Progress in NUCLEAR SCIENCE and TECHNOLOGY, Vol. 2, pp.620-627 (2011)

ARTICLE

Performance Evaluations of Advanced Massively Parallel Platforms Based on Gyrokinetic Toroidal Five-Dimensional Eulerian Code GT5D

Yasuhiro IDOMURA* and Sébastien JOLLIET

Japan Atomic Energy Agency, Higashi-Ueno 6-9-3, Taitou, Tokyo 110-0015,Japan

A gyrokinetic toroidal five dimensional Eulerian code GT5D is ported on six advanced massively parallel platforms and comprehensive benchmark tests are performed. A parallelisation technique based on physical properties of the gyrokinetic equation is presented. By extending the parallelisation technique with a hybrid parallel model, the scalability of the code is improved on platforms with multi-core processors. In the benchmark tests, a good scalability is confirmed up to several thousands cores on every platforms, and the maximum sustained performance of ~ 19.4 Tflops is achieved using 16,384 cores of BX900.

KEYWORDS: fusion plasmas, gyrokinetc simulation, Eulerian simulation, hybrid parallel model, MPI, OpenMP, SMP

I. Introduction

Turbulence is a common phenomenon observed in most fluids, and is characterised by its chaotic behaviour and turbulent mixing leading to much larger transport than the molecular diffusion. Such turbulent transport is often observed also in a fusion plasma. Although a theoretical model for the transport due to Coulomb collisions was established, the transport levels observed in the experiment normally exceeds predictions by the collisional transport model, and a dominating cause is considered as plasma turbulence driven by micro-instabilities, which are characterised by microscopic scale lengths typically given by the Larmor radius. Since the confinement performance is an important factor in determining the size and cost of a fusion reactor, the understanding of plasma turbulence is one of critical issues in fusion science. Because of its basic properties such as large degrees of freedom, strong nonlinearity, and sensitivity to the initial condition, a theoretical treatment of turbulence is an extremely difficult issue as in neutral fluid turbulence. In addition, plasma turbulence shows more complex properties such as interactions of multiple fluids (an electron fluid and other ion fluids) through electromagnetic fields, highly anisotropic turbulent structures due to the confinement field (see **Fig. 1**), forcing due to multiple micro-instabilities over wide spectral ranges, and kinetic effects such as wave-particle resonant interactions. Because of these complexity, a direct numerical simulation (DNS) is becoming important not only as a complementary approach to obtain physical understanding but also as a tool for predicting turbulent spectrum and transport in the experiment. However, the first principle model of fusion plasmas is given by a five dimensional (5D) gyrokinetic model[1] (see **Fig. 2**), in which the minimum scale length of turbulent fields is given by the Larmor radius ($\sim$mm). Because of this huge requirement on computational resources, DNS of experimentally relevant plasma turbulence was prohibitive before developments of modern supercomputers. The rapid progress of computational power extremely enhanced capabilities of gyrokinetic simulations,[2,3] which have been established as an essential tool for studying tokamak turbulent transport. Recent gyrokinetic simulations have provided qualitative understanding of rich physics behind tokamak turbulent transport as well as quantitative estimations for key engineering issues such as the plasma size scaling of turbulent transport.

However, capabilities of present day supercomputers are still not enough for simulating the next generation fusion reactor such as the international thermonuclear experimental reactor (ITER)[4] and DEMO reactors, which are several times larger than existing fusion devices. Therefore, it is very important to optimise gyrokinetic simulations on advanced massively parallel platforms. In the last few years, massively parallel platforms have had qualitative changes by introducing multi-core processors and quantitative improvements in the number of processing cores and available memory sizes. However, it is not so easy to extract the maximum computing power from such platforms, and sophisticated parallelisation methods are needed. In this work, we present novel parallelisation techniques based on the physical properties of the gyrokinetic model, which was implemented on a gyrokinetic toroidal 5D Eulerian code (GT5D),[5] and discuss its hybrid parallel extension for multi-core processors. The code is ported on six advanced massively parallel platforms, and their performances and computational properties are discussed. Through the benchmark tests, key issues, which become important in the parallelisation beyond $\sim 10^4$ cores, are discussed.

The manuscript is organised as follows. Section II presents the gyrokinetic model used in GT5D. Section III summarises numerical approaches. Section IV describes parallel imple-

*Corresponding author, E-mail: idomura.yasuhiro@jaea.go.jp

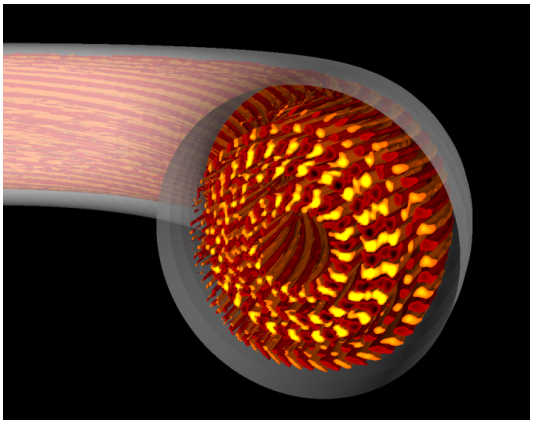

Fig. 1 The turbulent electrostatic potential observed in an ion turbulence simulation using GT5D. The turbulent fields typically show anisotropic structures aligned to the confinement magnetic field lines.

mentations and porting issues on BX900,[6] SR16000,[7] FX1,[8] Altix3700Bx2,[9] T2K(HA8000),[10] and BlueGene/P.[11] Section V gives the benchmark results. Finally, a summary is given in Section VI.

