June 22th, 2023
3rd IFERC workshop on the usage of GPU based system
for fusion applications

GPU Acceleration of 5D Global Gyrokineitc Code GKNET by OpenACC Directives

Contents

1. Introduction (6/21)
2. Utilization of Multi-GPU (3/21)
3. Direct GPU-GPU data transfer (3/21)
4. Installation of cuFFT (4/21)
5. Nonlinear Simulation of JT-60SA Plasma (3/21)
6. Summary (2/21)

Kenji IMADERA **Shuhei GENKO** **Shuhei OKUDA**

Graduate School of Energy Science, Kyoto University, Japan

Acknowledgement

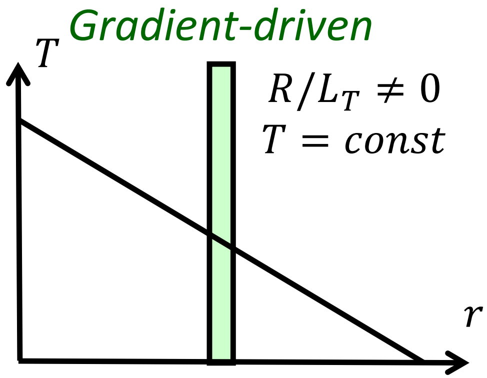
Masatoshi YAGI, Naoaki MIYATO, Haruki SETO (QST), Akira NARUSE (NVIDIA Japan)

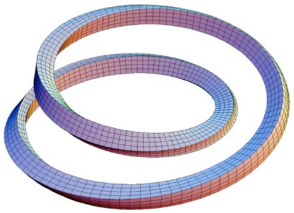
Global/Local Gyrokinetics

Local δf approach

$$\cancel{\partial_t f_{eq}} - [\cancel{H, f_{eq}}] = C\cancel{(f_{eq})} + \cancel{S}$$

$$\partial_t \delta f - [H, \delta f] - [\delta H, f_{eq}] - [\delta H, \delta f] = C(\delta f)$$





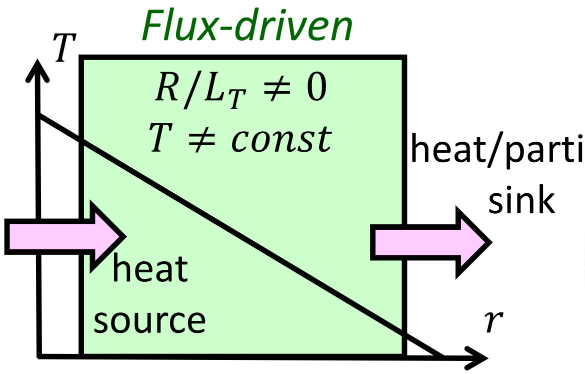
☺ Very powerful tool to estimate turbulent transport process

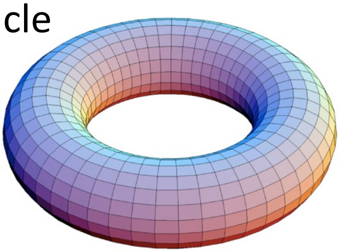
☺ Computationally efficient
-> multi(ion/electron)-scale simulation

Global full-f approach

$$\partial_t f_{eq} - [H, f_{eq}] = C(f_{eq}) + S$$

$$\partial_t \delta f - [H, \delta f] - [\delta H, f_{eq}] - [\delta H, \delta f] = C(\delta f)$$





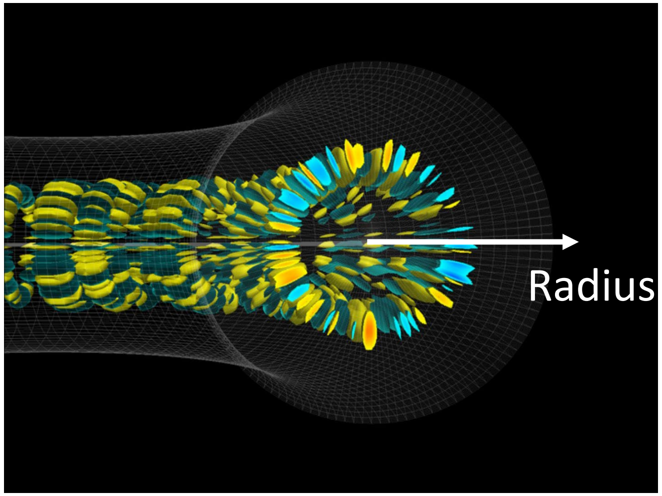
☺ Global effects can be treated precisely

☺ Mean E_r is self-consistently determined
-> Internal Transport Barrier (ITB)

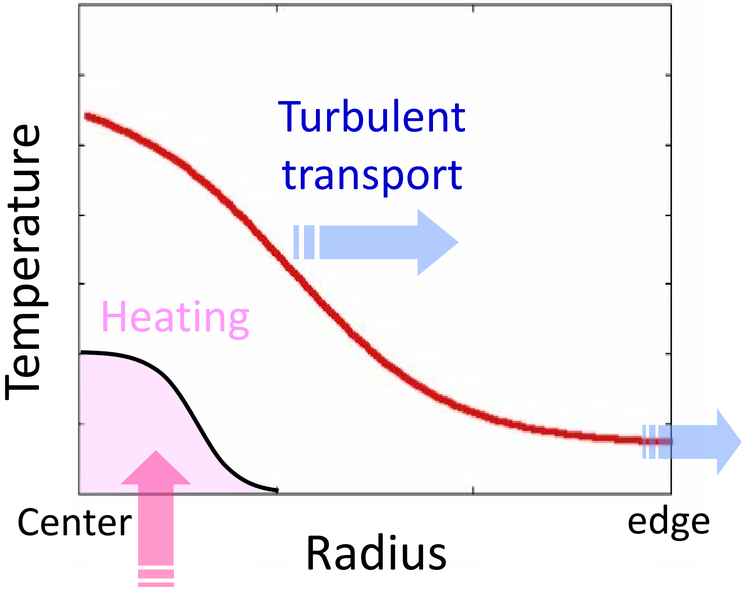
☺ Both neoclassical & turbulent transport process can be traced

Typical Flux-driven Simulation

Animation of turbulence structure



Animation of plasma temperature



- ✓ In flux-driven simulations, the external heat source is introduced, which sustains the background temperature profile. As the result, we can treat non-decaying turbulence over the confinement time.
- ✓ By means of such flux-driven simulations, we try to understand the relationship between background profile evolution and turbulence dynamics.

Global Gyrokinetic Code *GKNET*

5D phase-space Boltzmann equation for distribution functions

$$\underbrace{\frac{\partial f_s}{\partial t}}_{\text{Time derivatives of distribution function for species } s} + \underbrace{\sum_{i=1}^{5} \frac{dx_i}{dt} \frac{\partial f_s}{\partial x}}_{\text{Convection In 5D space}} = C_{coll}$$

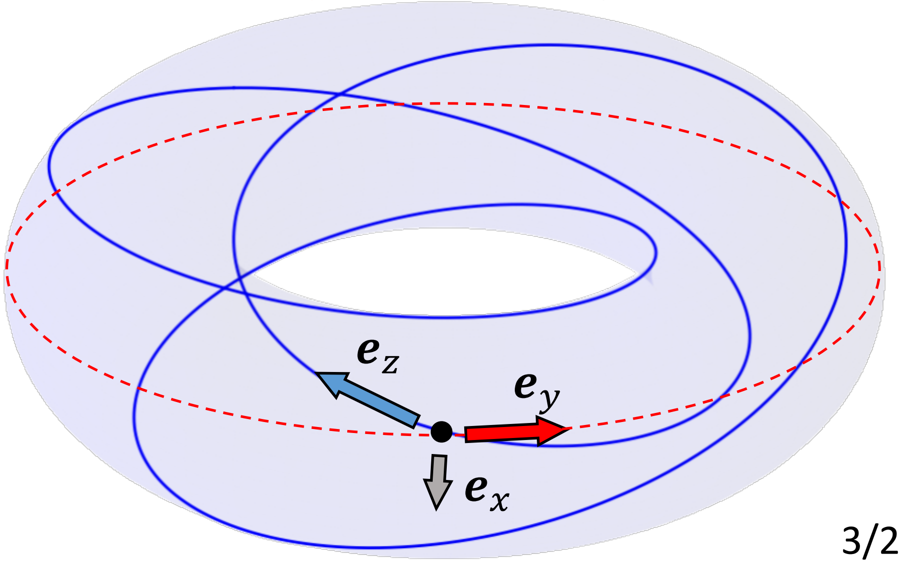
⟷

3D real-space Poisson equation for electrostatic potential

$$-\nabla^2 \underbrace{\phi}_{\text{Electrostatic potential (related to convection in the Boltzmann equation)}} = \underbrace{\sum_s e_s \int f_s d\boldsymbol{v}}_{\text{Total charge density}}$$

