation in a pseudospectral solver, involves `MPI_Alltoall` communications. Hence communication time is the most dominant component of the total time.

For the fluid and RBC solvers, it is difficult to disentangle the computation and communication times since they involve many functions, hence we report only the total time for these solvers. These results are presented in the following sections.

5. Scaling results of FFTK

We perform FFTK forward and inverse transforms several times (100 to 1000) for large N^3 grids, and then present an average time taken for a pair of forward and inverse transforms. We compute T_{comp} and T_{comm} for various combinations of grid sizes and number of cores, and observe that

$$T_{\text{comp}} = \frac{1}{c_1} p^{-\gamma_1}, \tag{17a}$$

$$T_{\text{comm}} = \frac{1}{c_1} n^{-\gamma_2}, \tag{17b}$$

$$T^{-1} = \frac{1}{C} p^{\gamma}, \tag{17c}$$

where c_1, c_2, C, γ_1, γ_2, and γ are constants, p is the number of cores, and n is the number of nodes. Hence the total time per FFT operation is

$$T = c_1 D \left(\frac{1}{p^{\gamma_1}}\right) + c_2 D \left(\frac{1}{n^{\gamma_2}}\right) = C \left(\frac{1}{p^{\gamma}}\right), \tag{18}$$

where $D = N^3$ is the data size. We measure T_{comp} and T_{comm} by computing the time taken by the respective code-segments using the MPI function `MPI_Wtime`. We record the time when the process enters and leaves the code segment, and then take their difference that yields T_{comp} and T_{comm}.

After the above general discussion, we now describe our results specific to the Blue Gene/P supercomputer.

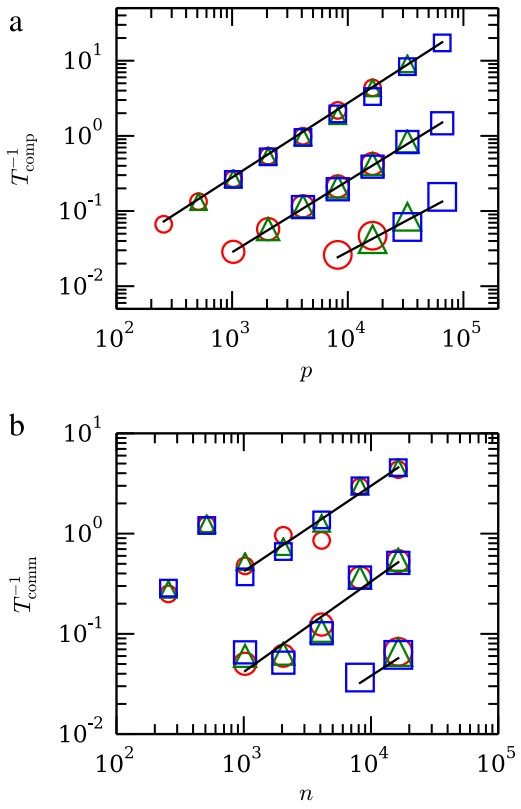


Fig. 3. Scalings of the FFTK library on Blue Gene/P: (a) Plot of inverse computation time T_{comp}^{-1} vs. p (number of cores) for 1ppn (red circle), 2ppn (green triangle), 4ppn (blue square). Here ppn represents number of MPI processes per node. The data for grids 2048^3, 4096^3, and 8192^3 are represented by the same symbols but with increasing sizes. The plots show that FFTK exhibits strong scaling in Blue Gene/P. (b) Plot of inverse communication time T_{comm}^{-1} vs. n (number of nodes) with the above notation. T_{comm}^{-1} for $p = 256$ and 512 exhibits a better scaling due to the slab decomposition employed. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5.1. Scaling on Blue Gene/P

In Fig. 3 we plot T_{comp} and T_{comm}. Our results indicate that the computation time scales as $T_{\text{comp}} \propto p^{-1}$. However, the communication among the nodes takes maximum time in an FFT operation. In the following discussion, we sketch the scaling arguments for T_{comm} that was first provided by Pekurovsky [24].

A Blue Gene/P supercomputer has a torus interconnect for which we estimate the *bisection bandwidth* B, which is defined as the available bandwidth when the network is bisected into two partitions. For 3D torus, bisection bandwidth is proportional to the area in the network topology, hence $B \propto (n')^{2/3}$, where n' is the number of nodes used in communication.

FFT involves `MPI_Alltoall` communication, hence, in the slab division, $n' = n$, the total number of nodes, and the data to be communicated in the network is $D = N^3$. Therefore, the inverse of the communication time for each FFT is

$$T_{\text{comm,slab}}^{-1} \sim \frac{B}{D} \sim n^{2/3}. \tag{19}$$

We also remark that the communication time depends on number of interacting nodes, not cores. The inter-core or intra-node communication (among the cores within a node) is typically much faster than the inter-node communication across an interconnect.

In the pencil division, the nodes are divided into row nodes (n_{row}) and column nodes (n_{col}). We estimate $n_{\text{row}} \approx n_{\text{col}} \approx n^{1/2}$.

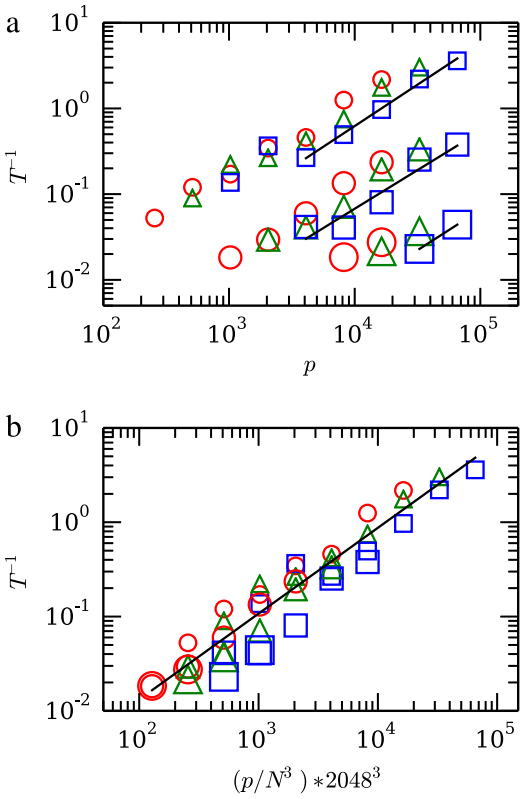


Fig. 4. Scalings of FFTK on Blue Gene/P: (a) Plot of inverse of total time T^{-1} vs. p exhibits strong scaling. (b) Plot of T^{-1} vs. p/N^3 exhibits weak scaling with the exponent $\gamma = 0.91 \pm 0.04$. We follow the same colour and symbol convention as Fig. 3. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
FFTK scaling on Blue Gene/P: The exponents γ_1 for the computation time (T_{comp}), γ_2 for the communication time (T_{comm}), and γ for the total time (T) [refer to Eq. (17) for definition]. The maximum nodes used is 16 384 with 1ppn, 2ppn, and 4ppn.

γ	ppn	2048^3	4096^3
γ_1	1	1.00 ± 0.01	0.97 ± 0.01
	2	1.00 ± 0.02	0.96 ± 0.01
	4	1.00 ± 0.03	0.95 ± 0.03
γ_2	1	0.7 ± 0.1	0.9 ± 0.1
	2	0.7 ± 0.1	0.8 ± 0.2
	4	0.7 ± 0.1	0.8 ± 0.2
γ	1	0.87 ± 0.05	0.94 ± 0.05
	2	0.81 ± 0.05	0.96 ± 0.09
	4	0.76 ± 0.07	0.9 ± 0.1

Hence each communication within a row (or a column) involves $n' \approx n^{1/2}$ number of nodes during `MPI_COM_ROW` or `MPI_COM_COL` communications. Hence, the effective bisection bandwidth, B_e, is

$$B_e \sim (n')^{2/3} = (n^{1/2})^{2/3} = n^{1/3}. \tag{20}$$

In this decomposition, the data per node is N^3/n. During a communication, either row nodes or column nodes are involved. Hence, the data to be communicated during a `MPI_Alltoall` operation is

$$D = (N^3/n) \times n^{1/2} = N^3/n^{1/2}. \tag{21}$$

Therefore, the inverse of communication time for each FFT is

$$T_{\text{comm}}^{-1} \sim \frac{B_e}{D} \sim n^{5/6} \approx n^{0.83}. \tag{22}$$

Table 2
Comparison of FFTK and P3DFFT on Blue Gene/P for 8192 nodes with 1ppn and 4ppn. Here p = nodes $\times$ppn.