II. Physical Model

We consider the electrostatic ion turbulence described by gyrokinetic ions and adiabatic electrons in an axisymmetric (in ζ see **Fig. 3**) toroidal configuration. GT5D is based on the modern gyrokinetic theory,[1] in which the gyrokinetic equation is simply given as the Liouville equation,

$$\begin{aligned}\frac{Df}{Dt} &\equiv \frac{\partial f}{\partial t} + \{f, H\} \\ &= \frac{\partial f}{\partial t} + \dot{\mathbf{R}} \cdot \frac{\partial f}{\partial \mathbf{R}} + \dot{v}_\parallel \frac{\partial f}{\partial v_\parallel} \\ &= 0, \end{aligned} \tag{1}$$

using the gyro-centre Hamiltonian,

$$H = H_0 + H_1, \tag{2}$$

$$H_0 = \frac{1}{2} m_i v_\parallel^2 + \mu B, \tag{3}$$

$$H_1 = e\langle\phi\rangle_\alpha, \tag{4}$$

and the gyrokinetic Poisson bracket operator,

$$\begin{aligned}\{F, G\} \equiv & \frac{\Omega_i}{B}\left(\frac{\partial F}{\partial \alpha}\frac{\partial G}{\partial \mu} - \frac{\partial F}{\partial \mu}\frac{\partial G}{\partial \alpha}\right) \\ & + \frac{\mathbf{B}^*}{m_i B_\parallel^*} \cdot \left(\nabla F \frac{\partial G}{\partial v_\parallel} - \frac{\partial F}{\partial v_\parallel}\nabla G\right) \\ & - \frac{c}{eB_\parallel^*}\mathbf{b} \cdot \nabla F \times \nabla G, \end{aligned} \tag{5}$$

in the gyro-centre coordinates $\mathbf{Z} = (t; \mathbf{R}, v_\parallel, \mu, \alpha)$. Here, $\mathbf{R}$ is a position of the guiding centre, $v_\parallel$ is the parallel velocity, μ is the magnetic moment, α is the gyro-phase angle, $\mathbf{B} = B\mathbf{b}$ is the magnetic field, $\mathbf{b}$ is the unit vector in the parallel direction, m_i and e are the mass and charge of ions, respectively, c is the velocity of light, $\Omega_i = (eB)/(m_i c)$ is the cyclotron frequency, $B_\parallel^* = \mathbf{b}\cdot\mathbf{B}^*$ is a parallel component of $\mathbf{B}^* = \mathbf{B} + (Bv_\parallel/\Omega_i)\nabla\times\mathbf{b}$, ϕ is the electrostatic potential, and the gyro-averaging operator is defined as $\langle\cdot\rangle_\alpha \equiv \oint \cdot d\alpha/2\pi$ (see **Fig. 2**). The nonlinear characteristics $\dot{\mathbf{Z}} = \{\mathbf{Z}, H\}$ are given as

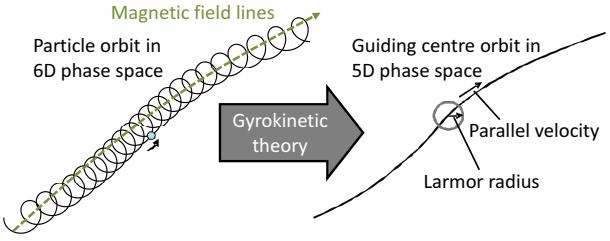

Fig. 2 Reduction of particle orbit from 6D to 5D by eliminating fast gyro-motion.

$$\begin{aligned}\dot{\mathbf{R}} &= v_\parallel \mathbf{b} \\ &\quad + \frac{c}{eB_\parallel^*}\mathbf{b}\times\left(e\nabla\langle\phi\rangle_\alpha + m_i v_\parallel^2 \mathbf{b}\cdot\nabla\mathbf{b} + \mu\nabla B\right), \end{aligned} \tag{6}$$

$$\dot{v}_\parallel = -\frac{\mathbf{B}^*}{m_i B_\parallel^*}\cdot(e\nabla\langle\phi\rangle_\alpha + \mu\nabla B). \tag{7}$$

By using the phase space volume conservation $\nabla\cdot(\mathcal{J}\dot{\mathbf{R}}) + \partial_{v_\parallel}(\mathcal{J}\dot{v}_\parallel) = 0$, the gyrokinetic equation (1) yields its conservative form,

$$\frac{\partial \mathcal{J} f}{\partial t} + \nabla\cdot(\mathcal{J}\dot{\mathbf{R}} f) + \frac{\partial}{\partial v_\parallel}(\mathcal{J}\dot{v}_\parallel f) = 0. \tag{8}$$

where $\mathcal{J} = m_i^2 B_\parallel^*$ is the Jacobian of the gyro-centre coordinates. By adding a collision term $C(f)$ and a source term S_{src}, a conservative gyrokinetic equation used in GT5D is written as,

$$\begin{aligned}\frac{\partial \mathcal{J} f}{\partial t} + \nabla\cdot(\mathcal{J}\dot{\mathbf{R}} f) + \frac{\partial}{\partial v_\parallel}(\mathcal{J}\dot{v}_\parallel f) \\ = \mathcal{J}C(f) + \mathcal{J}S_{src}. \end{aligned} \tag{9}$$

Ion-ion collisions are modelled by the linear Fokker-Planck collision operator,[12]

$$\begin{aligned}C(f) &= \frac{\partial}{\partial s}\left(\nu_{\perp 1} v^2 f\right) + \frac{\partial}{\partial u}\left(\nu_{\parallel 1} u f\right) \\ &\quad + \frac{1}{2}\frac{\partial^2}{\partial s^2}\left(\nu_{\perp 2} v^4 f\right) + \frac{1}{2}\frac{\partial^2}{\partial u^2}\left(\nu_{\parallel 2} v^2 f\right) \\ &\quad + \frac{\partial^2}{\partial s \partial u}\left(\nu_{\parallel\perp} v^3 f\right) + C_F, \end{aligned} \tag{10}$$

where $\nu_{\perp 1}, \nu_{\parallel 1}, \nu_{\perp 2}, \nu_{\parallel 2}$, and $\nu_{\parallel\perp}$ are collision frequencies, C_F is a field particle operator, $s = 2\mu B/m_i$ and $u = v_\parallel - U_\parallel$ are a moving frame with respect to the parallel flow velocity $U_\parallel$, and $v^2 = u^2 + s$.

