Multi-Node Multi-GPU Implementation of Micromagnetic Model for Spintronics Devices

TOSHIBA Leading Innovation

Kiwamu Kudo, Hirofumi Suto, Tazumi Nagasawa, Koichi Mizushima, and Rie Sato

Corporate Research & Development Center, Toshiba Corporation

Why Multi-Node & GPUs Micromagnetics?

- Micromagnetic (μMag) simulation is useful for developing spintronics devices, such as MRAMs, spin-torque oscillators (STOs), and HDDs.
- Large-scale & high-speed μMag simulator is indispensable for elucidating physical phenomena in such spintronics devices, e.g., inter-magnet interactions in MRAM cells array, STOs array, HDDs' recording media.
- The simplest implementation of μMag model is the finite-difference method (FDM) and has been widely used[1], mainly on CPU-cluster computing system.
- GPUs have been recently used to speed up FDM-based μMag simulations, and there are several open-source and commercial GPU-used simulators[2,3,4].
- In the conventional GPU-used μMag simulators, the number of spatially discretized meshes is limited (typically a few million) because of a single GPU memory capacity (e.g., 6GB for Tesla K20X).

➡ FDM-based μMag simulation code using "MPI+CUDA" on multiple node & multiple GPU computing system

Simulation Codes

- Our simple FDM-based μMag code solves the Landau-Lifshitz-Gilbert-Slonczewski (LLGS) equation for the magnetization vector field $\boldsymbol{m}(\boldsymbol{r},t)$:

$$\frac{\partial \boldsymbol{m}(\boldsymbol{r},t)}{\partial t} = -\gamma \boldsymbol{m}(\boldsymbol{r},t) \times \boldsymbol{H}_{\mathrm{eff}}(\boldsymbol{r},t) + \alpha(\boldsymbol{r},t)\boldsymbol{m}(\boldsymbol{r},t) \times \frac{\partial \boldsymbol{m}(\boldsymbol{r},t)}{\partial t}$$

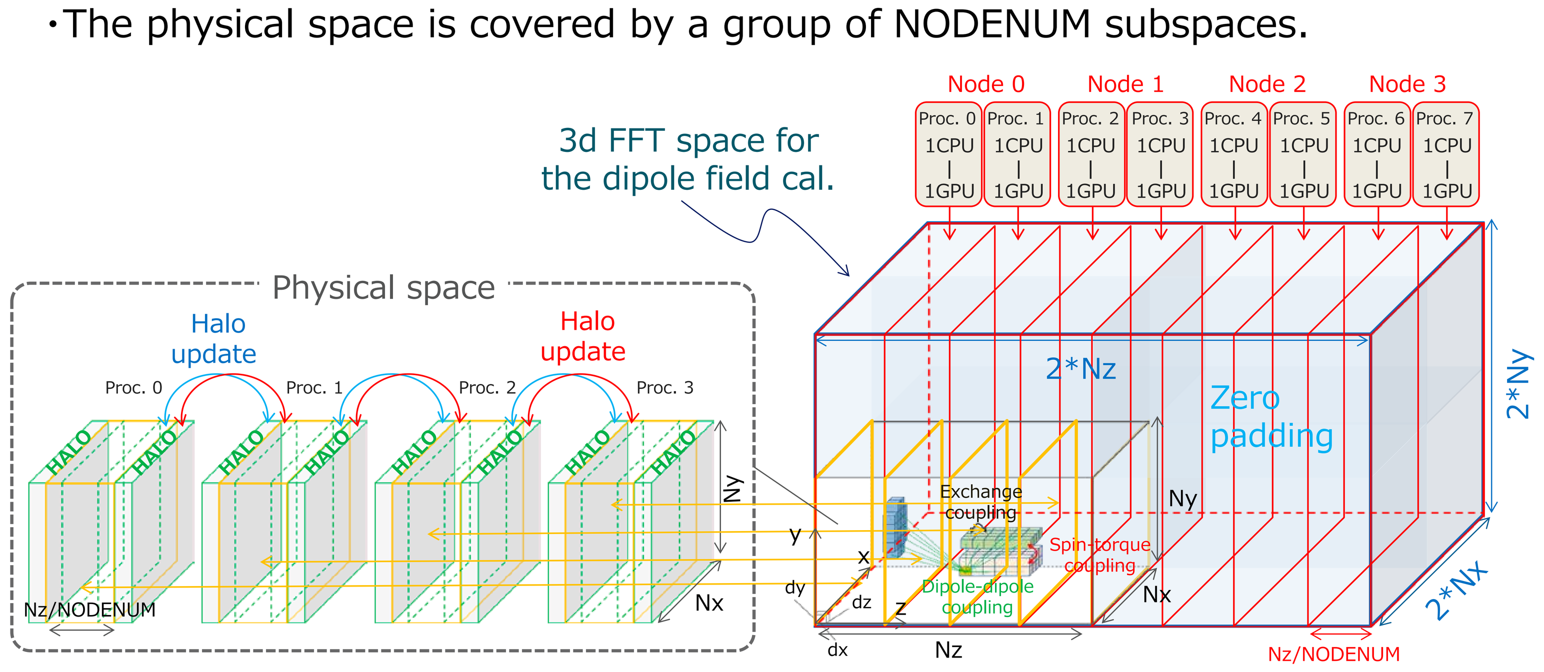
- The effective field, $\boldsymbol{H}_{\mathrm{eff}}(\boldsymbol{r},t)$, includes the external magnetic field, the dipole field, the exchange coupling, and the spin-torque coupling, etc.
- Various types of nanomagnets can be placed in a physical space.
- The physical space is discretized into Nx x Ny x Nz rectangular meshes.
- Heun's method (2nd-order Runge-Kutta method) is used for the time evolution of the LLGS equation.
- Fourier transform method is used to calculate the dipole magnetic field. The "3d-FFT space" of (2*Nx)*(2*Ny)*(2*Nz) meshes with zero padding is secured for the calculation.