- ✓ For performing flux-driven simulations in 5D phase-space, GKNET (GyroKinetic Numerical Experiment for Tokamak) has been developed and MPI-parallelized.
- ✓ To reduce the number of the simulation grid and the resultant calculation time, we recently introduced field aligned coordinate. [Okuda, PFR-2023]

$$\begin{cases} x = \rho \\ y = -\zeta + \int_0^\theta \nu(\rho, \theta') d\theta' \\ z = \theta \end{cases}$$



Global Gyrokinetic Code *GKNET* -Vlasov Solver-

5D phase-space Gyrokinetic Boltzmann equation

$$\frac{\partial f}{\partial t} + \frac{d\mathbf{R}}{dt} \cdot \frac{\partial f}{\partial \mathbf{R}} + \frac{dv_\parallel}{dt}\frac{\partial f}{\partial v_\parallel} = C_{coll} \qquad \boldsymbol{R} = (x, y, z)$$

$$\frac{d\mathbf{R}}{dt} \equiv \{\mathbf{R}, H\} = v_\parallel \mathbf{b}(\mathbf{R}) + \frac{c}{eB_\parallel^*(\mathbf{R}, v_\parallel)}\mathbf{b}(\mathbf{R}) \times \left[e\nabla\langle\phi(\mathbf{R})\rangle_\alpha + m_i v_\parallel^2 \mathbf{b}(\mathbf{R}) \cdot \nabla\mathbf{b}(\mathbf{R}) + \mu\nabla B(\mathbf{R})\right]$$

$$\frac{dv_\parallel}{dt} \equiv \{v_\parallel, H\} = -\frac{\mathbf{B}_\parallel^*(\mathbf{R}, v_\parallel)}{m_i B_\parallel^*(\mathbf{R}, v_\parallel)} \cdot \left[e\nabla\langle\phi(\mathbf{R})\rangle_\alpha + \mu\nabla B(\mathbf{R})\right]$$

<u>Vlasov solver</u>

- ✓ Spatial discretization: 4th-order Morinishi scheme [Idomura, JCP-2007]
- ✓ Time integration: 4th-order explicit Runge-Kutta scheme

Ex. 2D case of Morinishi scheme

$$\frac{\partial f}{\partial t} = \underline{-v_x \frac{\partial f}{\partial x}} \underline{- v_y \frac{\partial f}{\partial x}} = \underline{-\frac{1}{2}\left[v_x \frac{\partial f}{\partial x} + \frac{\partial(v_x f)}{\partial x}\right]} \underline{-\frac{1}{2}\left[v_y \frac{\partial f}{\partial y} + \frac{\partial(v_y f)}{\partial y}\right]} \quad \left(\because \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} = 0\right)$$

⬇ Discretize each derivative by using 4th-order central FDM

$$\left(\frac{\partial f}{\partial t}\right)_{i,j}^n = \underline{-\frac{1}{2}\left\{\left[v_{x,i,j}^n\left(\frac{4}{3}\frac{f_{i+1,j}^n - f_{i-1,j}^n}{2\Delta x} - \frac{1}{3}\frac{f_{i+2,j}^n - f_{i-2,j}^n}{4\Delta x}\right)\right] + \left[\frac{4}{3}\frac{v_{x,i+1,j}^n f_{i+1,j}^n - v_{x,i-1,j}^n f_{i-1,j}^n}{2\Delta x} - \frac{1}{3}\frac{v_{x,i+2,j}^n f_{i+2,j}^n - v_{x,i-2,j}^n f_{i-2,j}^n}{4\Delta x}\right]\right\}}$$

$$\underline{-\frac{1}{2}\left\{\left[v_{x,i,j}^n\left(\frac{4}{3}\frac{f_{i,j+1}^n - f_{i,j-1}^n}{2\Delta y} - \frac{1}{3}\frac{f_{i,j+2}^n - f_{i,j-2}^n}{4\Delta y}\right)\right] + \left[\frac{4}{3}\frac{v_{x,i,j+1}^n f_{i,j+1}^n - v_{x,i,j-1}^n f_{i,j-1}^n}{2\Delta y} - \frac{1}{3}\frac{v_{x,i,j+2}^n f_{i,j+2}^n - v_{x,i,j-2}^n f_{i,j-2}^n}{4\Delta y}\right]\right\}}$$

Global Gyrokinetic Code *GKNET* -Poisson Solver-

3D real-space Gyrokinetic quasi-neutrality condition

$$\nabla\cdot\left(\frac{m_i n(x)}{B(x,z)^2}\nabla_\perp\phi\right)-\frac{e^2 n}{T_e}\left[\phi-\langle\phi\rangle_f\right]=-2\pi e\iint\langle\delta f_i\rangle\frac{B_\parallel^*}{m_i}dv_\parallel d\mu$$

$$\Longrightarrow\quad (L_0+L_1)\phi(x,y,z)=s(x,y,z)$$

$$L_0=c_1(x,z)\frac{\partial^2}{\partial x^2}+c_2(x,z)\frac{\partial^2}{\partial y^2}+c_3(x,z)\frac{\partial}{\partial x}\frac{\partial}{\partial y}+c_4(x,z)\frac{\partial}{\partial x}+c_5(x,z)\frac{\partial}{\partial y}+c_6(x)$$

$$L_1=l_1(x,z)\frac{\partial}{\partial x}\frac{\partial}{\partial z}+l_2(x,z)\frac{\partial}{\partial y}\frac{\partial}{\partial z}+l_3(x,z)\frac{\partial^2}{\partial z^2}+l_4(x,z)\frac{\partial}{\partial z}$$

Poisson solver

Step-1 : FFT along the y direction ← because all the coefficients are independent to y

Step-2 : Set the initial guess $\hat{\phi}_n^{(0)}(x,z)$, and then solve $\hat{L}_0\hat{\phi}_n^{(1)}(x,z)+\hat{L}_{1,D}\hat{\phi}_n^{(1)}(x,z)=\hat{s}_n(x,z)-\hat{L}_{1,ND}\hat{\phi}_n^{(0)}(x,z)$ by using the 1D matrix solver

Step-3 : By repeating Step-2 (=Jacobi method), get the conveged solution $\hat{\phi}_n$

← because $\dfrac{\partial\phi}{\partial z}$ is higher order, a few iterations are enough for the convergence

Targets of This Talk

✓ We are aiming at the GPU acceleration of GKNET by using OpenACC directives on MARCONI 100 (CINECA, Italy).

A) Multi-GPU

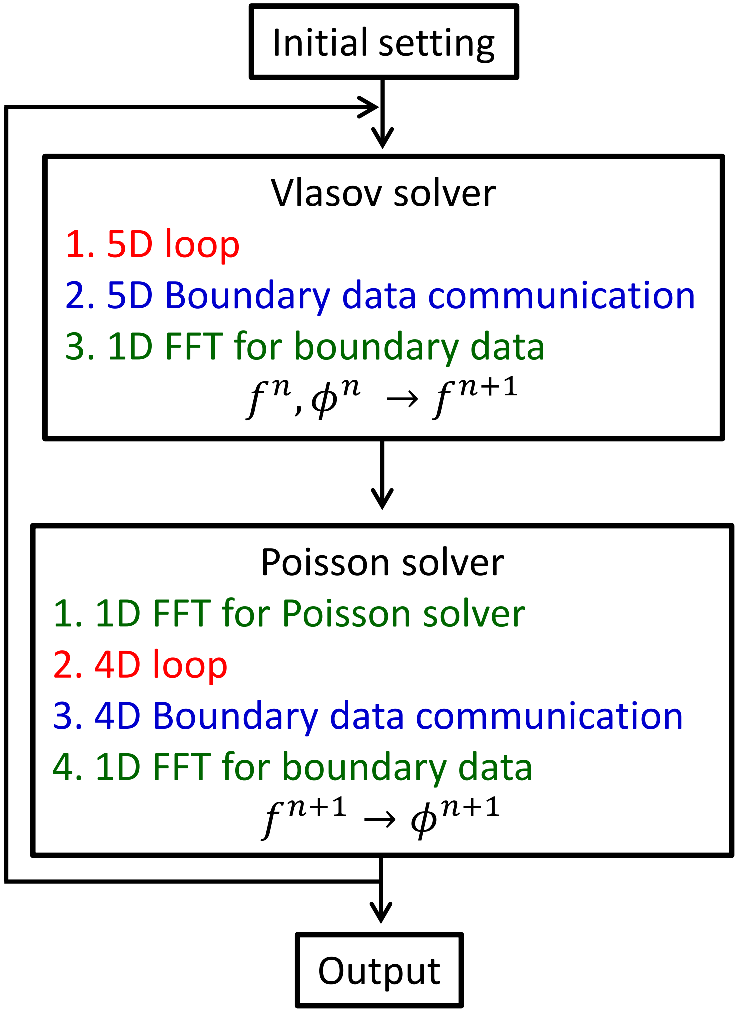
✓ By utilizing multi-GPU on each node, we accelerated 5D/4D loops in the Vlasov/Poisson solvers.