The self-consistency is imposed by the gyrokinetic Poisson

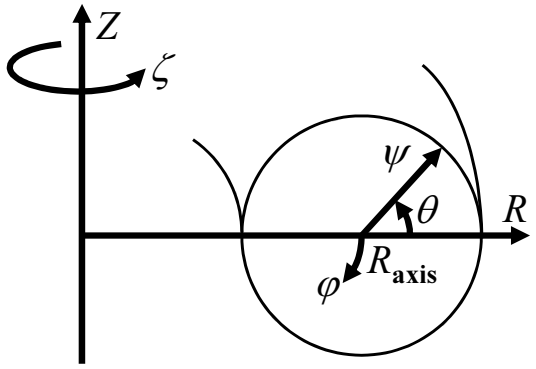

Fig. 3 The coordinate systems used in GT5D.

equation,

$$-\nabla_\perp \cdot \frac{\rho_{ti}^2}{\lambda_{Di}^2}\nabla_\perp \phi + \frac{1}{\lambda_{De}^2}\left(\phi - \langle\phi\rangle_f\right) = 4\pi e\left[\int f\delta([\mathbf{R}+\rho]-\mathbf{x})d^6Z - n_{0e}\right], \quad (11)$$

where $\mathbf{R}+\rho$ is a particle position, $d^6Z = m_i^2 B_\parallel^* d\mathbf{R}dv_\parallel d\mu d\alpha$ is the phase space volume of the gyro-centre coordinates, ρ_{ti} is the Larmor radius evaluated with the thermal velocity v_{ti}, λ_{Di} and λ_{De} are the ion and electron Debye lengths, and $\langle\cdot\rangle_f$ is a flux surface average operator in θ and φ (see **Fig. 3**).

III. Numerical Methods

1. Gyrokinetic Solver

The conservative gyrokinetic equation (9) is written by using operators,

$$\frac{\partial \mathcal{J}f}{\partial t} = \mathcal{L}(f) + \mathcal{N}(f) + \mathcal{D}(f), \quad (12)$$
$$\mathcal{L}(f) = -\mathcal{J}\{f, H_0\}, \quad (13)$$
$$\mathcal{N}(f) = -\mathcal{J}\{f, H_1\}, \quad (14)$$
$$\mathcal{D}(f) = \mathcal{J}C(f) + \mathcal{J}S_{src}, \quad (15)$$

where we separate the convective term into a stiff linear operator, $\mathcal{L}$ involving a fast parallel motion and a non-linear operator $\mathcal{N}$ given by turbulent dynamics. As seen in Eq. (9), the convective operators are partial difference operators in 4D space $(\mathbf{R}, v_\parallel)$, and they are discretised by using the 4th order accurate non-dissipative conservative finite difference (ND-CFD),[13)]

$$\mathcal{L} = (4/3)\delta_{1j}(\overline{\mathcal{J}V_{0j}}^{1j}\overline{f}^{1j}) - (1/3)\delta_{2j}(\overline{\mathcal{J}V_{0j}}^{2j}\overline{f}^{2j}), \quad (16)$$
$$\mathcal{N} = (4/3)\delta_{1j}(\overline{\mathcal{J}V_{1j}}^{1j}\overline{f}^{1j}) - (1/3)\delta_{2j}(\overline{\mathcal{J}V_{1j}}^{2j}\overline{f}^{2j}), \quad (17)$$

where the index j runs through $j = 1-4$, $\mathbf{V}_0$ and $\mathbf{V}_1$ are unperturbed and perturbed parts of the Hamiltonian flows given by Eqs. (6) and (7). The definitions of operators are given as

$$\delta_{nj}A \equiv [A(X_j + n\Delta_j/2) - A(X_j - n\Delta_j/2)]/(n\Delta_j), \quad (18)$$
$$\overline{A}^{nj} \equiv [A(X_j + n\Delta_j/2) + A(X_j - n\Delta_j/2)]/2, \quad (19)$$

where Δ_j is the grid width in the j direction. This finite difference enables accurate and robust calculations of the turbulent convection by exactly conserving both f and f^2. The stencil grids of $\mathcal{L}$ and $\mathcal{N}$ consist of 16 neighbouring grid points in a 4D array. It is noted that in the convective operators, we use the cylindrical coordinates $\mathbf{R} = (R, \zeta, Z)$ to avoid a singularity at the centre of the plasma (see **Fig. 3**). In this case, two Hamiltonian flows depend on four coordinates, $\mathbf{V}_0 = \mathbf{V}_0(R, Z, v_\parallel, \mu)$ and $\mathbf{V}_1 = \mathbf{V}_1(R, \zeta, Z, \mu)$. In the dissipative operator $\mathcal{D}$, the collision term involving convection and diffusion in 2D velocity space $(v_\parallel, \mu)$ is discretised by the 6th order accurate centred finite difference.