Grid	p	Time/step (s) FFTK	Time/step (s) P3DFFT
4096^3	8192×1	8.18	8.06
4096^3	8192×4	4.14	4.06
8192^3	8192×1	71.2	70.0
8192^3	8192×4	45.7	46.2

We performed our scaling tests on arrays of size 2048^3, 4096^3, and 8192^3 using cores ranging from 256 to 65 536. Each node of Blue Gene/P has 4 cores, hence we performed our simulations on 1, 2, and 4 cores per node, denoted as 1ppn, 2ppn, and 4ppn respectively. We present all our results in Fig. 3, with the subfigures (a, b) exhibiting the inverse of computation and communication timings respectively. We represent 1ppn, 2ppn, and 4ppn results using circles, triangles, and squares respectively, and the grid sizes 2048^3, 4096^3, and 8192^3 using increasing sizes of the same symbols. Fig. 3 shows that T_{comm}^{-1} for $p = 256$ and 512 shows better scaling than those for larger number of processors. This is attributed to the slab decomposition. Note however that the slab decomposition is possible only when number of processors is smaller than the number of planes of the data (N for N^3 grid). In Fig. 4(a, b), we exhibit T^{-1} vs. p and T^{-1} vs. p/N^3 to test strong and weak scaling, respectively.

We compute the exponents γ_1, γ_2, and γ of Eq. (17) using linear regression on the data of Fig. 3 and 4(a). In Table 1, we list the exponents for 1ppn, 2ppn, and 4ppn and grids sizes of 2048^3 and 4096^3. As expected, the exponent $\gamma_1 \approx 1$ since the data is equally distributed among all the cores. The exponent γ_2 is approximately 0.7 for 2048^3 for all three cases, but it ranges from 0.8 to 0.9 for 4096^3. The increase in γ_2 with the grid size is probably due to the larger packets communicated for 4096^3 grids. The exponent γ_2 is quite close to the theoretical estimate of $5/6 \approx 0.83$ for 4096^3 grid [see Eq. (22)]. Our computation also shows the best match for γ_2 with the theoretical estimate is for 1ppn, and it decreases slightly for larger ppn. The variation with ppn is due to *cache misses*.

We revisit Fig. 4(a) that shows a power law scaling, $T^{-1} \propto p^{\gamma}$, close to the ideal exponent $\gamma = 5/6$ [see Eq. (22)]. This feature is called *strong scaling*. Naturally the larger grids take longer time than the smaller grids. However, when we increase p and N^3 proportionally, all our results collapse into a single curve, as exhibited in T^{-1} vs. p/N^3 plot of Fig. 4(b). Thus FFTK exhibits both strong and weak scaling.

Interestingly T_{comp} and T_{comm} are comparable on the Blue Gene/P, which is due to the fact that the compute processors are slow, but the interconnect is relatively fast. Hence the total time T is impacted by both T_{comp} and T_{comm}. As a result, the γ is reasonably close to unity, thus yielding an approximate linear scaling, at least for the 4096^3 grid. We also remark that we have performed FFT for 8192^3 grid with 8192 and 16 384 nodes; it was not possible with lower number of nodes due to memory limitations. We do not have a reliable scaling exponent for 8192^3 grid due to lack of data points.

We compare the timings of FFTK with the popular library P3DFFT for 4096^3 and 8192^3 grids using 8192 nodes with 1ppn and 4ppn. The comparison listed in Table 2 indicates that both the libraries are equally efficient. For comparison we also compute the efficiency of our computation using effective FLOP rating. A pair of forward and inverse FFT involves $2 \times 5N^3 \log_2 N^3$ floating point operations [15]. Using this formula we estimate the effective FLOP rating of the supercomputer for various grid sizes and ppn. The results are listed in Table 3. A comparison of the above performance with the average theoretical rating of each core (approximate 3.4 Giga FLOP/s) suggests that the efficiency of the system for a FFT

Table 3
FFTK on Blue Gene/P: Effective FLOP rating in Giga FLOP/s of Blue Gene/P cores for various grid sizes and ppn. The efficiency E is the ratio of the effective per-core FLOP rating and the peak FLOP rating of each core (approximately 3.4 G FLOP/s).

Grid	ppn	Giga FLOP/s	E
2048^3	1	0.38	0.11
	2	0.28	0.082
	4	0.17	0.050
4096^3	1	0.36	0.11
	2	0.25	0.073
	4	0.14	0.041
8192^3	1	0.36	0.11
	2	0.26	0.076
	4	0.15	0.044

ranges from approximately 5% (for 4ppn) to 10% (for 1ppn) of the peak performance. The loss of performance is due to large communication time and cache issues during a FFT operation (see Appendix). Typically, efficiency of a HPC system is measured using $T_p/(pT_1)$ where T_p is the time taken to perform operation using p processors. The data for large grids, e.g. 1024^3, cannot fit in the memory of a single processor, hence we cannot compute T_1 and hence $T_p/(pT_1)$. Therefore we use a more stringent measure. We measure the efficiency as the ratio of the per-core FLOP rating and the peak rating. We list this efficiency in Table 3.

In the next subsection we will discuss the scaling of FFTK on Cray XC40.

5.2. Scaling on Cray XC40

Each node of Cray XC40 has 32 compute cores, with each core having an approximate rating of 36.8 Giga FLOP/s. Thus each core of Cray XC40 is approximately 10 times faster than that of Blue Gene/P. Hence, for given grid size and p, T_{comp} for Cray XC40 is much smaller than that for Blue Gene/P.

The Cray XC40 employs dragonfly topology which consists of hierarchy of structures that yields bandwidth proportional to the number of interacting nodes. Hence the bandwidth is

$$B_e \sim n', \tag{23}$$

where n' is the number of interacting nodes. For the pencil decomposition, the total number of nodes n is divided as $n = n_{\text{row}} \times n_{\text{col}}$. Hence $n' = n/n_{\text{row}}$ for `MPI_COMM_COL` communicator and $n' = n/n_{\text{col}}$ for `MPI_COMM_ROW` communicator. Note that the data to be communicated during `MPI_Alltoall` operation is $(N^3/n)n'$. Hence, the inverse of the communication time is

$$T_{\text{comm}}^{-1} \sim \frac{B_e}{D} \approx \frac{n'}{(N^3/n)n'} \sim n, \tag{24}$$

implying a linear scaling.

Comparison of T_{comm} for Blue Gene/P and Cray XC40 reveal that the bandwidth is larger for Cray XC40 than Blue Gene/P. Hence, for a given set of N, n, and D (the data to be communicated), the time for communication is smaller for Cray XC40 than Blue Gene/P (see Eqs. (22), (24)). Thus, the data communication in XC40 is more efficient than that in Blue Gene/P. However, we will show later that the gain in the speed of the interconnect is in commensurate with the increase in the computational power of the processor. Hence, for the FFT computation, the overall efficiency of Cray XC40 is lower than that for Blue Gene/P. We also remark that Hadri et al. [16] showed that the maximum global bandwidth between the 18 groups of two cabinets is approximately 57% of the peak performance, hence we expect suboptimal performance for communications for FFT due to `MPI_Alltoall` data exchange.

We performed FFTK scaling on grids of sizes 768^3 to 6144^3 using cores ranging from 1536 to 196 608 (3×2^{16}). Each node

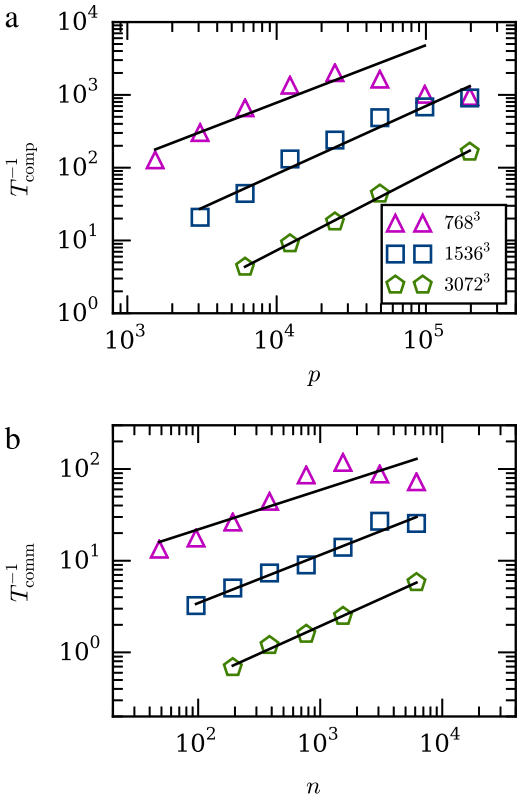


Fig. 5. Scalings of FFTK on Cray XC40 for the FFF basis: (a) Plots of T_{comp}^{-1} vs. p (number of cores) for 768^3, 1536^3, and 3072^3 grids. (b) Plots of T_{comm}^{-1} vs. n (number of nodes) using the above convention.

contains 32 cores, which implies a somewhat large number of ppn combinations. Hence we choose to use all the cores in a given node for maximum utilization. Both the grid sizes and number of cores are of the form 3×2^n since the maximum number of cores in Cray XC40, 3×2^{16}, is divisible by 3.

We perform scaling analyses for the FFF and SFF basis. However we present our results on T_{comp}, T_{comm}, and T in Figs. 5 and 6 for the FFF basis only since SFF basis has similar behaviour. We observe that the total computation time for the process is an order of magnitude smaller than the total communication time. Hence the efficiency of FFT is dominated by the `MPI_Alltoall` communication of the FFT. The figures show that $T_{\text{comp}}^{-1} \sim p^{\gamma_1}$, $T_{\text{comm}}^{-1} \sim n^{\gamma_2}$, and $T^{-1} \sim p^{\gamma}$, with minor deviations from the power law arising for 768^3 grid with large p's ($p \geq 98\,000$). Thus the data exhibits a strong scaling nearly up to 196 608 cores. We also observe that all the data nearly collapse to a single curve when we plot T^{-1} vs. p/N^3, hence FFTK exhibits both weak and strong scaling nearly up to 196 608 cores.