B) Direct GPU-GPU data transfer

✓ Instead of CPU-GPU data transfer, direct GPU-GPU data transfer is introduced for 5D/4D boundary data exchange in the Vlasov/Poisson solvers.

C) Batched CuFFT

✓ For boundary data in the Vlasov/Poisson solvers & for solving the quasi-neutrality condition in the Poisson solver, the batched cuFFT is installed.



Initial setting

Vlasov solver

1. 5D loop
2. 5D Boundary data communication
3. 1D FFT for boundary data

$f^n, \phi^n \rightarrow f^{n+1}$

Poisson solver

1. 1D FFT for Poisson solver
2. 4D loop
3. 4D Boundary data communication
4. 1D FFT for boundary data

$f^{n+1} \rightarrow \phi^{n+1}$

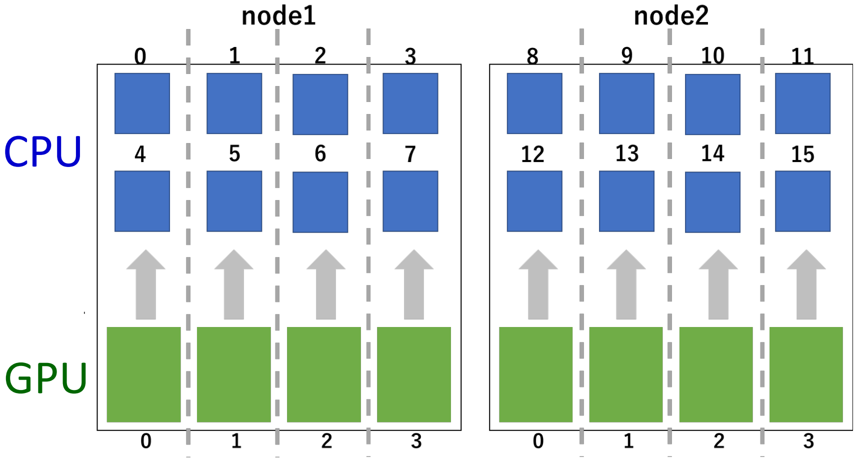
Output

Utilization of Multi-GPU - 1

Typical 5D loop in Vlasov solver

```
def = acc_get_num_devices (acc_devise_nvidia)
gpuid = mod(rank, def)
call acc_set_device_num(gpuid,
acc_device_defaut)

!$acc data copy(…) &
!$acc& copyin(...) &
!$acc& create(…)

!$acc wait
!$acc kernels
!$acc loop collapse(4) gang vector
DO x_i = 3, N_x_p+2
   DO y_i = 3, N_y_p+2
      DO z_i = 3, N_z_p+2
         DO v_i = 3, N_v+2
           DO u_i = 3, N_u+3

              Heavy calculation

         END DO
      END DO
   END DO
END DO
!$acc end kernels
```

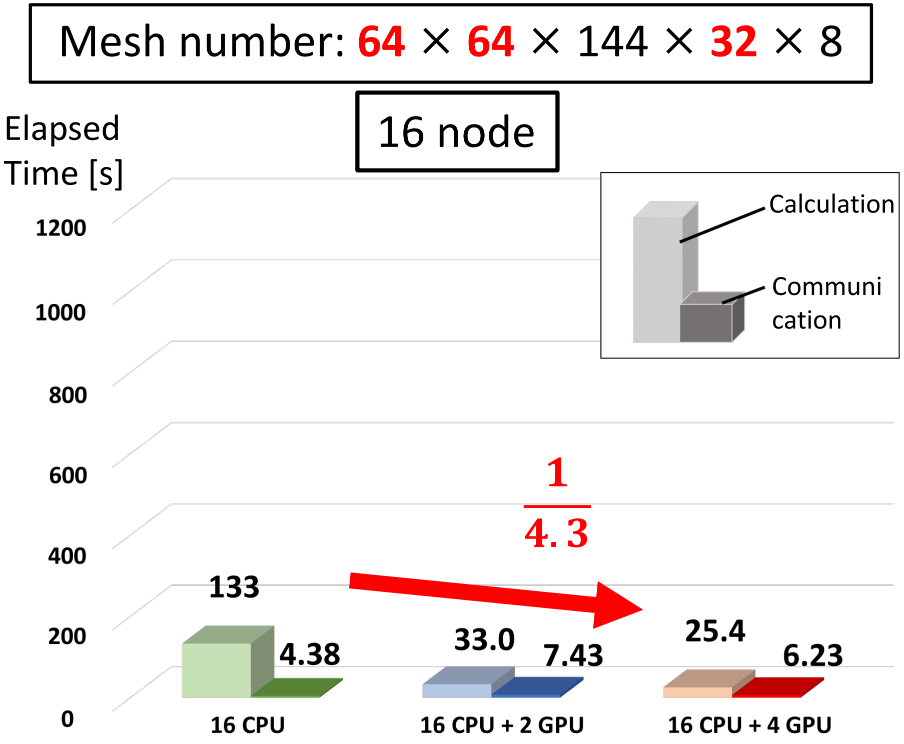
Image of GPU allocation to CPUs

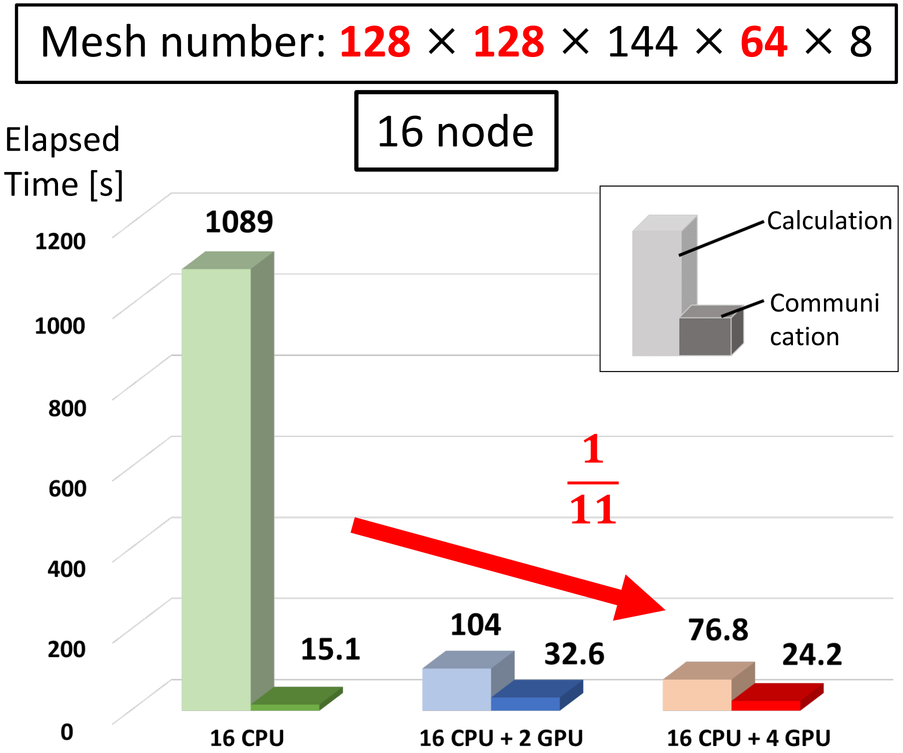


- ✓ Each CPU is explicitly linked to the GPU in same node.
- ✓ The OpenACC data directives (copy, copyin, etc.) are utilized for CPU-GPU data transfer.
- ✓ The 4D/5D loops are collapsed to one loop and then distributed to each GPU.