2. Time Integration

In the time-integration, in order to overcome a severe Courant-Friedrichs-Lewy (CFL) condition, which is determined by $\mathcal{L}$, we adopt the 2nd order accurate additive semi-implicit Runge-Kutta method (ASIRK)[14)] as

$$\mathcal{J}f^{n+1} = \mathcal{J}f^n + \omega_1 h_1 + \omega_2 h_2, \quad (20)$$
$$h_1 = \Delta_t\left[\mathcal{L}(f^n + a_1 h_1) + \mathcal{N}(f^n) + \mathcal{D}(f^n)\right], \quad (21)$$
$$h_2 = \Delta_t\left[\mathcal{L}(f^n + c_{21}h_1 + a_2 h_2) + \mathcal{N}(f^n + b_{21}h_1) + \mathcal{D}(f^n + b_{21}h_1)\right], \quad (22)$$

where f^n is the distribution function at the n-th time step, Δ_t is the time step width, and coefficients are given as $\omega_1 = \omega_2 = 1/2$, $a_1 = a_2 = 1 - \sqrt{2}/2$, $b_{21} = 1$, and $c_{21} = \sqrt{2} - 1$. In solving Eqs. (21) and (22), we treat the time dependent non-linear operator $\mathcal{N}$ and the dissipative operator $\mathcal{D}$ explicitly, and solve ϕ and the collision frequencies (which are updated with evolving plasma profiles) once at each stage. We then solve the linear equation by an iterative approach based on a generalised conjugate residual method.[15)] Since the iterative processes use only the time independent linear operator $\mathcal{L}$, which is calculated in the initial setting, the computational cost of the ASIRK is significantly lower than full implicit approaches. It is noted that the size of linear operator is enormous $\sim 10^{10}$ with 16 stencil grids in 4D, and therefore, it can not be stored as a matrix even with a sparse format. Therefore, the present version of the iterative solver does not use any preconditioning. In a typical parameter with $\Delta_t \sim 10\Omega_i^{-1}$, the iterative solver converges to the machine precision with several tens of iterations.

3. Field Solver

Since the gyrokinetic Poisson equation (11) involves an integral flux surface average operator, the electrostatic potential is solved in the flux coordinates (ψ, θ, φ), where ψ is the poloidal flux, θ is the straight-field-line poloidal angle, and $\varphi = -\zeta$ is the toroidal angle (see **Fig. 3**). By applying a toroidal mode expansion in φ and a 2D finite element approximation in (ψ, θ), we have a discrete expression of ϕ,

$$\phi(\psi, \theta, \varphi) = \sum_n \sum_l \hat{\phi}_{nl}\Lambda_l(\psi, \theta)e^{in\varphi}, \quad (23)$$

where n is the toroidal mode number and l is an index of 2D finite elements $\Lambda(\psi, \theta)$. In this work, we use bi-quadratic spline

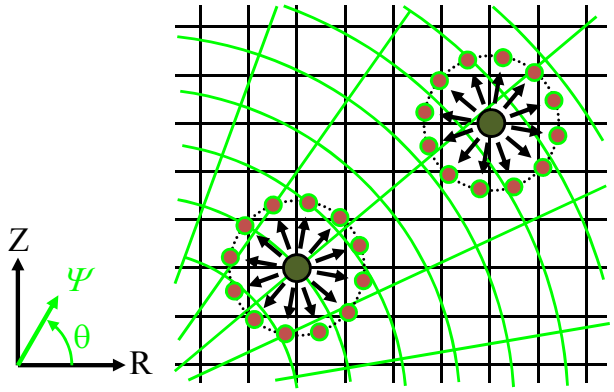

Fig. 4 Coordinate transforms between (R, ζ, Z) and (ψ, θ, φ) using a finite Larmor radius operator.

elements. The boundary condition is given by the Dirichlet boundary condition, $\phi = 0$, at the edge, while the natural boundary condition is imposed at the magnetic axis. We then write a matrix form of Eq. (11) as

$$\sum_k \mathcal{M}_{nlk}\hat{\phi}_{nk} = \hat{g}_{nl} - \hat{n}_{e0nl}, \tag{24}$$

$$\mathcal{M}_{nlk} = \frac{1}{4\pi e_i}\int \left[\frac{\rho_{ti}^2}{\lambda_{Di}^2}\nabla_\perp\Lambda_l\cdot\nabla_\perp\Lambda_k + \frac{1}{\lambda_{De}^2}\left(\Lambda_l\Lambda_k - \delta(n)\Lambda_l\frac{\int\Lambda_k J d\theta'}{\int J d\theta'}\right)\right] d\mathbf{x}_\perp. \tag{25}$$

In evaluating $\hat{g}_{nl}$, we use a similar technique as in a finite element PIC method.[16] Firstly, as shown in **Fig. 4**, we define a full finite Larmor radius (FLR) operator based on a finite point sampling technique,[17]

$$\mathcal{G}_l(\mathbf{R}_\perp, \mu) = \frac{1}{N_\alpha}\sum_{j=1}^{N_\alpha}\Lambda_l(\mathbf{R}_\perp + \rho(\mathbf{R}_\perp, \mu, \alpha_j)). \tag{26}$$

Since the operator is evaluated at fixed Eulerian grids in phase space, it is time independent and we can choose arbitrary large number for N_α. By using the full FLR operator, $\hat{g}_{nl}$ is written as

$$\hat{g}_{nl} = \int \mathcal{G}_l(\mathbf{R}_\perp, \mu)\hat{f}_n(\mathbf{R}_\perp, v_\parallel, \mu)\mathcal{J}d\mathbf{R}_\perp dv_\parallel d\mu. \tag{27}$$