Nodes & GPUs Allocation

・For a simple implementation on 2 CPUs—2 GPUs/node environment, the 3d-FFT space is partitioned in the z direction into 2*NODENUM subspaces.

・1CPU—1GPU pair controlled by an MPI process is allocated to each subspace.

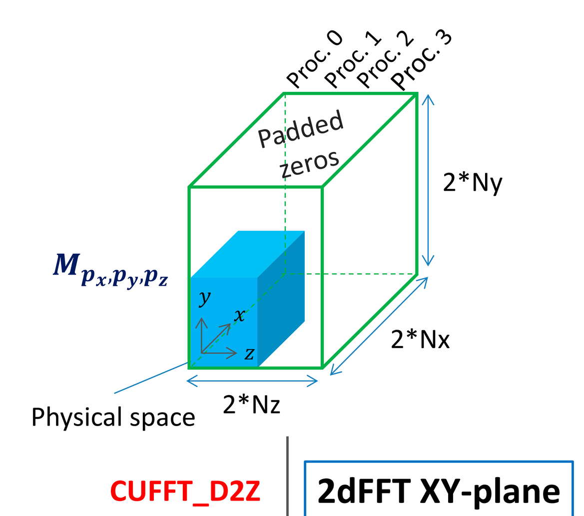
・The physical space is covered by a group of NODENUM subspaces.



[Example of hardware allocation (NODENUM=4)]

Dipole Field Calculation Using 3dFFT

2dFFT→Transpose→1dFFT → Multiplication →1dFFT→Transpose→2dFFT



Calculation Procedure of the Dipole Field $\boldsymbol{H}_{p_x,p_y,p_z}$

Forward FFT of magnetization vector (x, y, z components):