Utilization of Multi-GPU - 2

Benchmark test of Vlasov solver



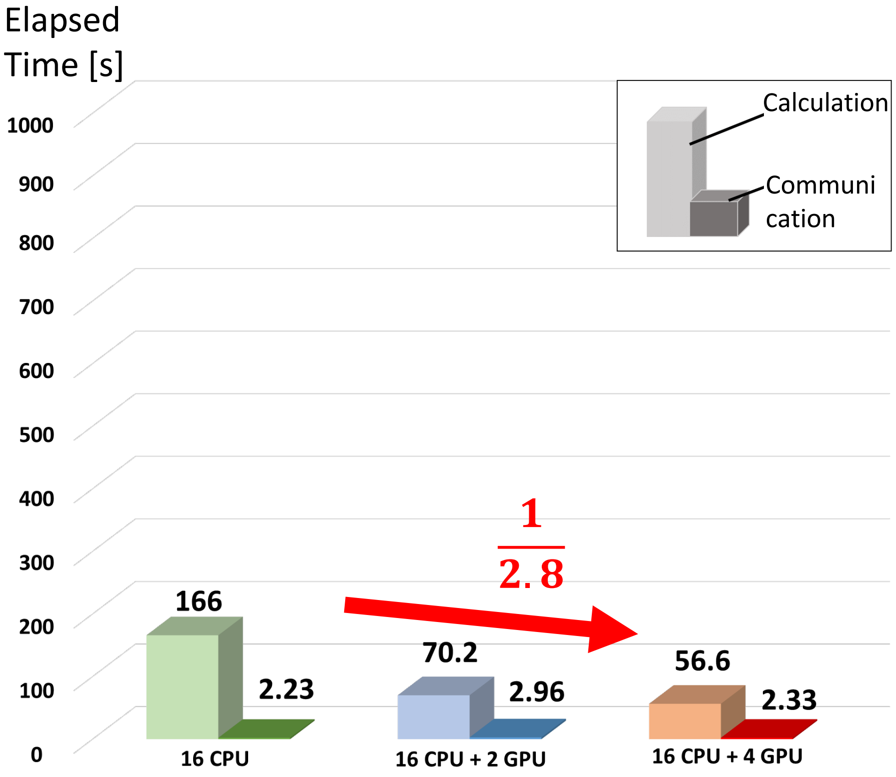


- ✓ By increasing the number of GPUs on each nodes, the calculation time is reduced, which tendency is enhanced in the larger problem size case.
- ✓ But the speed-up rate is still low because the FFT part is not GPU parallelized yet.

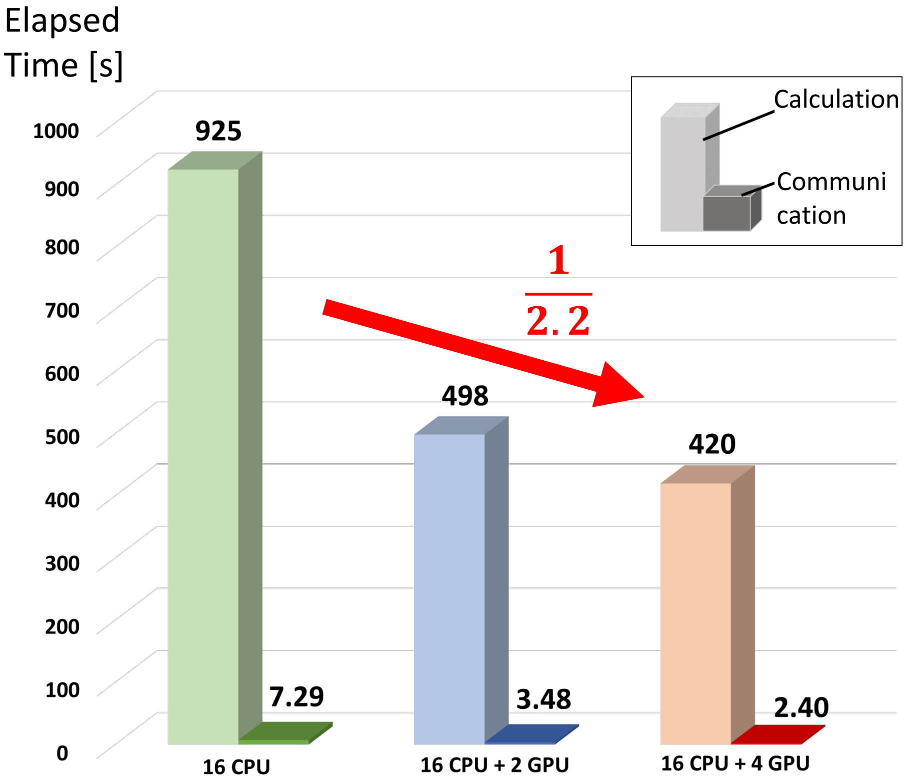
Utilization of Multi-GPU - 3

Benchmark test of Poisson solver

Mesh number: **64** × **64** × 144 × **32** × 8



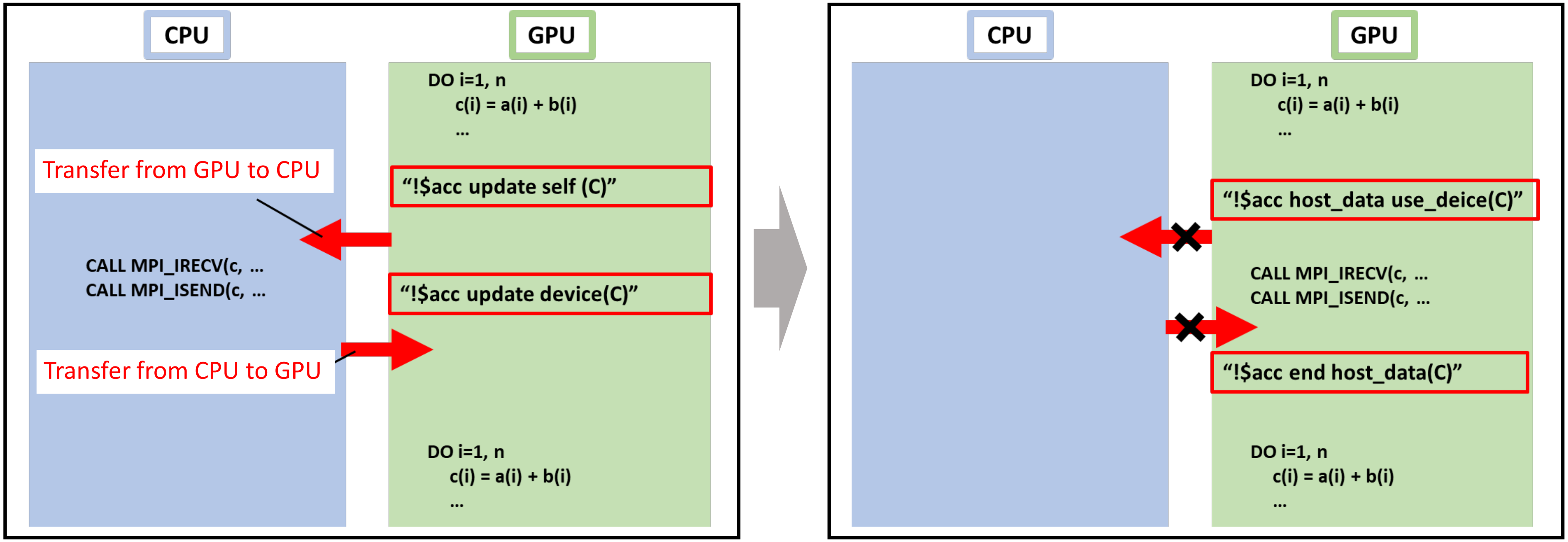
Mesh number: **128** × **128** × 144 × **64** × 8



✓ Same tendency can be observed but the FFT part is not GPU parallelized yet.