On the contrary, the gyro-averaged electrostatic potential in Eq. (4) is given as

$$\langle\hat{\phi}_n\rangle_\alpha(\mathbf{R}_\perp, \mu) = \sum_l \mathcal{G}_l(\mathbf{R}_\perp, \mu)\hat{\phi}_{nl}. \tag{28}$$

Since both gyro-averaging processes are defined using the same FLR operator $\mathcal{G}_l$, they do not produce any self-force. It is noted that in Eq. (27), $\hat{g}_{nl}$ is given in (ψ, θ), while $\hat{f}_n$ is written in (R, Z), and a mapping between two coordinates and a computation of the FLR effect are performed simultaneously by multiplying the full FLR operator. Recently, we implemented a straight-field-line solver, in which the discrete Fourier transform (DFT) and field aligned filtering operator $\mathcal{F}$, is applied to Eq. (20) as $\mathcal{F}\mathcal{M}\mathcal{F}^{-1}\mathcal{F}\hat{\phi} = \mathcal{F}(\hat{g} - \hat{n}_{e0})$. This approach reduces the size of finite element matrix by two orders of magnitudes by solving only field-aligned Fourier components, which becomes dominant in fusion plasmas (see **Fig. 1**). It was demonstrated that this filtering does not affect simulation results.[18] The gyrokinetic Poisson equation is solved by the LU decomposition in LAPACK.

IV. Parallel Implementation and Porting

1. Parallel Implementation

In GT5D, the gyrokinetic solver treats evolutions of f on 5D grids in the cylindrical coordinates $(N_R, N_\zeta, N_Z, N_{v\parallel}, N_\mu)$, while the field solver uses 3D grids in the flux coordinates $(N_\psi, N_\theta, N_\varphi)$. In the parallel implementation, we use a hierarchy of three MPI communicators, consisting of $n_{MPI} = (n_R \times n_Z) \times n_\mu$ MPI processes, and each operators are parallelised using their physical properties. In Eqs. (16) and (17), $\mathcal{L}$ and $\mathcal{N}$ are discretised by 4D finite difference operators in $(R, \zeta, Z, v_\parallel)$, where μ appears in V_0 and V_1 as a parameter, and operators with different μ can be computed in parallel. Anisotropic turbulent structures require high resolution on the (R, Z) grids (see **Fig. 1**). Therefore, we use a 3D domain decomposition in (R, Z, μ) with $(n_R \times n_Z) \times n_\mu$ domains, where we set $n_\mu = N_\mu$. Since the operators are independent in the μ direction, communications of boundary conditions are confined in $n_R \times n_Z$ MPI processes. On the other hand, $\mathcal{D}$ is a 2D finite difference operator in $(v_\parallel, \mu)$, and we use a 2D domain decomposition in (R, Z). Although the change of parallelisation axes induces a transpose of the data, this collective communication is performed among n_μ MPI processes. The field solver is parallelised in a Fourier space using a 2D domain decomposition in (n, μ), where Fourier modes are solved only for $n = 0 \sim N_\varphi/4$ to avoid aliasing. It is noted that because of the gyro-average of ϕ in Eq. (28), the field solver depend also on μ. A transpose of f from (R, Z, μ) decomposition to (n, μ) decomposition is performed after the DFT in ζ. This collective communication uses $n_R \times n_Z$ MPI processes. In Eq. (27), after multiplying the FLR operator (26), the data is summed up in the μ direction. This collective communication is performed among n_μ MPI processes. In the above parallel implementation, a key technique is to confine MPI communications in a partial set of MPI communicators by using physical properties and multi-dimensional properties of each operators. This suppress the size of MPI communicators below ~ 100, and avoid global communications with n_{MPI} MPI processes. In addition to the parallelisation using MPI, all the MPI processes are subject to the SMP parallelisation with n_{SMP} cores. In the hybrid parallelisation, the code uses 4D parallelisation axes where the total number of cores is given by $n_R \times n_Z \times n_\mu \times n_{SMP}$.

2. Cost Distribution

Typical computational costs of GT5D on BX900 are listed in **Table 1**. In the cost chart, the cost of $\mathcal{N}$ is only 8.5% of $\mathcal{L}$, and a dominant part of the computational cost comes from an implicit part of ASIRK which uses $\mathcal{L}$ many times. Compared with it, other costs are sub-dominant, and in particular, the

Table 1 Typical computational costs of GT5D code on BX900. Parameters used are $(N_R, N_\zeta, N_Z, N_{v\parallel}, N_\mu) = (240, 64, 240, 128, 32)$, $(N_\psi, N_\theta, N_\varphi) = (192, 384, 64)$, and $n_R \times n_Z \times n_\mu \times n_{SMP} = 8 \times 4 \times 32 \times 4 = 4,096$ cores.

Operations	Cost per a time step (msec)	Percentage (%)
$\mathcal{L}, \mathcal{N}$	7520.8	41.7
ASIRK	6409.8	35.6
$\mathcal{D}$	1271.8	7.1
Field	132.0	0.7
Calc. total	15334.4	85.1
Comm. $\mathcal{L}, \mathcal{N}$	1343.4	7.5
Comm. ASIRK	318,1	1.8
Comm. $\mathcal{D}$	587.1	3.3
Comm. Field	436.9	2.4
Comm. total	2685.6	14.9

cost of field solver is negligible, thanks to the straight-field-line solver. In the communication part, costs of the collective communications explained in the above are rather small, and the total communication cost is $\sim 15\%$ at $4,096$ cores. Because of this low communication property, GT5D can easily scale beyond $\sim 10^4$ cores.