The exponents for various grids for the FFF and SFF basis are listed in Table 4. We observe that $\gamma_1 \approx 1$, except for 768^3, thus yielding a linear scaling for the computation time. The exponent γ_2 of the communication time ranges from 0.43 to 0.70 as we increase the grid size from 768^3 to 3072^3, which may be due to increased efficiency of the network for larger data size. As described above, the total time is dominated by the communication time, hence $\gamma \approx \gamma_2$. Also, the SFF basis appears to be slightly more efficient than the FFF basis.

We compute the effective FLOP rating of the supercomputer by dividing the total number of floating operations for a pair of FFT ($2 \times 5N^3 \log_2 N^3$) with the total time taken. This operation yields the average rating of each core of Cray XC40 as 0.45 to 0.64 Giga

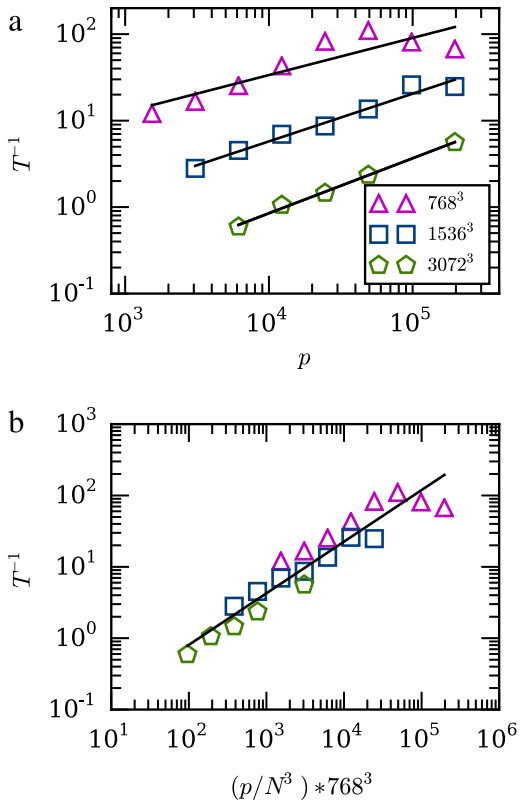


Fig. 6. Scalings of FFTK on Cray XC40 for the FFF basis: (a) plots of T^{-1} vs. p for 768^3, 1536^3, and 3072^3 grids. (b) plots of T^{-1} vs. p/N^3 exhibits weak scaling with an exponent of $\gamma = 0.72 \pm 0.03$.

Table 4
FFTK scaling on Cray XC40 for the FFF and SFF basis: The exponents γ_1 for the computation time (T_{comp}), γ_2 for the communication time (T_{comm}), and γ for the total time (T) [refer to Eq. (17) for definition]. Maximum cores used: 196 608.

Grid	γ_1	γ_2	γ
FFF			
768^3	0.79 ± 0.14	0.43 ± 0.09	0.43 ± 0.09
1536^3	0.93 ± 0.08	0.52 ± 0.04	0.55 ± 0.04
3072^3	1.08 ± 0.03	0.60 ± 0.02	0.64 ± 0.02
SFF			
768^3	0.82 ± 0.13	0.44 ± 0.03	0.46 ± 0.04
1536^3	0.97 ± 0.07	0.63 ± 0.02	0.66 ± 0.01
3072^3	0.99 ± 0.04	0.70 ± 0.05	0.73 ± 0.05

Table 5
FFTK on Cray XC40: Effective FLOP rating in Giga FLOP/s of Cray XC40 cores for various grid sizes and ppn. The efficiency E is the ratio of the effective per-core FLOP rating and the peak FLOP rating of each core (approximately 36 G FLOP/s).

Grid size	768^3	1536^3	3072^3
GFlop/s	0.45	0.53	0.64
E	0.013	0.015	0.018

FLOP/s that translates to 1.3% to 1.8% of the peak performance (36 Giga FLOP/s) (see Table 5).

It is important to contrast the efficiencies of the two HPC supercomputers discussed in this paper. The efficiency of Blue Gene/P at approximately 4% (for 4ppn) is higher than that of Cray XC40 (∼1.5%). A node of Cray XC40 comprises of 32 cores, each with a peak rating of 36 Giga FLOP/s rating. Hence maximum compute power per node is 1177.6 Giga FLOP/s. On the other hand, a Blue Gene/P node contains 4 cores with a peak rating of $4 \times 3.4 = 13.6$ Giga FLOP/s. Thus, each node of Cray XC40 has approximately 100 times more computational power. However, the interconnect of Cray XC40 is not faster than that of Blue Gene/P in the same ratio.

Note that the dragonfly topology of Cray XC40 appears to be more efficient than the torus topology of Blue Gene/P (see Eqs. (22), (24)), yet the ratio of the speedup is much less than 100. Therefore, for data communication the processors of Cray have to idle longer than those of Blue Gene/P. For the Blue Gene/P, the relatively slower processors do not have to idle as long. This is especially critical for FFT which is communication intensive. Thus, the faster processor and relatively slower interconnect of Cray XC40 result in an overall lower efficiency compared to Blue Gene/P. This is essentially the reason for the lower efficiency of Cray XC40 at 1.5% compared with 4% of the Blue Gene/P. In this sense, the efficiency of hardware depends on the application; for FFT computation, a faster switch is more important than a faster processor.

5.3. Comparison between FFTK and P3DFFT libraries

In this section we describe key features and performance of FFTK library. Another library P3DFFT has similar features. Therefore, it is important to compare the features and performances of the two FFT libraries, which are briefly described below:

1. P3DFFT library has functions to perform Fourier transforms along the three directions. It also has functions to perform Sine transform, Cosine transform or Chebyshev transform only along one direction, and Fourier transforms along the rest two directions [25]. Thus P3DFFT can be used for mixed basis functions like SFF and ChFF (see Section 2 for definitions). In contrast, FFTK can perform, Fourier, Sine, or Cosine transforms along any of the three directions. Thus it can be used for solving problems in FFF, SFF, SSF, and SSS basis. Hence, FFTK has more features than P3DFFT. SSS is very import basis function for physical application, like, reversal studies of Rayleigh–Bénard convection [36]. The FFTK library does not yet support ChFF basis function, but it is under development.
2. In the present paper, we report scaling of FFTK on Cray XC40 for number of processors up to 196 608. The P3DFFT library has been scaled up to 65 536 cores [24] on Cray XT5 machine. To best of our knowledge, no other group has performed detailed scaling studies on FFT on processors more than that for FFTK. Many researchers have performed spectral simulations of fluid flows on a large number of cores, for example, Yeung et al. [42] used 262 144 cores for their 8192^3 simulation, but they did not report scaling results of FFT. Note that, scaling study of FFT is important for the optimized performance of spectral codes.
3. As we show in this paper, the performance of FFT depends critically on the features of processors as well as that of interconnect. Here we perform comparative study of FFTK on BlueGene/P and Cray XC40. One of our findings is that for FFT computations, the efficiency of Cray XC40 is lower than BlueGene/P even though Cray XC40 is more modern HPC system than Blue Gene/P. This is because the efficiency of interconnects has not grown in commensurate with that of processors. We are not aware of similar extensive comparisons and analysis of scaling results for FFT.
4. As shown in Table 2, on BlueGene/P, FFTK and P3DFFT are equally efficient.

After extensive discussion on FFTs, in the next section, we will present the scaling results of the fluid spectral solver.

Table 6
Scaling exponent of the total time for the fluid solver on Blue Gene/P and Cray XC40 for various grids (definition: $T \sim p^{-\gamma}$).

Blue Gene/P		Cray XC40	
Grid size	γ	Grid size	γ
2048^3	0.95 ± 0.05	768^3	0.28 ± 0.15
4096^3	0.8 ± 0.1	1536^3	0.44 ± 0.06
–	–	3072^3	0.68 ± 0.02

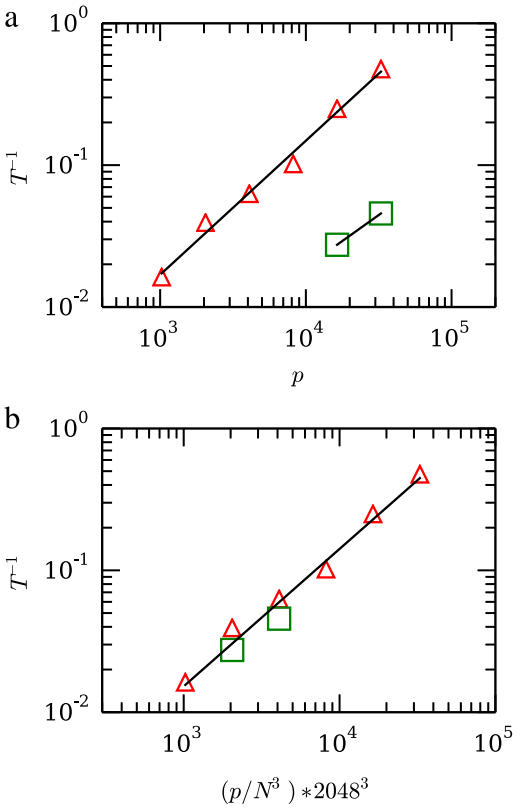


Fig. 7. Scaling of the fluid spectral solver on Blue Gene/P: (a) Plot of T^{-1} vs. p for 2048^3 (red triangle) and 4096^3 (green square) grids exhibits strong scaling. (b) Plot of T^{-1} vs. p/N^3 exhibits weak scaling with an exponent of $\gamma = 0.97 \pm 0.06$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

6. Scaling of fluid spectral solver

We perform high-resolution fluid simulation using spectral method on Blue Gene/P and Cray XC40, i.e., we solve Eqs. (5), (6) on these systems. We assume the flow to be incompressible, and use periodic boundary condition for which FFF basis function is appropriate.