Direct GPU-GPU Data Transfer - 1

- ✓ We transferred the boundary data from GPU to CPU before the MPI data communications (Left case).
- ✓ To improve this part, direst GPU-GPU data transfer is utilized by using “acc_host_data_use_device” (Right case).



```
DO i=1, n
    c(i) = a(i) + b(i)
    …
“!$acc update self (C)”
CALL MPI_IRECV(c, …
CALL MPI_ISEND(c, …
“!$acc update device(C)”
DO i=1, n
    c(i) = a(i) + b(i)
    …
```

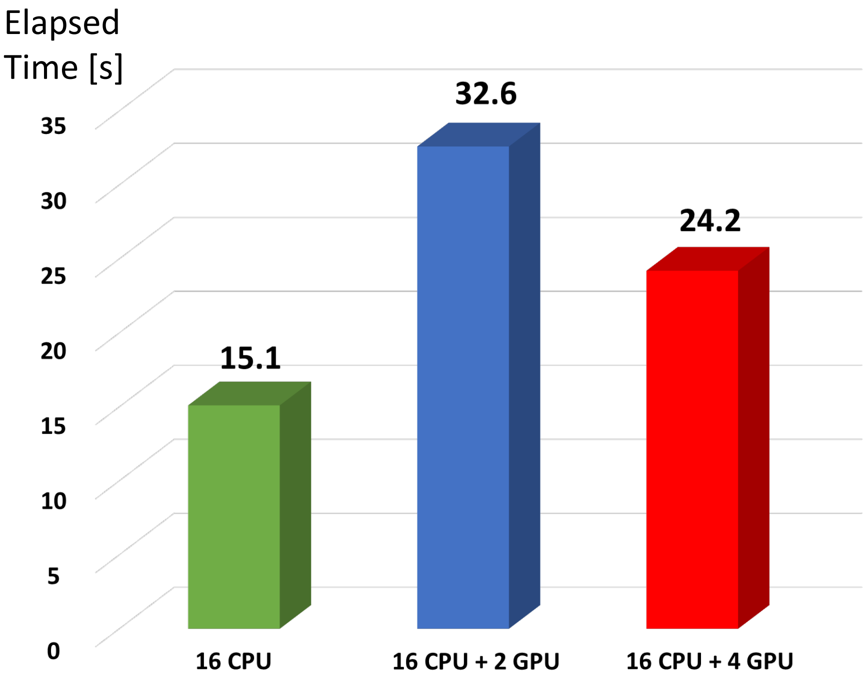
```
DO i=1, n
    c(i) = a(i) + b(i)
    …
“!$acc host_data use_deice(C)”
CALL MPI_IRECV(c, …
CALL MPI_ISEND(c, …
“!$acc end host_data(C)”
DO i=1, n
    c(i) = a(i) + b(i)
    …
```

Direct GPU-GPU Data Transfer - 2

Benchmark test of Vlasov solver

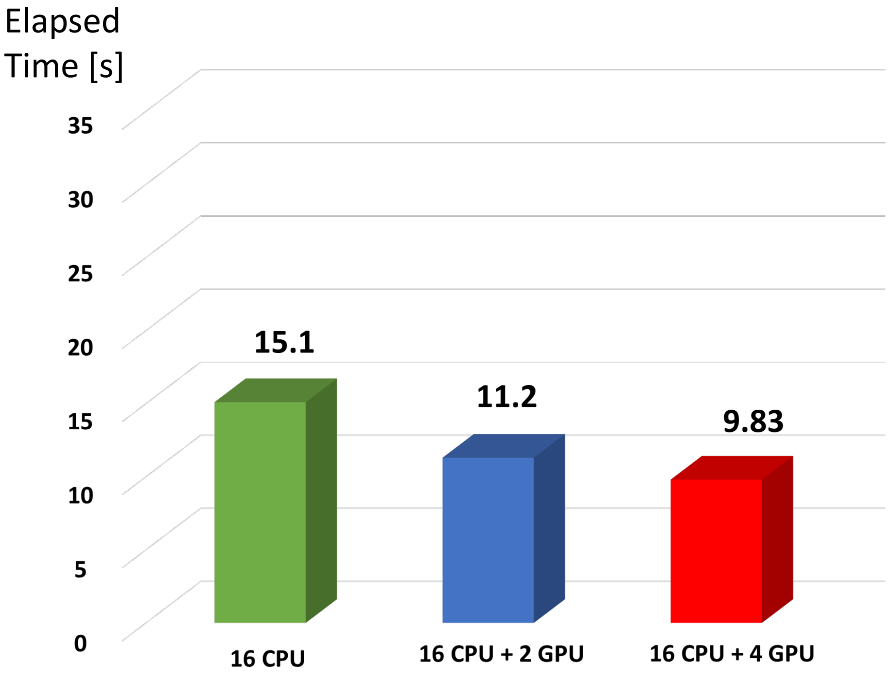
CPU-GPU communication

	16CPU	16CPU+1GPU	16CPU+4GPU
Calculation	1089	104	76.8
Communication	15.1	32.6	24.2



Direct GPU-GPU communication

	16CPU	16CPU+1GPU	16CPU+4GPU
Calculation	1089	104	76.8
Communication	15.1	11.3	9.83



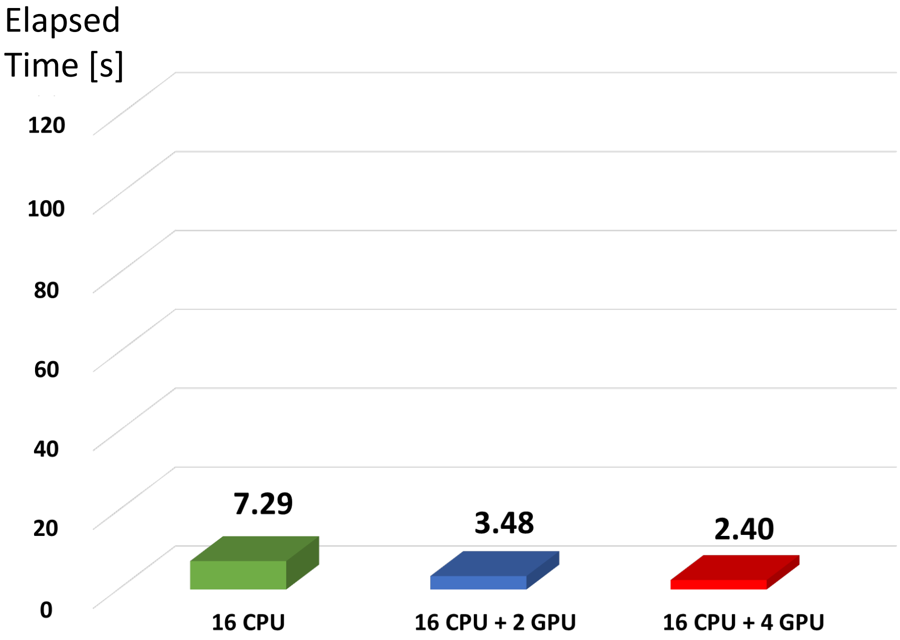
- ✓ Since the CPU-GPU data transfer is skipped by the direct GPU-GPU one, the communication time is reduced.