3. Porting Issues

A summary of platforms used in the present benchmark tests is given in **Table 2**. GT5D was originally developed on Altix3700Bx2. In porting the code on these platforms, several cares are taken. Firstly, although LAPACK used in the field solver is common on all the platforms, the DFT is implemented using a multi-stream 1D complex fast Fourier transform (FFT) available in numerical libraries on each platforms. Secondly, depending on the processor type, several tuning techniques are applied. Since the Itanium2 processor on Altix3700Bx2 is a single core processor, the code was developed for a flat-MPI parallelisation. However, most of recent platforms use multi-core processors, and therefore, a hybrid parallelisation technique with MPI and SMP is applied using an auto SMP parallelisation function of each compiler. Auto parallelisation functions on Fujitsu Fortran and on Hitachi Fortran are effective for most of loops, except for less than ten loops which involve some data dependency such as a reduction operation. By applying additional explicit SMP parallelisations (OpenMP on Fujitsu Fortran and *poption* directives on Hitachi Fortran), the SMP parallelisation is applied to all the loops. On the other hand, auto parallelisation functions on Intel Fortran and IBM XL Fortran do not work for most of loops, and they requires a full explicit parallelisation using OpenMP, which has not been implemented yet. Therefore, we use a flat MPI parallelisation on Altix3700Bx2 and BlueGene/P, while a hybrid parallelisation is applied on SR16000, HA8000, FX1, and BX900. It is noted that on BX900, both Intel Fortran and Fujitsu Fortran are available, and we use a flat MPI parallelisation when Intel Fortran is used. On IBM processors, a special care has to be taken to use double floating point unit (FPU) pipelines efficiently. On SR16000, the system operates with a simultaneous multi-threading (SMT) mode, where two SMP threads are assigned to a single core. This improves the usage of double FPU pipelines without any change in user's program. On the other hand, we applied additional optimisation to use double FPU SIMD operations on BlueGene/P.

Table 3 Summary of the sustained performance (Gflops per core) and the peak ratio (%) of the GT5D kernel code with a) single core operation, b) embarrassingly parallel MPI (EP) operation, and c) SMP operation. The size of 4D array used is $(52, 84, 68, 104)$.

	BX900	SR16k	FX1	Altix	T2K	BG/P
Single						
Gflops	3.41	2.40	2.49	1.92	1.48	0.47
%	29.1	12.8	24.9	30.0	13.7	13.7
EP						
Gflops	1.86	3.17	1.33	1.15	0.68	0.43
%	15.9	16.9	13.3	18.1	7.4	12.7
SMP						
Gflops	1.85	3.05	1.36	-	0.68	-
%	15.8	16.2	13.6	-	7.4	-

V. Benchmark Results

1. GT5D Kernel

We start the benchmark by looking at serial and SMP performances on each processors. To this end, we prepare a GT5D kernel code which consists of $\mathcal{L}$ and vector operations in a generalised conjugate residual method. This kernel treats large 4D arrays (a single array size is $\sim$ 200MB in the benchmark in **Table 3**), and the 4D finite difference operator has large strides of memory access. Therefore, its performance strongly depends on the cache and the memory bandwidth available on each processors. To look at these effects, we measured the performance in three different conditions, a) single core operation, b) embarrassingly parallel MPI operation (which does not involve any communication among processes), and c) SMP operation. In b) and c), a computing node is filled with MPI and SMP processes, which share the cache, the memory bandwidth, and the memory on the node. In c), the number of SMP threads is chosen to be same as the number of cores per a processor (SR16000 uses 4 threads per a processor because of SMT), and multiple SMP operations are processed with an embarrassingly parallel mode. Table 3 shows a summary of the benchmark results. Although the number of floating point operations changes depending on the platform, in the benchmark, we estimate floating points per a second (flops) by using the flop counts on Altix3700Bx2 and the processing time on each platform. In the case a), BX900 and Altix3700Bx2 show excellent performances reaching at the sustained performance of $\sim 30\%$. On BlueGene/P, the sustained performance is improved from 8.6% to 13.7% by applying optimisations to use double FPU SIMD operations. In the case b), most of platforms show degradation of performances, except for SR16000. This improvement is due to SMT opera-

Table 2 Summary of platforms used in the benchmark

Machines	BX900	SR16000	FX1	Altix3700Bx2	T2K(HA8000)	BlueGene/P
Processor	Nehalem-EP	POWER6	SPARC64VII	Itanium2	Opteron	PowerPC450
GHz	2.93	4.7	2.5	1.6	2.3	0.85
Gflops/core	11.72	18.8	10.0	6.4	9.2	3.4
On chip cache(MB)	8	4	6	6	2	8
Cores/Processor	4	2	4	1	4	4
Cores/Node	8	32	4	128	16	4
Memory/Node(GB)	24	256	32	819	32	4
Interconnect	InfinibandQDR	InfinibandDDR	InfinibandDDR	NUMAlink	Myrinet10G	BG/P network
GB/s	8×2	16×2	2×2	3.2×2	5×2	5.1×2
Compiler	Fujitsu Fortran	Hitachi Fortran	Fujitsu Fortran	Intel Fortran	Hitachi Fortran	IBM XL Fortran
FFT Library	SSL2	MATRIX	SSL2	SCSL	MATRIX	FFTW

tion, which efficiently fills double FPU pipelines. As a result, the sustained performance of SR16000 becomes comparable to BX900 and Altix3700Bx2. On most of platforms (BX900, FX1, Altix3700Bx2, and T2K), the sustained performance is decreasing to about a half of the case a), which clearly shows that the memory access is limiting the total performance. On the other hand, BlueGene/P shows a small degradation of the sustained performance, suggesting that its performance is still limited by the FPU operation rather than the memory access. In the case c), all the platforms tested show no degradation of the sustained performance compared with the case b), which shows that costs related to overheads such as the barrier synchronisation is very small.