The fluid simulation requires 15 arrays each of size N^3 to store three components of the velocity field in real and Fourier spaces, the force field, the nonlinear term $\mathbf{u} \cdot \nabla \mathbf{u}$, and three temporary arrays [37]. We employ the fourth-order Runge Kutta scheme for time stepping, and dealias the nonlinear terms using the 2/3-rd rule. Each time step requires 36 FFT operations. We refer the reader to Boyd [3], Canuto et al. [5], and Verma et al. [37] for further details and validation tests of the fluid solver. We run our simulations for 10 to 100 time steps depending on the grid size. The time reported in the present section is an average over these many time steps.

The most expensive part of a pseudospectral simulation is the FFT that consumes approximately 70% to 80% of the total time. In addition, a flow solver has many functions including I/O, hence it is tedious to find patterns for the computation and communication components separately. Therefore, in the following discussion we report the scaling of the total time for the flow solvers.

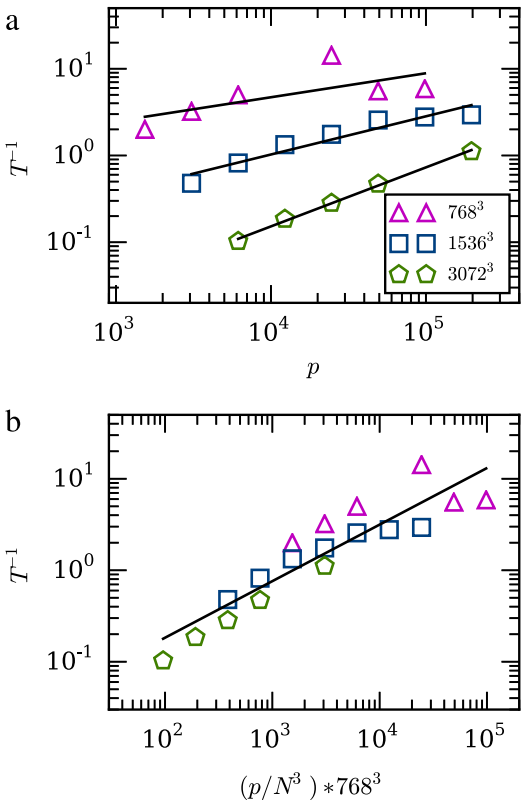


Fig. 8. Scaling of the fluid spectral solver on Cray XC40: (a) Plot of T^{-1} vs. p for 768^3, 1536^3, and 3072^3 grids exhibits strong scaling. (b) Plot of T^{-1} vs. p/N^3 exhibits weak scaling with an exponent $\gamma = 0.62 \pm 0.07$.

6.1. Blue Gene/P

We performed the fluid simulation on 2048^3 and 4096^3 grids using cores ranging from 1024 to 65 536. In Fig. 7(a) we plot the inverse of the total time per iteration vs. p. We observe that $T^{-1} \sim p^{\gamma}$. The exponents γ listed in Table 6 shows that $\gamma = 0.95 \pm 0.05$ and $\gamma = 0.8 \pm 0.1$ for grid sizes of 2048^3 and 4096^3 respectively. This demonstrates the fluid solver exhibits a strong scaling.

The above data nearly collapses into a single curve in the plot of T^{-1} vs. p/N^3, as shown in Fig. 7(b). Hence we conclude that our fluid solver also exhibits weak scaling. The exponent of the weak scaling is $\gamma = 0.97 \pm 0.06$.

6.2. Cray XC40

We performed fluid simulations on grid sizes of 768^3, 1536^3 and 3072^3 grids using cores ranging from 1536 to 196 608. We employ periodic boundary conditions along all the walls. As done for Blue Gene/P supercomputer, we compute the total time taken for each iteration of the solver.

Fig. 8(a) shows that the plots of $T^{-1} \propto p^{\gamma}$, except for 768^3 grid with large p's ($p \geq 98\,000$). Thus fluid solver exhibits a strong scaling, except for 768^3 grid that scales up to $p \lessapprox 98\,000$. We observe that γ for the 768^3, 1536^3 and 3072^3 grids are approximately 0.28, 0.44 and 0.68 respectively. The three curves for the three different grids collapse into a single curve when the X-axis is chosen as p/N^3

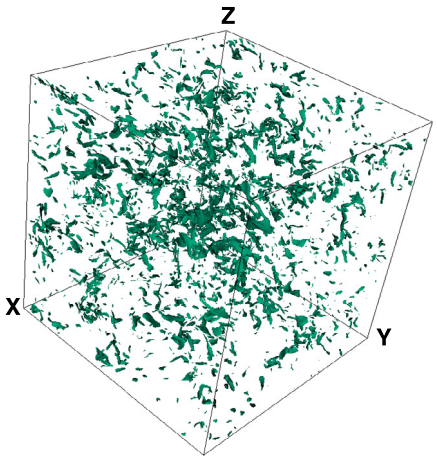


Fig. 9. Isosurface of the contours of constant vorticity $|\nabla \times \mathbf{u}|$ (30% of the maximum value). The figure indicates regions of strong vorticity in the flow.

(except for $p \geq 98\,000$). This result shows a common scaling when the number of cores and data sizes are increased by an equal factor. Thus our fluid solver exhibits both weak and strong scaling nearly up to 196 608 cores of the Cray XC40.

We performed a fluid simulation for a longer time on a 512^3 grid in a periodic box of size $(2\pi)^3$. The Reynolds number of the flow at the steady state is approximately 1100. Fig. 9 exhibits an isosurface of the contours of constant magnitudes of the vorticity under steady state. In the figure we observe intense localized vorticity, as reported in literature.

7. Scaling of RBC spectral solver

We performed high-resolution simulations of Rayleigh–Bénard convection (RBC) by solving Eqs. (13), (14). The fluid is assumed to be contained in a box of unit dimension. Presently, we report the scaling results for FFF (periodic boundary condition) and SFF (free-slip boundary condition) basis functions. Note that in SFF basis, $u_z = \partial_z u_x = \partial_z u_y = 0$ at the top and bottom plates, and periodic along the side walls. The temperatures at the top and bottom plates are assumed to be constant (conducting walls), while at the side walls, the temperature is assumed to be periodic. For the energy spectrum and flux computations, we employ a free-slip boundary condition.

A RBC simulation requires 18 arrays of size N^3. We time step the solver using the fourth-order Runge Kutta scheme that requires 52 FFT per time step. For further details and validation tests of the RBC solver, we refer the reader to Verma et al. [37]. For scaling tests we run our simulations for 10 to 100 time steps. The time reported in this section is an average over the relevant time steps. The results of our simulations on the Blue Gene/P and Cray XC40 supercomputers are as follows:

7.1. Blue Gene/P

We performed RBC simulations on 2048^3 and 4096^3 grids using cores ranging from 8192 to 65 536. In Fig. 10(a, b) we plot T^{-1} vs. p and T^{-1} vs. p/N^3 respectively. Here T is the time taken per step, and p is the number of cores. We observe that $T^{-1} \sim p^{\gamma}$ with the exponent $\gamma = 0.71$ and 0.68 for the 2048^3 and 4096^3 grids respectively (see Table 7). Thus the RBC solver indicating a strong scaling. As exhibited in Fig. 10(b), the data nearly collapses into a single curve when we plot T^{-1} vs. p/N^3, hence exhibiting a weak scaling as well.