Direct GPU-GPU Data Transfer - 3

Benchmark test of Poisson solver

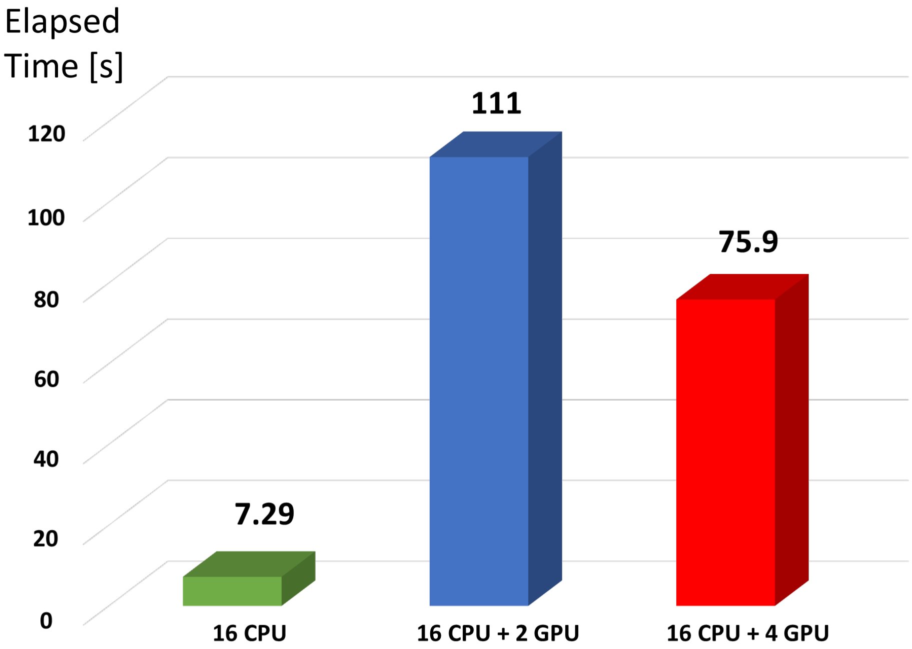
CPU-GPU communication

	16CPU	16CPU+2GPU	16CPU+4GPU
Calculation	925	498	420
Communication	7.29	3.48	2.40



Direct GPU-GPU communication

	16CPU	16CPU+2GPU	16CPU+4GPU
Calculation	925	498	420
Communication	7.29	111	75.9



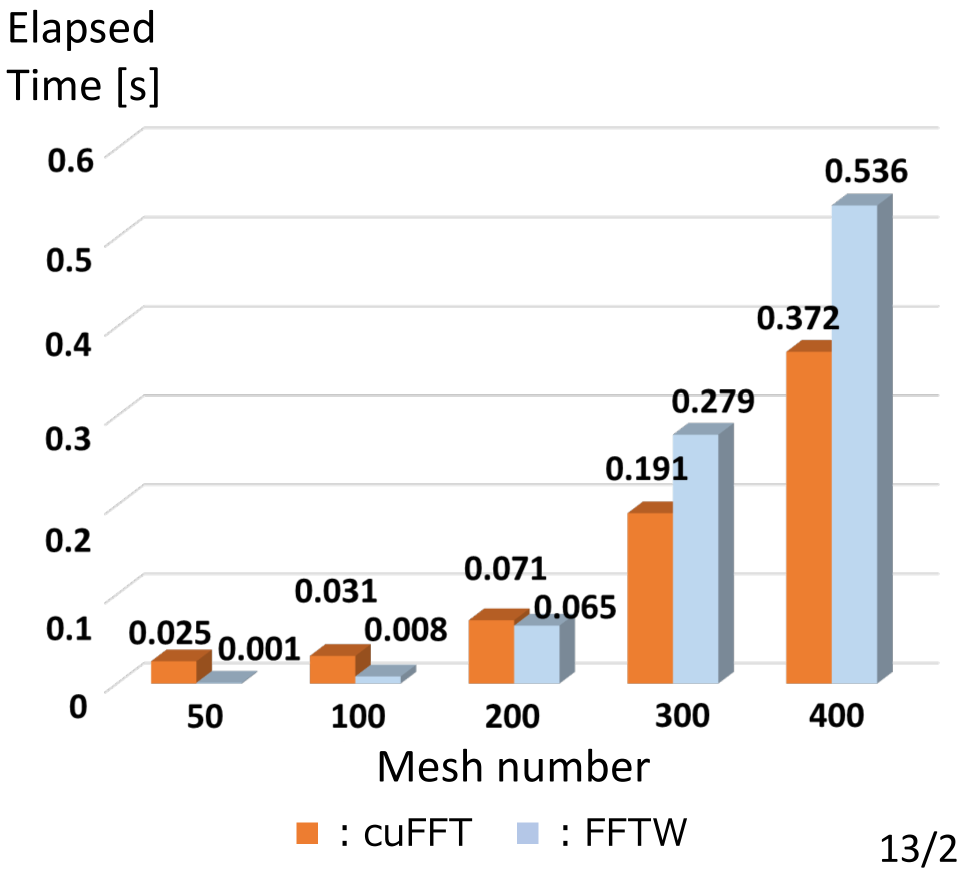
✓ The direct GPU-GPU data transfer has not been successfully installed, but the data communication time of Poisson solver is originally small.

Installation of cuFFT - 1

- ✓ In the original GKNET, FFTW is used for FFT calculation on CPUs. To accelerate this part, cuFFT is installed.

- ✓ As is shown by the simple 3D FFT test below, the efficiency of cuFFT is confirmed only in the large-size problem.

- ✓ But, taking into account for the CPU-GPU data transfer for using FFTW, cuFFT is expected to be faster than FFTW even in the small-size problem.



Installation of cuFFT - 2

Call of cuFFT

```
…

ierr1  = cufftPlanMany(plan, 1, …, CUFFT_Z2Z, Nbatch)
   if (ierr1 /= CUFFT_SUCCESS) then
     print *, 'cufftPlanMany: error', ierr1
   end if

…

!$acc host_data use_device(in, out)
ierr1 = cufftExecZ2Z(plan, in, out, CUFFT_FORWARD)
if (ierr1 /= CUFFT_SUCCESS) then
print *, 'cufftExecC2C: error', ierr1
end if
!$acc end host_data

…

ierr1 = cufftDestroy(plan)
   if (ierr1 /= CUFFT_SUCCESS) then
     print *, 'cufftDestroy: error', ierr1
   end if
```

✓ Since we need 1D FFT for 3D data as follows, batched cuFFT is utilized.

$$f\left(x, k_y, z\right) = \sum_i f\left(x, y_i, z\right) \exp\left(i k_y y_i\right)$$

✓ In this case, Nbatch=N_x*N_z is set.

✓ In addition, direct GPU-GPU data transfer is also utilized.

Installation of cuFFT - 3

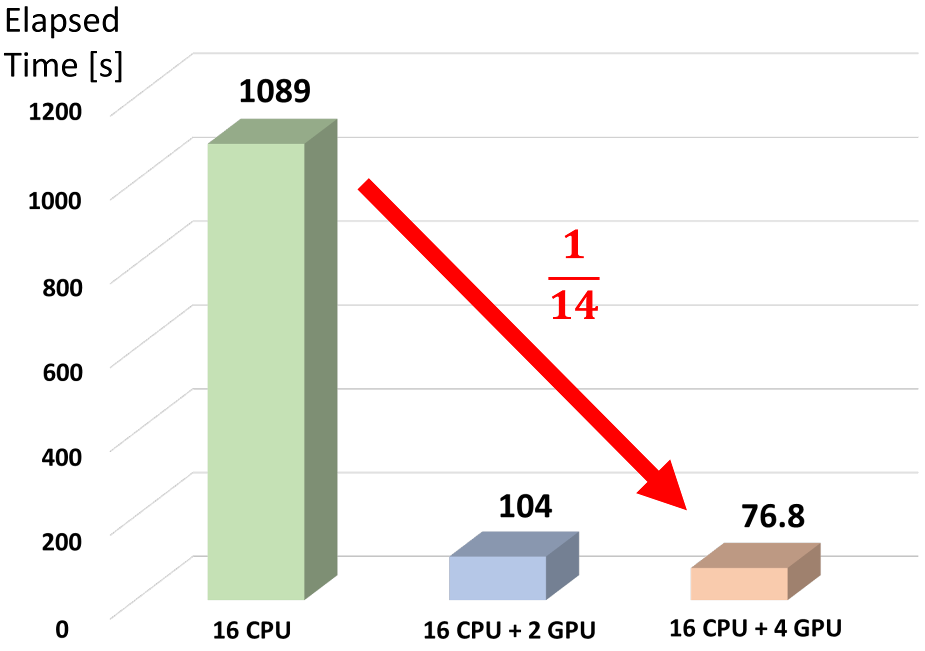
Benchmark test of Vlasov solver

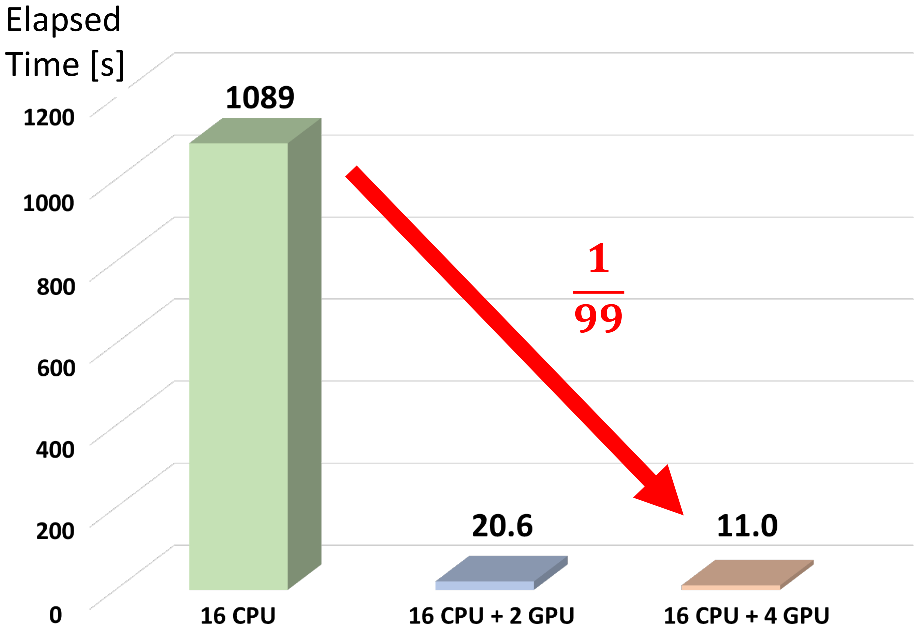
FFTW

	16CPU	16CPU+1GPU	16CPU+4GPU
Calculation	1089	104	76.8
Communication	15.1	11.3	9.83

cuFFT

	16CPU	16CPU+1GPU	16CPU+4GPU
Calculation	1089	20.6	11.0
Communication	15.1	11.3	9.83