2. Hybrid Parallel

Table 4 shows comparisons of flat-MPI and hybrid parallel operations of GT5D using 4,096 cores on BX900. In the comparison, we also tested a flat-MPI operation with Intel Fortran. In flat-MPI cases, Intel Fortran and Fujitsu Fortran show similar performances. Compared with these performances, hybrid cases show two important improvements. Firstly, the memory usage is significantly decreased in the hybrid parallel case due to the reduction of 1) boundary buffers in the domain decomposition, 2) replicas of operators used in the field solver, 3) working arrays used in the data transpose, and 4) buffers used by the system in MPI communications. In the present test case, the total memory usage is reduced by $\sim 59\%$ at $n_{SMP} = 4$ and $\sim 67\%$ at $n_{SMP} = 8$. The 1/3 reduction of memory usage is a significant reduction of the computational cost. Secondly, as is expected, the reduction of MPI processes leads to the reduction of MPI communication costs. In the 4,096 core case, the reduction of communication cost is $\sim 10\%$ of the total cost. This improvement becomes more pronounced beyond 10^4 cores, and the hybrid parallelization becomes essential in this regime. The reduction of MPI processes is one of key issues in achieving good scalability. On BX900, the best performance is obtained at $n_{OMP} = 4$, because it fits the number of cores per a processor.

3. Scaling

Figure 5 shows the sustained performances obtained in the strong scaling tests. Because of the memory limit, Altix3700Bx2 and T2K use a slightely small problem size with $(N_R, N_\zeta, N_Z, N_{v\parallel}, N_\mu) = (240, 64, 240, 100, 32)$, while the other platforms use $(N_R, N_\zeta, N_Z, N_{v\parallel}, N_\mu) = (240, 64, 240, 128, 32)$. The compiler options used in each platforms are summarized in Table 5. In the benchmark, flops are estimated based on the flop counts on Altix3700Bx2 and the processing time on each platform. On every platforms, GT5D scales very well up to the maximum available cores. The sustained performances around $\sim 2,000$ cores are summarized in Table 6. The performance roughly follows the sustained performance of the GT5D kernel except for SR16000, which has the minimum interconnect bandwidth per a thread (0.25GB/s/threads). On BX900, the sustained performance is above 10% up to 8,192 cores, and the maximum sustained performance at 16,384 cores is $\sim$ 19.4Tflops (10.1%). At 16,384 cores, the communication cost is $\sim 35.2\%$ of the total cost. According to the Amdahl's law, a processing time with n cores T_n is written as

$$\frac{T_n}{T_1} = \frac{1}{1 - a + a/n}, \tag{29}$$

where a is the parallel efficiency (or the ratio of operations which are parallelised). When one has the processing time T_n and T_m at n and m cores, Eq. (29) yields a as

$$a = \frac{T_n - T_m}{T_n \frac{m-1}{m} - T_m \frac{n-1}{n}}. \tag{30}$$

From the performance data of BX900 at 2,048 and 16,384 cores, the parallel efficiency is estimated as $a \sim 0.999969$.

Table 4 Comparisons of computational performances of flat-MPI and hybrid parallel operations on BX900. The problem size is $(N_R, N_\zeta, N_Z, N_{v\parallel}, N_\mu) = (240, 32, 240, 128, 32)$, $(N_\psi, N_\theta, N_\varphi) = (192, 384, 64)$. 4,096 cores with $n_R \times n_Z \times n_\mu \times n_{SMP} = 16 \times 8 \times 32 \times 1$, $n_R \times n_Z \times n_\mu \times n_{SMP} = 8 \times 4 \times 32 \times 4$, and $n_R \times n_Z \times n_\mu \times n_{SMP} = 4 \times 4 \times 32 \times 8$ are used.

Fortran Compiler	Intel	Fujitsu	Fujitsu	Fujitsu
SMP	1	1	4	8
Total/step(msec)	10231	10397	9287	10929
Comm./step(msec)	2415	2608	1517	1534
Total memory(GB)	3641	3754	1545	1250
Sustained Gflops	4979	4900	5486	4661

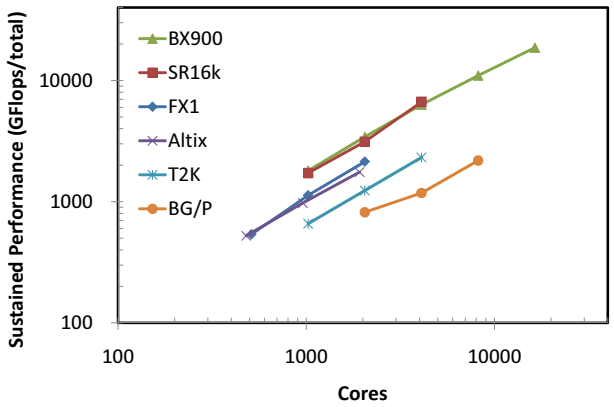

Fig. 5 The strong scaling of GT5D. The problem size is $(N_R, N_\zeta, N_Z, N_{v\parallel}, N_\mu) = (240, 64, 240, 100, 32)$ for Altix3700Bx2 and T2K. The other platforms use $(N_R, N_\zeta, N_Z, N_{v\parallel}, N_\mu) = (240, 64, 240, 128, 32)$.

Table 5 Compiler options used on each platforms.