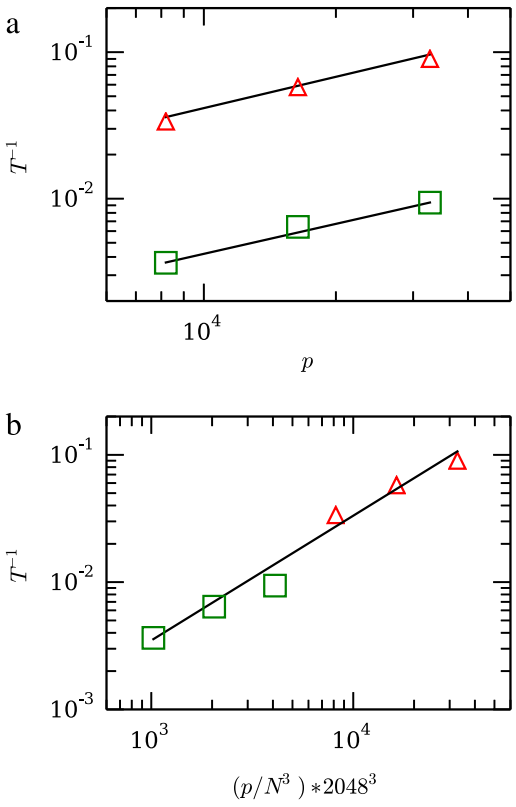


Fig. 10. Scaling of the RBC solver on Blue Gene/P for the FFF basis: (a) Plot of T^{-1} vs. p for 2048^3 (red triangle) and 4096^3 (green square) grids exhibits strong scaling. (b) Plot of T^{-1} vs. p/N^3 exhibits weak scaling with an exponent of $\gamma = 0.68 \pm 0.08$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 7
Scaling exponents of the total time for the RBC solver on Blue Gene/P and Cray XC40 for various grids for the FFF and SFF basis functions (definition: $T \sim p^{-\gamma}$).

Blue Gene/P		Cray XC40	
Grid size	γ	Grid size	γ
FFF			
2048^3	0.71 ± 0.04	768^3	0.49 ± 0.14
4096^3	0.68 ± 0.08	1536^3	0.64 ± 0.04
–	–	3072^3	0.74 ± 0.03
SFF			
–	–	768^3	0.62 ± 0.06
–	–	1536^3	0.74 ± 0.09
–	–	3072^3	0.80 ± 0.05

7.2. Cray XC40

We simulated RBC on 768^3, 1536^3 and 3072^3 using cores ranging from 1536 to 168 608 for FFF and SFF basis. For 3300 iterations of RBC simulation on 2048^3 grid, the total simulation is 1.7×10^5 seconds, thus the time per iteration of RBC on 2048^3 grid is approximately 51.5 s. For the FFF basis, the inverse of the total time plotted in Fig. 11(a) scales as $T^{-1} \sim p^{\gamma}$ with $\gamma = 0.49$, 0.64 and 0.74 for 768^3 (except for $p = 196\,608$), 1536^3 and 3072^3 grids respectively (also see Table 7). The plot indicates a strong scaling for the RBC solver. In Fig. 11(b) we plot T^{-1} vs. p/N^3; here the data collapses into a single curve (see Fig. 11) thus indicating a weak scaling. In Fig. 12 we plot T^{-1} vs. p and T^{-1} vs. p/N^3 for the SFF basis. The exponents listed in Table 7 show that the FFF and SFF scale in a similar manner, with the SFF basis scaling slightly

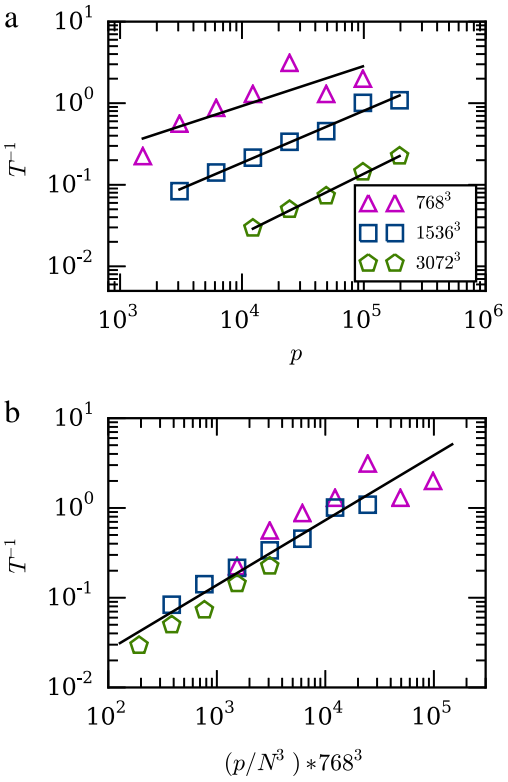


Fig. 11. Scaling of the RBC spectral solver for the FFF basis on Cray XC40: (a) Plot of T^{-1} vs. p for 768^3, 1536^3, and 3072^3 grids exhibits strong scaling. (b) Plot of T^{-1} vs. p/N^3 exhibits weak scaling with an exponent of $\gamma = 0.72 \pm 0.06$.

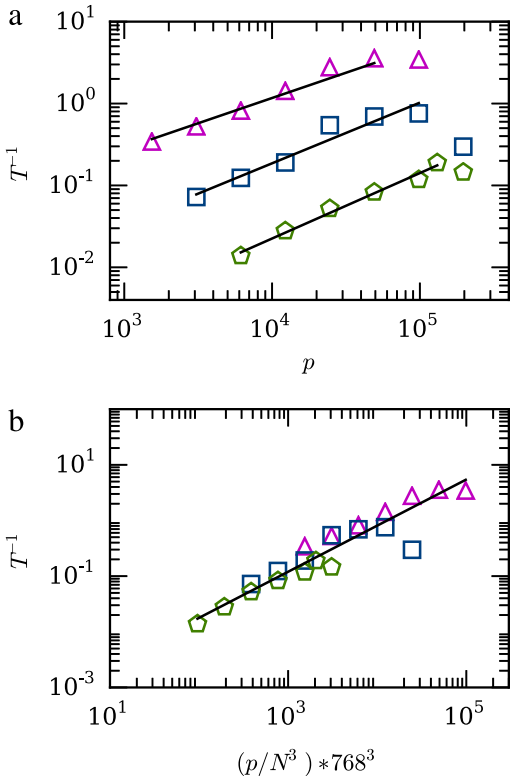


Fig. 12. Scaling of the RBC spectral solver for the SFF basis on Cray XC40: (a) Plot of T^{-1} vs. p for 768^3, 1536^3, and 3072^3 grids exhibits strong scaling. (b) Plot of T^{-1} vs. p/N^3 exhibits weak scaling with an exponent of $\gamma = 0.83 \pm 0.03$.

better than the FFF basis. The plots and the scaling exponents demonstrate that our RBC solver exhibits both strong and weak scaling up to nearly 196 608 cores.

We use Tarang to study energy spectrum and energy flux of RBC and to resolve the long-standing question on the spectral indices. Kumar et al. [18] studied these quantities for 512^3 grid and showed that the turbulent RBC exhibits Kolmogorov's spectrum similar to hydrodynamic turbulence. Here we present a more conclusive spectrum by performing RBC simulations on a 4096^3 grid simulations using 65 536 cores with Rayleigh number $\mathrm{Ra} = 1.1 \times 10^{11}$ and the Prandtl number $\mathrm{Pr} = 1$. The energy spectrum and flux were computed using the steady-state data, which is achieved after around 1000 time steps. Fig. 13 exhibits isocontours of constant temperatures exhibiting hot and cold structures.

In Fig. 14(a) we exhibit the compensated energy spectra $E(k)k^{5/3}$ and $E(k)k^{11/5}$, which correspond to the compensated Kolmogorov spectrum and the Bolgiano–Obukhov spectrum respectively. The constancy of $E(k)k^{5/3}$ in the inertial range indicates that turbulent convection exhibits Kolmogorov spectrum, similar to hydrodynamic turbulence. The energy flux, exhibited in Fig. 14(b), is also constant in the inertial range, again confirming Kolmogorov-like phenomenology for RBC. Thus our DNS helps resolve one of the outstanding questions in turbulent convection.

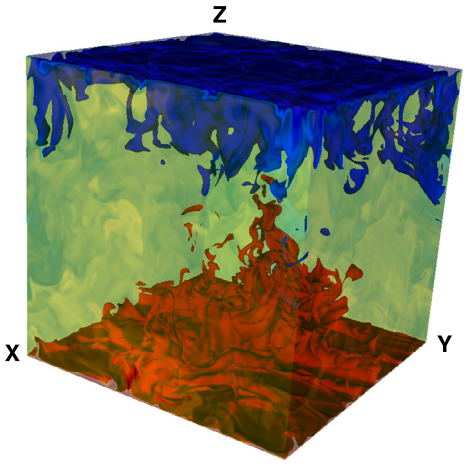


Fig. 13. Isosurface of the contours of constant temperature. The structures with red and blue colours indicate respectively the hot and cold plumes of the flow. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

8. Conclusions

In this paper we perform scaling studies of a FFT library, FFTK, and a pseudospectral code Tarang on two different HPC supercomputers—Blue Gene/P (Shaheen I) and Cray XC40 (Shaheen II) of KAUST. We vary grids from 768^3 to 8192^3 on cores ranging from 1024 to 196608. The number of cores used for FFTK and Tarang are one of the largest in this area of research. We also remark that on Blue Gene/P, the efficiency of FFTK is similar to that of P3DFFT. The main results presented in this paper are as follows:

1. We analyse the computation and communication times for FFTK. We observe that the computation time $T_{\mathrm{comp}} \sim p^{-1}$ where p is the number of cores, while the communication time $T_{\mathrm{comm}} \sim n^{-\gamma_2}$ where n is the number of nodes.