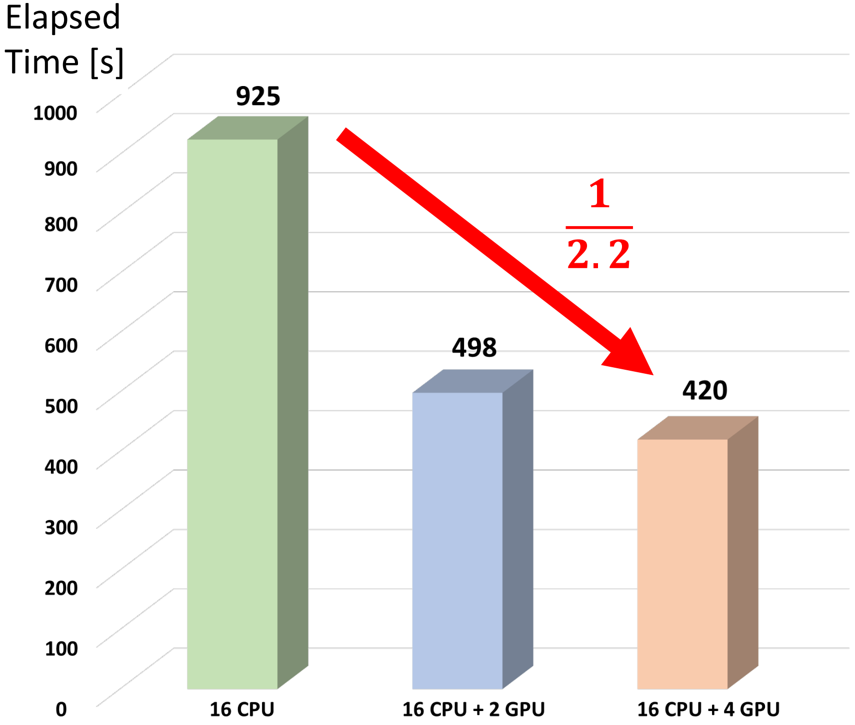
- ✓ The problem size in 3D real space is relatively low, but the calculation time is largely reduced by the installation of cuFFT because the CPU-GPU data transfer is skipped.

Installation of cuFFT - 4

Benchmark test of Poisson solver

FFTW

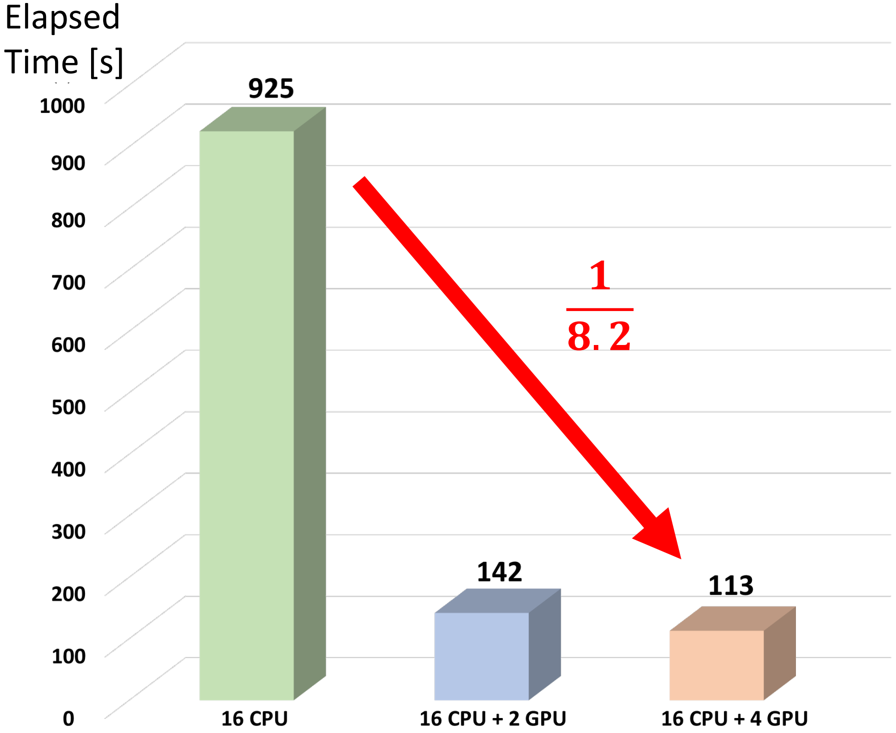
	16CPU	16CPU+2GPU	16CPU+4GPU
Calculation	925	498	420
Communication	7.29	3.48	2.40



$\frac{1}{2.2}$

cuFFT

	16CPU	16CPU+2GPU	16CPU+4GPU
Calculation	925	142	113
Communication	7.29	3.48	2.40

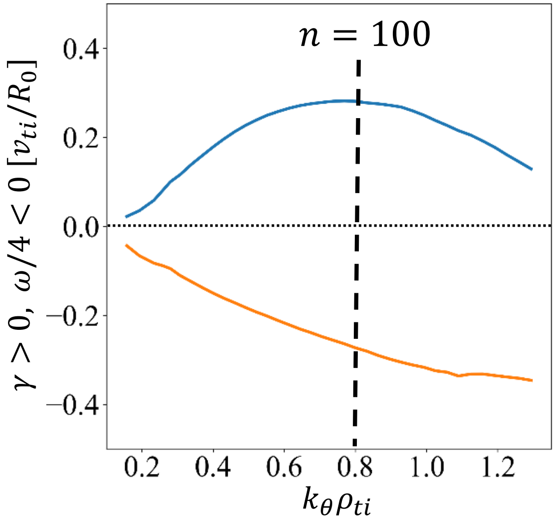


$\frac{1}{8.2}$

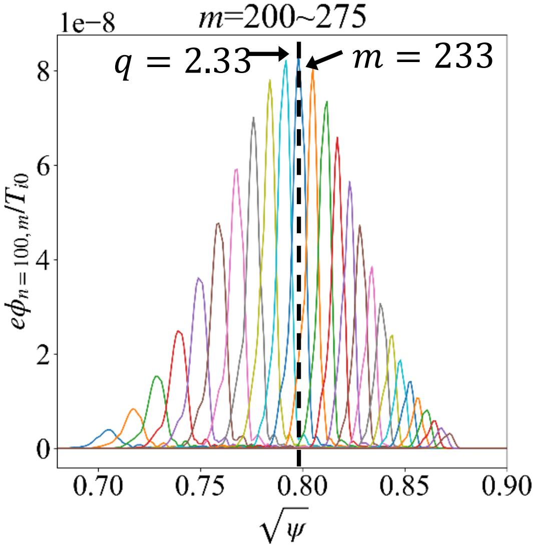
- ✓ Same tendency can be observed but the speed-up rate is still low. This is considered to originate from the 1D matrix solver for the Jacobi method.