BX900	-Kfast -Kparallel,OMP -Kmfunc=2 -Kocl
SR16k	-Oss -64 -precexp=basic -model=L1
FX1	-Kfast -Kparallel,OMP -Kmfunc=2
	-Kocl -Kprefetch_cache_level=1
Altix	-O3 -ipo
T2K	-Oss -64 -HF90 -precexp=basic
BG/P	-qarch=450d -qtune=450 -O3 -qhot

By using a, the parallelisation efficiency (or the ratio of calculation time to the total processing time) is given as

$$E_n = \frac{T_n}{T_1 n} = \frac{1}{n(1-a)+a}. \tag{31}$$

This relation indicates that in the present benchmark case, the calculation and communication costs become comparable ($E_n = 0.5$) at $n \sim 30,000$ cores. It is noted that in a strong scaling test, the size of the benchmark problem is limited by the memory size available at the minimum number of cores, and the parallel efficiency may be further improved by using larger problem sizes.

VI. Summary

In this work, we have presented the performances of GT5D code on advanced massively parallel platforms. GT5D solves the 5D gyrokinetic equations, and all the operators in the gyrokinetic solver and the field solver are highly parallelised with multi-dimensional domain decomposition by taking advantages of physical properties of each operators. One key technique in the parallelisation is to confine MPI communications in a subset of communicators and to avoid global communications among all the MPI processes. Another key technique is to reduce the number of MPI processes via a hybrid parallelisation with MPI and SMP. By using the 4D $(n_R, n_Z, n_\mu, n_{SMP},)$ hybrid parallelisation, the sizes of MPI communicators are suppressed below ~ 100 even beyond 10^4 cores. The hybrid parallel technique is also effective in reducing the memory usage. The code is ported on six massively parallel platforms, and comprehensive benchmark tests were performed. The benchmark of GT5D kernel showed that its performance is limited mainly by the memory access, and that the sustained performance strongly depends on the processor type. The scaling tests of GT5D also showed similar tendency in the sustained performance. In the benchmark results, a good scalability was confirmed up to several thousands cores on every platforms, and the maximum sustained performance of ~ 19.4 Tflops was achieved using 16,384 cores on BX900.

Table 6 Summary of the sustained performances (Tflops) and the peak ratio (%) at $\sim 2,000$ cores in the scaling tests shown in **Fig. 5**.

	BX900	SR16k	FX1	Altix	T2K	BG/P
Cores	2048	2048	2048	1920	2048	2048
SMP	4	4	4	1	4	1
Tflops	3.42	3.12	2.14	1.75	1.23	0.81
%	14.2	8.1	10.5	14.3	6.6	11.7

Compared with the present benchmark parameter, the problem size of the next generation fusion reactor will be several times larger in the size (more than an order of magnitude larger in the volume). In addition, a computation of electromagnetic turbulence requires a treatment of fast electron motion, which is faster than ions by $\sim \sqrt{m_i/m_e}$. It is noted that the field solver and parallelization techniques developed on GT5D can be applied also to electromagnetic turbulence, provided that its amplitude is small and the lowest order magnetic configuration is given by nested magnetic surfaces. To achieve an order of magnitude larger computing power, a plausible option would be to use many-core processors, provided that the memory access can keep up with the increasing number of cores. In addition, improvements in the MPI parallelisation would be required. The present version of GT5D uses only three coordinates (R, Z, μ) as the MPI parallelisation axes, and we still have a possibility to improve the parallelisation using remaining two coordinates $(\zeta, v_\parallel)$. This multi-dimensional property is the strength of gyrokinetic simulations. In future works, further improvements of GT5D will be addressed toward Peta and Exa scale computation.

Acknowledgments

The benchmark tests were performed on Altix3700Bx2, BX900, and FX1 at JAEA, SR16000 at NIFS, FX1 at Nagoya Univ., T2K(HA8000) at Univ. Tokyo, and BlueGene/P at EPFL. We would like to thank Drs. N. Nakajima and L. Villard for their support in using machines at NIFS and EPFL. One of the authors (Y.I.) is supported by the MEXT, Grant No. 22866086.

References

1) A. J. Brizard, T. S. Hahm, *Rev. Mod. Phys.*, **79**, 421 (2007).

2) X. Garbet, Y. Idomura, L. Villard, T.-H. Watanabe, *Nucl. Fusion*, **50**, 043002 (2010).

3) Y. Idomura, T.-H. Watanabe, H. Sugama, *C. R. Physique*, **7**, 650 (2006).
4) http://www.iter.org/
5) Y. Idomura, M. Ida, T. Kano *et al.*, *Comput. Phys. Commun.*, **179**, 391 (2008).
6) http://ts.fujitsu.com/products/standard_servers/blade/bx900/primergy_bx900_index.html
7) http://www.hitachi.co.jp/Prod/comp/hpc/SR_series/sr16000/index.html
8) http://jp.fujitsu.com/solutions/hpc/products/fx1.html
9) http://www.sgi.com/products/remarketed/altix3000/index.html
10) http://www.hitachi.co.jp/Prod/comp/OSD/pc/ha/index.html
11) http://www-03.ibm.com/systems/deepcomputing/solutions/bluegene/
12) X. Q. Xu, M. N. Rosenbluth, *Phys. Fluids*, **B3**, 627 (1991).
13) Y. Idomura, M. Ida, S. Tokuda *et al.*, *J. Comput. Phys.*, **226**, 244 (2007).
14) X. Zhong, *J. Comput. Phys.*, **128**, 19 (1996).
15) S. C. Eisenstat, H. C. Elman, M. H. Schultz, *SIAM J. Numer. Anal.*, **20**, 345 (1983).
16) M. Fivaz *et al.*, *Comput. Phys. Commun.*, **111**, 27 (1998).
17) W. W. Lee, *J. Comput. Phys.*, **72**, 243 (1987).
18) S. Jolliet *et al*, *J. Comput. Phys.* in press.