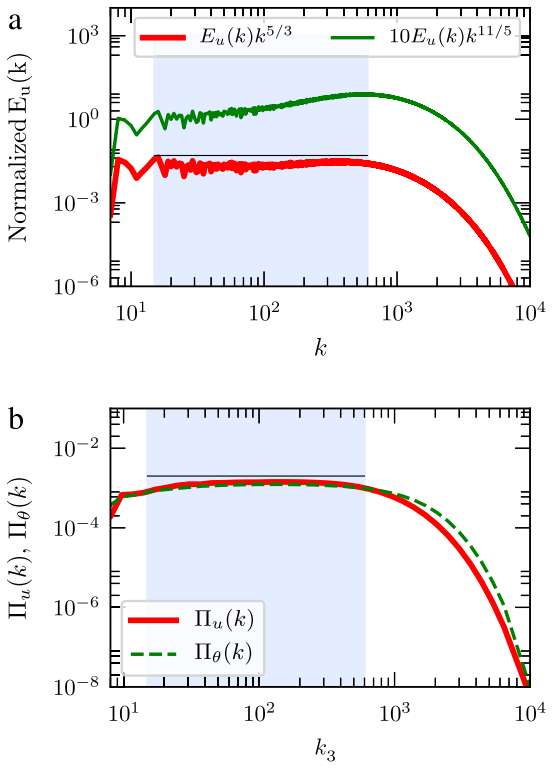


Fig. 14. For RBC simulation on 4096^3 grid with Pr $= 1$ and Ra $= 1.1 \times 10^{11}$: (a) Plots of the normalized kinetic energy spectra $E(k)k^{5/3}$ (Kolmogorov–Obukhov, red curve) and $E(k)k^{11/5}$ (Bolgiano–Obukhov, green curve). Flatness of $E(k)k^{5/3}$ indicates Kolmogorov-like phenomenology for RBC. (b) Plot of kinetic energy flux $\Pi_u(k)$ and entropy flux $\Pi_\theta(k)$, both exhibit a constant flux in the inertial range. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
Source: From Verma et al. [38]. Reprinted under a CC BY licence (http://creativecommons.org/licenses/by/3.0/).

2. Regarding FFTK, for Blue Gene/P, the communication time is comparable to the computation time due the slower core and faster switch. In Cray XC40 however the communication dominates computation due to faster cores. For FFTK, the total time scales as $T \sim p^{-\gamma}$ with γ ranging from 0.76 to 0.96 for Blue Gene/P. For Cray XC40, γ lies between 0.43 to 0.73. In Section 5.2 we argue that the dragonfly topology of Cray XC40 is more efficient than torus topology of Blue Gene/P. Yet, the speedup of interconnect for Cray XC40 is not much higher than that for Blue Gene/P.
3. Cray XC40 exhibits lower efficiency (∼1.5%) than Blue Gene/P (∼4%). This is in-spite of the fact that ratio of the per-node compute power of Cray XC40 and Blue Gene/P is approximately 100. The relative loss of efficiency for XC40 is because the efficiency of its interconnect has not scaled in commensurate with that of its processor. For communication intensive application like FFT, the speed of interconnect is more important than that of the compute cores.
   The above observation indicates that the performance of a HPC system depends on the application. We need to be cautious about this, and factor into account the speed of the processor, interconnect, memory, and input–output. In future, we plan to do an extensive study of these factors for FFT.
4. The fluid solver of Tarang exhibits weak and strong scaling on both the supercomputers. The exponent γ for Blue Gene/P varies from 0.8 to 0.95, but it ranges from 0.28 to 0.68 for Cray XC40.
5. The solver for Rayleigh–Bénard convection also shows weak and strong scaling on both the supercomputers. The corresponding γ for Blue Gene/P ranges from 0.68 to 0.71, but it lies between 0.49 to 0.80 for Cray XC40.
6. The scaling of FFTK, and fluid and RBC solvers scale quite well up to 196 608 cores of Cray XC40. We however observe saturation at 98 000 cores for 768^3 grid, possibly due to smaller data size. To best of our knowledge, there is no such detailed scaling study on FFT up to these many processors.
7. The scaling of different basis functions (e.g., FFF and SFF) are similar. However the performance in the SFF basis is slightly better than that in the FFF basis.

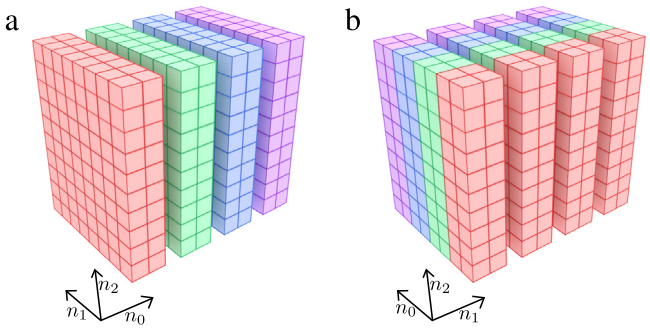


Fig. A.15. Data division for FFT with transpose: (a) Complex data of size $n_0 \times n_1 \times (n_2/2+1)$ in Fourier space. (b) Real data of size $n_1 \times n_0 \times n_2$ in real space. Note that the axes n_0 and n_1 are exchanged during the transpose.

Thus, FFTK and Tarang scale nearly up to 196 608 cores. Thus these codes are capable of simulating turbulence at very high-resolution. FFTK would also be useful for other applications, e.g., image processing, density functional theory, etc.

Acknowledgements

We thank Shashwat Bhattacharya for his valuable help on plotting. Our numerical simulations were performed at Cray XC40 *Shaheen II* at KAUST supercomputing laboratory (KSL), Saudi Arabia. We thank KAUST scientists for the kind support while performing our simulations. This work was supported by research grants SERB/F/3279 from Science and Engineering Research Board India, research grant K1052 and baseline research grant BAS/1/1349-01-01 from KAUST, and R&D/20130307 from IIT Kanpur; and by KAUST baseline research funds of Ravi Samtaney.

Appendix. Transpose-free fast Fourier transform

In this appendix we describe how we avoid local transpose in FFTK to save communication time. For simplicity we illustrate this procedure using slab decomposition with complex data of size $n_0 \times n_1 \times (n_2/2 + 1)$. The corresponding real space data is of the size $n_0 \times n_1 \times n_2$.

The usual FFT implementation involving transpose is illustrated below. The complex data is divided along n_0. If there are p processors, then each processor has $(n_0/p) \times n_1 \times (n_2/2 + 1)$ complex data. See Fig. A.15(a) for an illustration. A typical inverse transform involves three steps:

1. Perform two-dimensional inverse transforms (complex-to-real `c2r`) on n_0/p planes each having data of size $n_1 \times (n_2/2+1)$.

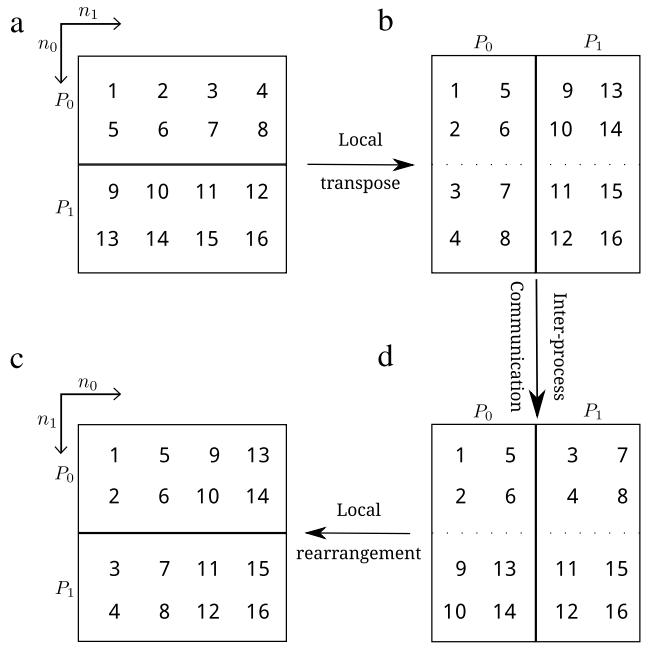


Fig. A.16. The standard transpose procedure in during a FFT. It involves two local transposes and a `MPI_Alltoall`.

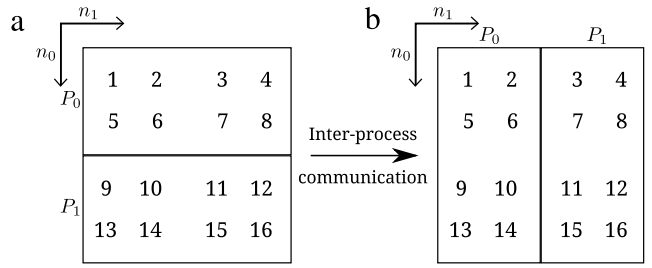


Fig. A.17. Transpose using strided `MPI_Isend/MPI_Recv` that does not require a local transpose. This is employed in the transpose-free FFT.