Nonlinear Simulation of JT-60SA Plasma - 1

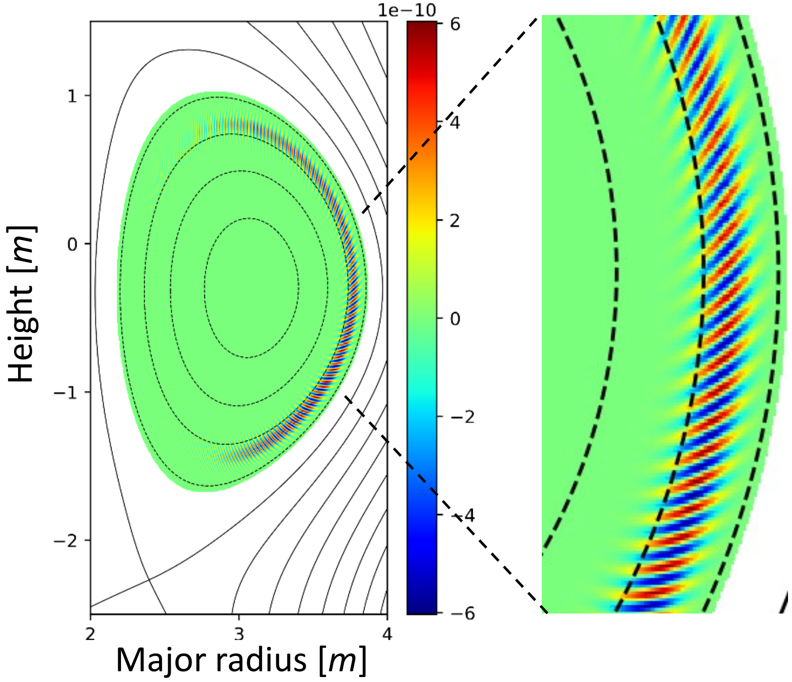
Dispersion relation



The poloidal harmonics of $\phi_{n=100}$



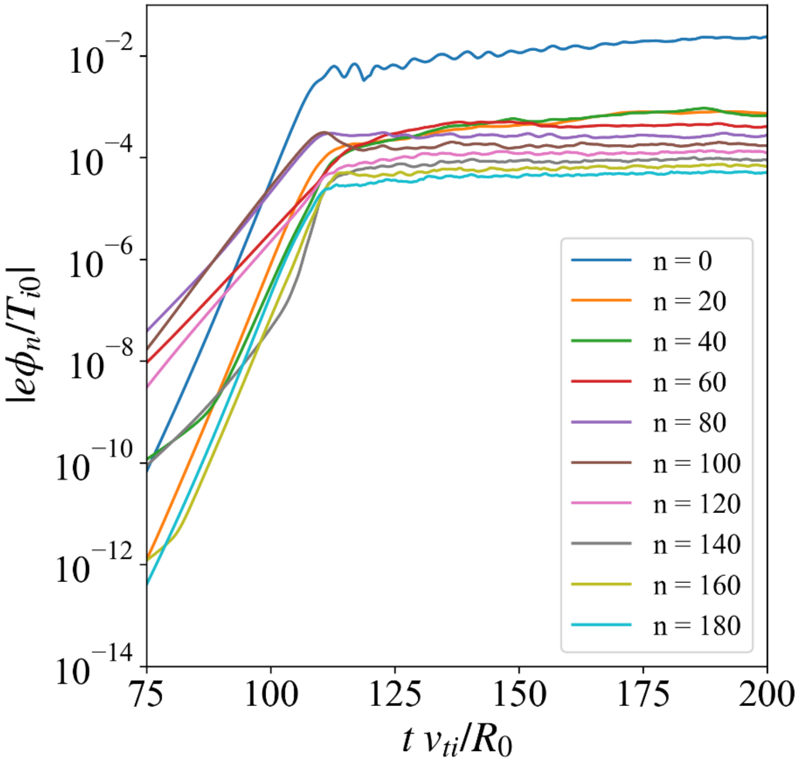
$\phi_{n=100}$ in a linear phase



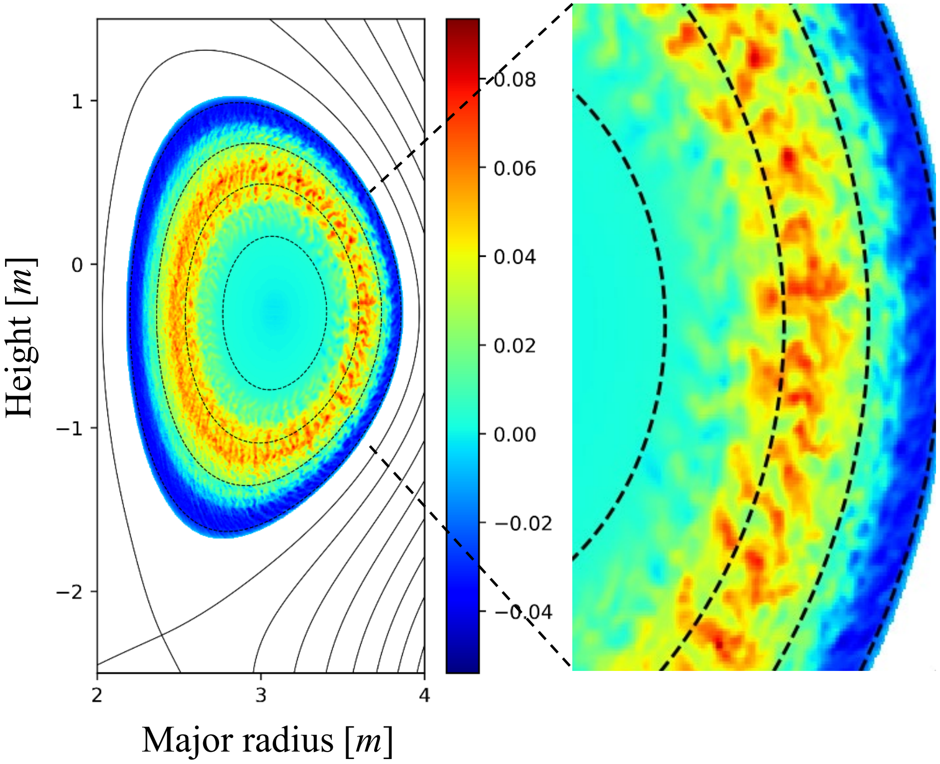
- ✓ The system size is $a_0/\rho_{ti0} = 294$ and the applied mesh number in real space is (720, 256, 32), respectively.
- ✓ Very high poloidal mode number instabilities, such as $(m, n) = (233, 100)$, have been resolved.

Nonlinear Simulation of JT-60SA Plasma - 2

Time development of the amplitude of the electrostatic potential



$\phi(\zeta = 0)$ after the nonlinear saturation



✓ The zonal flow has been generated at the position corresponding to the linear instabilities.

Nonlinear Simulation of JT-60SA Plasma - 3

- ✓ Consider the number of meshes required to resolve the mode $m_{\mathrm{res}} = 160 \times 2.3 \approx 370$ resonating with $n = 160$, which is unstable in this case.

- ✓ Assuming that 8 times as many meshes as the mode number are required, we need $N_\theta = 370 \times 8 \approx 30$ in the flux-surface coordinate system (ρ, θ, ζ).

- ✓ On the other hand, in the field aligned coordinate system (x, y, z), $N_z = 32$ is enough as this case.

- ✓ By utilizing the field aligned coordinate system , the required meshes are reduced to $3000 \div 32 \approx 1/94$.

Summary - 1

Summary-1

- ✓ We have accelerated global gyrokinetic code by using OpenACC directives.
- ✓ Especially we have tried (A)the utilization of multi-GPU, (B) the direct GPU-GPU data transfer, (C) the installation of cuFFT.
- ✓ We have achieved 83 times speed-up for Vlasov solver and 9.1 times speed-up for Poisson solver.
- ✓ As the result, the total speed-up rate was 13.

	Vlasov		NEUTRAL		
	Calc.	Comm.	Calc.	Comm.	Total
MPI	1089	15.1	925	7.29	2036
Multi-GPU	76.8	24.2	420	2.40	523
Multi-GPU+ Direct transfer	76.8	9.83	420	2.40	509
Multi-GPU+ Direct transfer + cuFFT	11.0	9.83	130	2.40	153

$\frac{1}{13}$

Summary - 2

Summary-2

- ✓ We have also implemented a field-aligned coordinate system to GKNET.
- ✓ Realistic tokamak geometries, including up-down asymmetric equilibria, have been also implemented.
- ✓ The result shows that the linear ITG instability with high poloidal modes and resultant zonal flow generation are properly traced. In this case, it is estimated that the number of computational grids can be reduced to 1/94 compared to that of the flux surface coordinate system.

Future plans

- ✓ The development of GKNET will be extended to address tokamak edge turbulence. Currently, an interface code is being developed in the SOL/divertor region.
- ✓ In addition, we will resolve some issues for GPU acceleration (Matrix solver, GPU-GPU direct transfer, etc.)