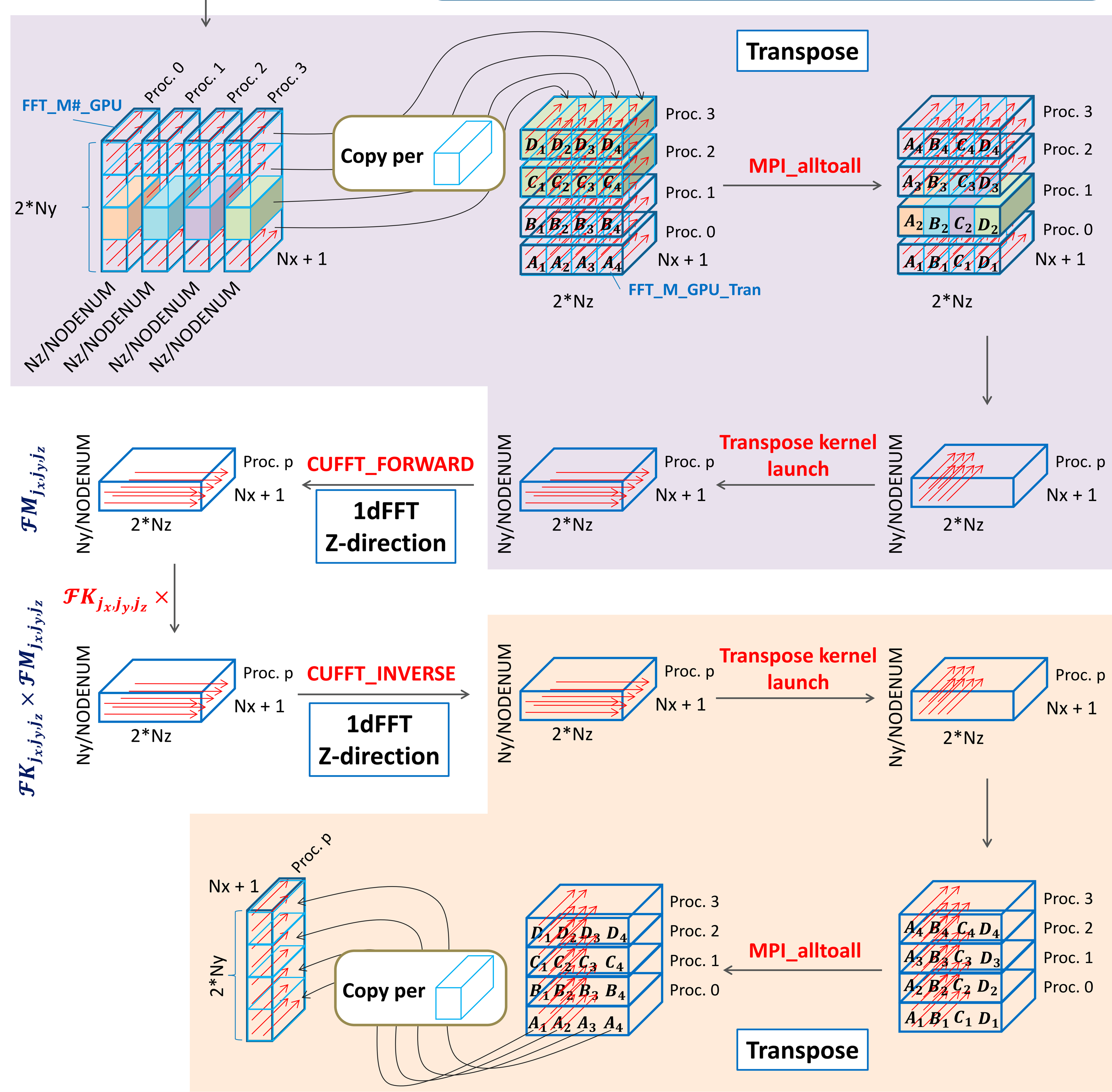
$$\mathcal{F}\boldsymbol{M}_{j_x,j_y,j_z} = \sum_{p_z=0}^{2N_z-1}\sum_{p_y=0}^{2N_y-1}\sum_{p_x=0}^{2N_x-1} \boldsymbol{M}_{p_x,p_y,p_z}\, e^{-2\pi i(j_x p_x/2N_x + j_y p_y/2N_y + j_z p_z/2N_z)}$$

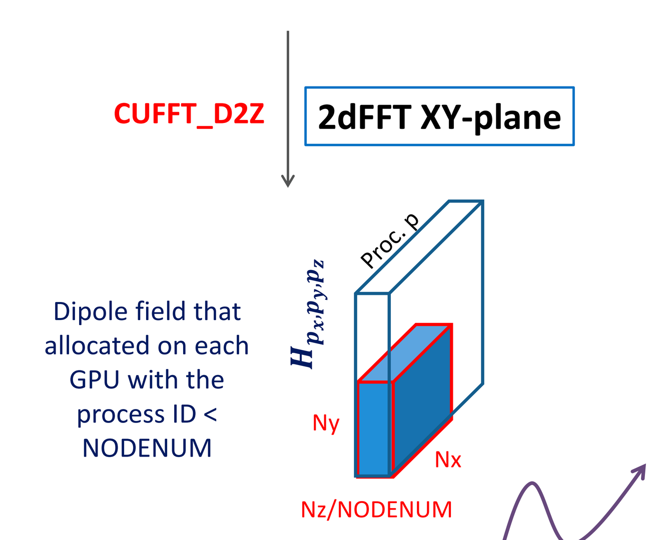
Inverse FFT (x, y, z components):

Fourier transformed demagnetizing tensor

$$\boldsymbol{H}_{p_x,p_y,p_z} = \frac{1}{8N_xN_yN_z}\sum_{j_z=0}^{2N_z-1}\sum_{j_y=0}^{2N_y-1}\sum_{j_x=0}^{2N_x-1} \mathcal{F}K_{j_x,j_y,j_z}\mathcal{F}\boldsymbol{M}_{j_x,j_y,j_z}\, e^{2\pi i(j_x p_x/2N_x + j_y p_y/2N_y + j_z p_z/2N_z)}$$

This dipole field calculation needs the heaviest computational cost in the μMag code because of the all-to-all communications between GPUs.