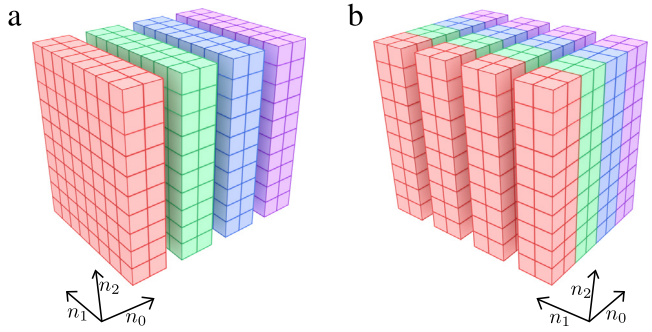


Fig. A.18. Data division for a transpose-free FFT: (a) Complex data of size $n_0 \times n_1 \times (n_2/2+1)$ in Fourier space. (b) Real data of size $n_0 \times n_1 \times n_2$ in real space. Note that there is no exchange of axes here. Compare it with Fig. A.15.

2. Perform transpose on the array along n_0-n_1 axis. This operation involves local transpose and `MPI_Alltoall` operations (to be described below).
3. After the data transfer, the data along the n_0 axis resides in the respective processors. Now in each processor, we perform one-dimensional real-to-real (`r2r`) inverse transforms on $(n_1/p) \times n_2$ column each having data of size n_0.

In Fig. A.16, we illustrate this transpose operation using a simple example involving 16 data points and 2 processors. In the first step, the local data is transposed, as illustrated in Fig. A.16(a, b). In the example, in process P_0, the data [[1,2,3,4], [5,6,7,9]] gets transformed to [[1,5],[2,6],[3,7],[4,8]]. After the local transpose, chunks of data are transferred among the processors using `MPI_Alltoall` function (Figs. A.16(b) to A.16(c)). In this process, the blocks [[3,7],[4,8]] and [[9,13],[10,14]] are exchanged between P_0 and P_1. After this data transfer, there is another local transpose that transforms the data from Figs. A.16(c) to A.16(d).

This complete operation is called "transpose" because it is similar to matrix transpose. After transpose, we are ready for FFT operations along the n_0 axis. Note that the data along the rows of Fig. A.16(d) are consecutive, that makes it convenient for the FFT operation. We remark that the popular FFTW library employs the above procedure involving transpose.

An advantage of the above scheme is that the FFT is performed on consecutive data sets that minimizes cache misses. However, the aforementioned FFT involves two local transposes, which are quite expensive. To avoid this, we have devised a FFT which is based on transpose-free data transfer. This process is described below.

In the transpose-free procedure, we replace the transpose operations (item 2 in the above list) with transpose-free interprocessor communication. We employ `MPI_Type_vector` and `MPI_Type_create_resized` to select strided data to be exchanged among the processors. We illustrate the communication process in Fig. A.17. Here, the data block [[3, 4], [7, 8]] is transferred from P_0 to P_1, and the data block [[9, 10], [13, 14]] is transferred from P_1 to P_0 using MPI functions `MPI_Isend/MPI_Recv` or `MPI_All_to_all`. Note that the data to be transferred are not consecutive, hence we need the MPI functions such as `MPI_Type_vector` and `MPI_Type_create_resized` to create strided-data set. The data structure before and after the interprocess communication are shown in Fig. A.18(a, b) respectively. Here the data axes are not exchanged, however the columnar data along the n_0 axis are not contiguous. For example, the data of column [1,5,9,13] of Fig. A.17(b) are staggered by 1. As a result, FFT along the n_1 axis involves consecutive data, but not along n_0. The latter FFT however can be performed using strided FFTW functions.

Now let us briefly compare the performances of the two methods. The transpose-free scheme avoids local transpose, hence it saves some communication time compared to the usual FFT. A flip side of the transpose-free scheme is that it needs strided FFT that is prone to cache misses because the data are not contagious. Note, however, that intelligent cache prefetch algorithms [1] could helps in efficient implementation of strided FFT.

To compare the efficiencies of the aforementioned FFT schemes, we performed FFTs using both the schemes. Since FFTW involves local transposes, we use this as one of the benchmark programs. We wrote a transpose-free FFT function as the other benchmark program. The tests were performed on IBM BlueGene/P (Shaheen I) of KAUST for a pair of forward and inverse transforms on 1024^3 and 2048^3 grids.

In Fig. A.19(a, b) we present the results for the 1024^3 and 2048^3 grids. In the figure the time taken by FFTW and transpose-free FFT are shown by red circle and black diamonds respectively. We observe that the transpose-free FFT is 10% to 16% more efficient for 1024^3 data, and 5% to 14% more efficient for 2048^3 data. The gain by the transpose-free FFT decreases as the number of processors are increased. The difference in time is a sum of two factors: (a) gain by avoidance of local transpose, and (b) loss due to strided FFT. We

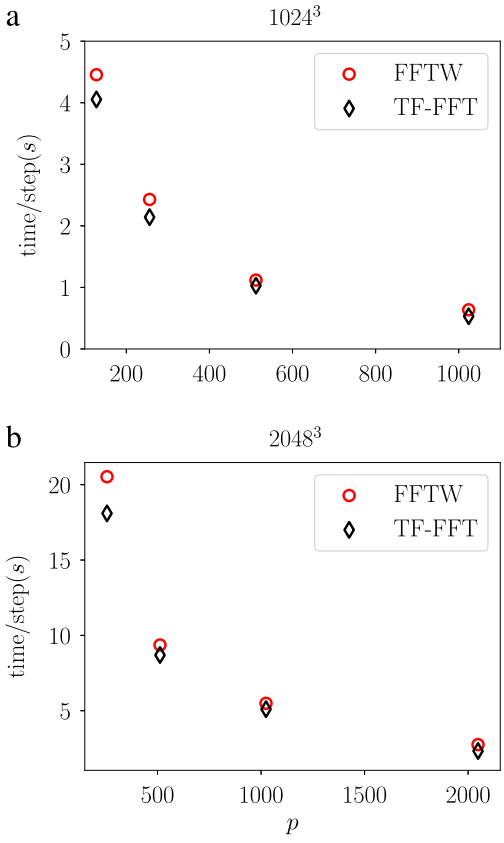


Fig. A.19. Comparison between FFTs with transpose and without transpose for (a) 1024^3 grid, and (b) 2048^3 grid. Transpose-free FFT is marginally superior than the one with transpose. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

need a more detailed diagnostics to analyse the two algorithms. For example, we need to separately compute the computation and communication time. Also, it will be useful to estimate the time for the collection of the strided data, as well as that of the strided FFT. These works will be performed in future.

The FFT operations in FFTK, which has pencil decomposition, in transpose-free. The only difference between the slab-based FFT described in this appendix and the pencil-based FFT is that the data exchange in pencil-based FFT takes place among the respective communicators. For example, among the `MPI_COM_ROW` and `MPI_COM_COL` communicators of Fig. 2.

References

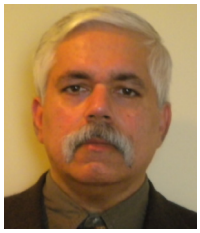
[1] S.G. Berg, Cache Prefetching. Technical Report, UW-CSE, 2004.
[2] Blitz++, The open source meta-template library, www.blitz.sourceforge.net.
[3] J.P. Boyd, Chebyshev and Fourier Spectral Methods, Dover Publishers, New York, 2001.
[4] W.J. Buchanan, The Handbook of Data Communications and Networks, Springer, 2004.
[5] C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zhang, Spectral Methods in Fluid Turbulence, Springer-Verlag, Berlin, 1988.
[6] A. Chan, P. Balaji, W. Gropp, R. Thakur, Communication analysis of parallel 3D FFT for flat cartesian meshes on large blue gene systems, in: High Performance Computing - HiPC, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 350–364.
[7] J.W. Cooley, J.W. Tukey, An algorithm for the machine calculation of complex fourier series, Math. Comp. 19 (90) (1965) 297–301.
[8] K. Czechowski, C. Battaglino, C. McClanahan, K. Iyer, P.K. Yeung, R. Vuduc, On the communication complexity of 3D FFTs and its implications for Exascale, in: The 26th ACM International Conference, ACM, New York, USA, 2012, pp. 205–214.
[9] V. Dallas, S. Fauve, A. Alexakis, Statistical equilibria of large scales in dissipative hydrodynamic turbulence, Phys. Rev. Lett. 115 (20) (2015) 204501.
[10] D.A. Donzis, K.R. Sreenivasan, The bottleneck effect and the Kolmogorov constant in isotropic turbulence, J. Fluid Mech. 657 (2010) 171.
[11] D.A. Donzis, K.R. Sreenivasan, P.K. Yeung, The batchelor spectrum for mixing of passive scalars in isotropic turbulence, Flow Turbul. Combust. 85 (3–4) (2010) 549–566.
[12] D.A. Donzis, P.K. Yeung, D. Pekurovsky, Turbulence simulations on $\mathcal{O}(10^4)$ processors, in: Proc TeraGrid, 2008.
[13] D.A. Donzis, P.K. Yeung, K.R. Sreenivasan, Dissipation and enstrophy in isotropic turbulence: Resolution effects and scaling in direct numerical simulations, Phys. Fluids 20 (4) (2008) 045108.
[14] FFTW, The open source fast Fourier transform library, http://www.fftw.org.
[15] M. Frigo, S.G. Johnson, The design and implementation of FFTW3, Proc. IEEE 93 (2) (2005) 216–231 Special issue on Program Generation, Optimization, and Platform Adaptation.
[16] B. Hadri, S. Kortas, S. Feki, R. Khurram, G. Newby, Overview of the KAUST's Cray X40 System–Shaheen II, in: CUG2015 Proceedings, 2015.
[17] Y. Kaneda, T. Ishihara, M. Yokokawa, K. Itakura, A. Uno, Energy dissipation rate and energy spectrum in high resolution direct numerical simulations of turbulence in a periodic box, Phys. Fluids 15 (2) (2003) L21.
[18] A. Kumar, A.G. Chatterjee, M.K. Verma, Energy spectrum of buoyancy-driven turbulence, Phys. Rev. E 90 (2) (2014) 023016.
[19] R. Kumar, M.K. Verma, R. Samtaney, Energy transfers and magnetic energy growth in small-scale dynamo, Europhys. Lett. 104 (5) (2014) 54001.
[20] P.D. Mininni, D.L. Rosenberg, R. Reddy, A.G. Pouquet, A hybrid MPI-OpenMP scheme for scalable parallel pseudospectral computations for fluid turbulence, Parallel Comput. 37 (6–7) (2011) 316–326.
[21] D. Nath, A. Pandey, A. Kumar, M.K. Verma, Near isotropic behavior of turbulent thermal convection, Phys. Rev. Fluids 1 (2016) 064302.
[22] S.A. Orszag, Comparison of pseudospectral and spectral approximation, Stud. Appl. Math. 51 (3) (1972) 253–259.
[23] A. Pandey, M.K. Verma, P.K. Mishra, Scaling of heat flux and energy spectrum for very large Prandtl number convection, Phys. Rev. E 89 (2014) 023006.
[24] D. Pekurovsky, P3DFFT: a framework for parallel computations of Fourier transforms in three dimensions, SIAM J. Sci. Comput. 34 (4) (2012) C192–C209.
[25] D. Pekurovsky, P3DFFT: User Guide (https://www.p3dfft.net/user-guide).
[26] H.K. Pharasi, K. Kumar, Oscillatory instability and fluid patterns in low-Prandtl-number Rayleigh-Bénard convection with uniform rotation, Phys. Fluids 25 (10) (2013) 104105–104121.
[27] M. Pippig, D. Potts, Scaling parallel fast fourier transform on bluegene/p. Jülich BlueGene/P Scaling Workshop, 2010.
[28] K.S. Reddy, M.K. Verma, Strong anisotropy in quasi-static magnetohydrodynamic turbulence for high interaction parameters, Phys. Fluids 26 (2014) 025109.
[29] D.F. Richards, J.N. Glosli, B. Chan, M.R. Dorr, E.W. Draeger, J.-L. Fattebert, W.D. Krauss, T. Spelce, F.H. Streitz, M.P. Surh, J.A. Gunnels, Beyond homogeneous decomposition: Scaling long-range forces on massively parallel systems, in: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09, ACM, New York, NY, USA, 2009, pp. 60:1–60:12.
[30] C. Rorai, P.D. Mininni, A.G. Pouquet, Stably stratified turbulence in the presence of large-scale forcing, Phys. Rev. E 92 (1) (2015) 013003.
[31] D.L. Rosenberg, A.G. Pouquet, R. Marino, P.D. Mininni, Evidence for Bolgiano-Obukhov scaling in rotating stratified turbulence using high-resolution direct numerical simulations, Phys. Fluids 27 (5) (2015) 055105.
[32] M.K. Sharma, A. Kumar, M.K. Verma, S. Chakraborty, Structures and energy spectrum of strongly rotating decaying turbulence, J. Turbul. (2017) Under review in.
[33] T. Sutou, K. Tamura, Y. Mori, H. Kitakami, Design and implementation of parallel modified prefix span method, in: High Performance Computing, 5th International Symposium, ISHPC 2003 Tokyo-Odaiba, Japan, 2003.
[34] T.L. Veldhuizen, Arrays in Blitz++, in: Proceedings of the Second International Symposium on Computing in Object-Oriented Parallel Environments, ISCOPE '98, Springer-Verlag, London, UK, UK, 1998, pp. 223–230.
[35] M.K. Verma, Statistical theory of magnetohydrodynamic turbulence: recent results, Phys. Rep. 401 (5) (2004) 229–380.
[36] M.K. Verma, S.C. Ambhire, A. Pandey, Flow reversals in turbulent convection with free-slip walls, Phys. Fluids 27 (4) (2015) 047102.
[37] M.K. Verma, A.G. Chatterjee, K.S. Reddy, R.K. Yadav, S. Paul, M. Chandra, R. Samtaney, Benchmarking and scaling studies of a pseudospectral code Tarang for turbulence simulations, Pramana 81 (2013) 617–629.
[38] M.K. Verma, A. Kumar, A. Pandey, Phenomenology of buoyancy-driven turbulence: recent results, New J. Phys. 19 (2017) 025012.

[39] M.K. Verma, P.K. Mishra, A. Pandey, S. Paul, Scalings of field correlations and heat transport in turbulent convection, Phys. Rev. E 85 (1) (2012) 016310.
[40] P.K. Yeung, D.A. Donzis, K.R. Sreenivasan, High-Reynolds-number simulation of turbulent mixing, Phys. Fluids 17 (8) (2005) 081703.
[41] P.K. Yeung, K.R. Sreenivasan, Spectrum of passive scalars of high molecular diffusivity in turbulent mixing, J. Fluid Mech. 716 (2013) R14.
[42] P.K. Yeung, X.M. Zhai, K.R. Sreenivasan, Extreme events in computational turbulence, Proc. Natl. Acad. Sci. USA 112 (41) (2015) 12633.
[43] 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: ACM/IEEE Conference, IEEE, 2002.

Anando G. Chatterjee is pursuing Ph.D. from Department of Physics, Indian Institute of Technology Kanpur, India in high-performance computing. His main scientific interests are: high-performance computing, distributed and parallel programming and GPGPU computing.

Mahendra K. Verma received his Ph.D. degree from University of Maryland, College Park. Presently he is a Professor at the Physics Department of IIT Kanpur, India. He is a recipient of Swarnajayanti fellowship, and author of an introductory book "Introduction to Mechanics". Mahendra's research interests include turbulence, nonlinear dynamics, and high-performance computing. He is a lead developer of the spectral code TARANG that can simulate variety of fluid flows including fluids, magnetohydrodynamics, thermal convection.

Abhishek Kumar received Ph.D. degree from Department of Physics, Indian Institute of Technology Kanpur, India in the field of fluid turbulence in January 2017. He currently works at Department of Physics, Indian Institute of Technology Kanpur as a postdoctoral researcher. His primary scientific interests are high-performance computing and the numerical study of Rayleigh–Bénard convection, stably stratified turbulence, and geophysical fluid dynamics.



Ravi Samtaney is a Professor of Mechanical Engineering within the PSE Division, with a secondary appointment in applied mathematics at the KAUST in Saudi Arabia. He also serves as the Associate Dean of the PSE Division. Prior to joining KAUST, Prof. Samtaney was a research physicist at Princeton Plasma Physics Laboratory, Princeton University. Previously, he held the position of senior research associate in the Aeronautics, and Applied & Computational Mathematics at Caltech. His professional experience includes working as a research scientist at NASA Ames Research Center. Prof. Samtaney holds a Ph.D. from Rutgers University in Mechanical & Aerospace Engineering. Prof. Samtaney's main area of expertise is in computational physics of fluids and plasmas, numerical methods, and high-performance computing.

Bilel Hadri is a computational scientist at the Supercomputing Lab at KAUST since July 2013. He is leading efforts in benchmarking and performance optimization and helping in coordinating strategic efforts for systems procurements, upgrades and provides regular training to users. He received his Master in Applied Mathematics and his PhD in Computer Science from the University of Houston in 2008. He joined the National Institute for Computational Science at Oak Ridge National Lab as a computational scientist in December 2009 following a Postdoctoral Position in June 2008 at the University of Tennessee Innovative Computing Laboratory lead by Dr. Jack Dongarra. His expertise area includes performance analysis tuning and optimization, System Utilization Analysis, Monitoring and Library Tracking Usage, Porting and Optimizing Scientific Applications on Accelerator Architectures (NVIDIA GPUs, Intel Xeon Phi), Linear Algebra, Numerical Analysis and Multicore Algorithms.

Rooh Khurram is working as a Staff Scientist at KAUST Supercomputer Lab at King Abdullah University of Science and Technology (KAUST) in Saudi Arabia. He has conducted research in finite element methods, high performance computing, multiscale methods, fluid structure interaction, detached eddy simulations, in-flight icing, and computational wind engineering. He has over 20 years of industrial and academic experience in CFD. He specializes in developing custom made computational codes for industrial and academic applications. His industrial collaborators include: Boeing, Bombardier, Bell Helicopter, and Newmerical Technologies Inc. Before joining KAUST in 2012, Rooh worked at the CFD Lab at McGill University and the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign. Rooh received his Ph.D. from the University of Illinois at Chicago in 2005. In addition to a Ph.D. in Civil Engineering, Rooh has degrees in Mechanical Engineering, Nuclear Engineering, and Aerospace Engineering.