Transpose kernel

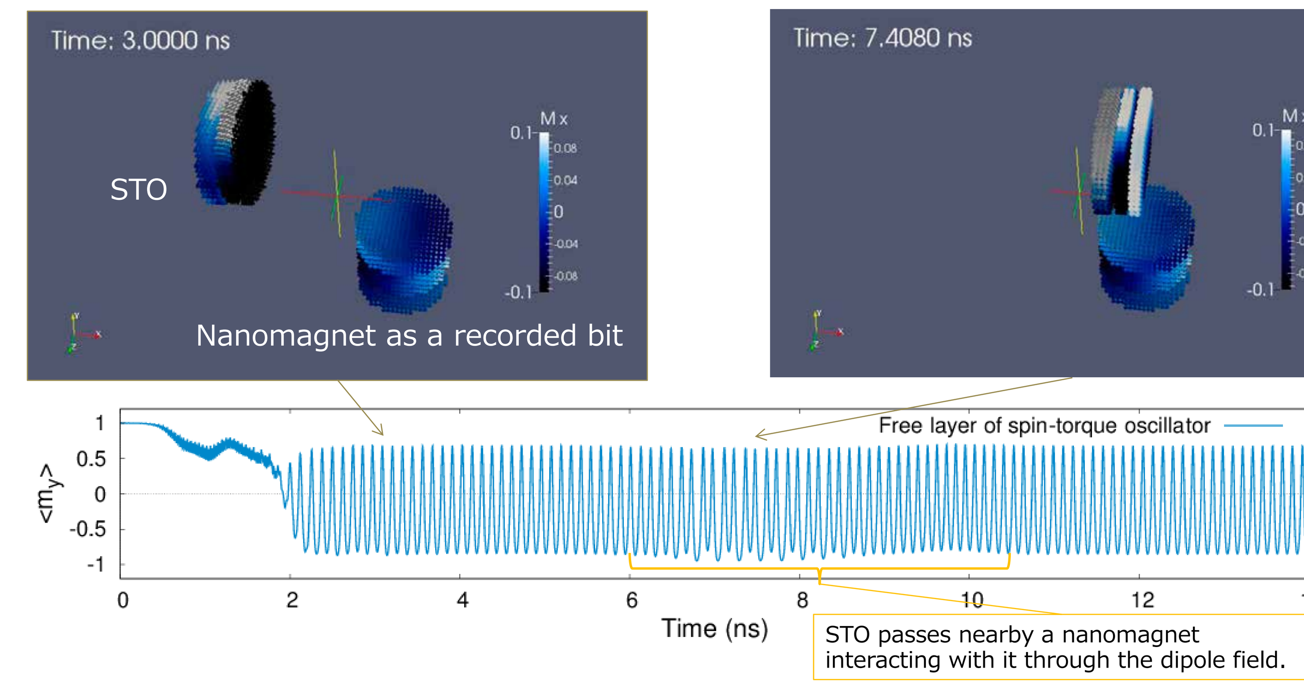
```
__global__ void transposeCoalesced3D_ZXY
(cufftDoubleComplex *odata, cufftDoubleComplex *idata,
int width, int height, int depth){

    __shared__ cufftDoubleComplex tile[TILE_DIM][TILE_DIM+1];

    int xIndex = blockIdx.x * blockDim.x   + threadIdx.x;
    int yIndex = blockIdx.y * blockDim.y   + threadIdx.y;
    int zIndex = blockIdx.z * blockDim.z   + threadIdx.z;

    int xIndexOut = blockIdx.x * blockDim.x   + threadIdx.z;
    int zIndexOut = blockIdx.z * blockDim.z   + threadIdx.x;

    if ( xIndex < width && zIndex < depth ){
        int index_in  = xIndex + yIndex*width + zIndex*width*height;
        tile[threadIdx.x][threadIdx.z] = idata[index_in];
    }
    __syncthreads();

    if ( xIndexOut < width && zIndexOut < depth ){
        int index_out = zIndexOut + xIndexOut*depth + yIndex*depth*width;
        odata[index_out] = tile[threadIdx.z][threadIdx.x];
     }
}
```

This kernel is a 3d extension of a sample code described in CUDA BEST PRACTICE GUIDE v5.0. The Grids & Blocks are: <<<GridDim(Round((Nx+1),TILE_DIM),Ny/NODENUM,Round(2*Nz,TILE_DIM)),BlockDim(TILE_DIM,1,TILE_DIM)>>>.

Functionality Check

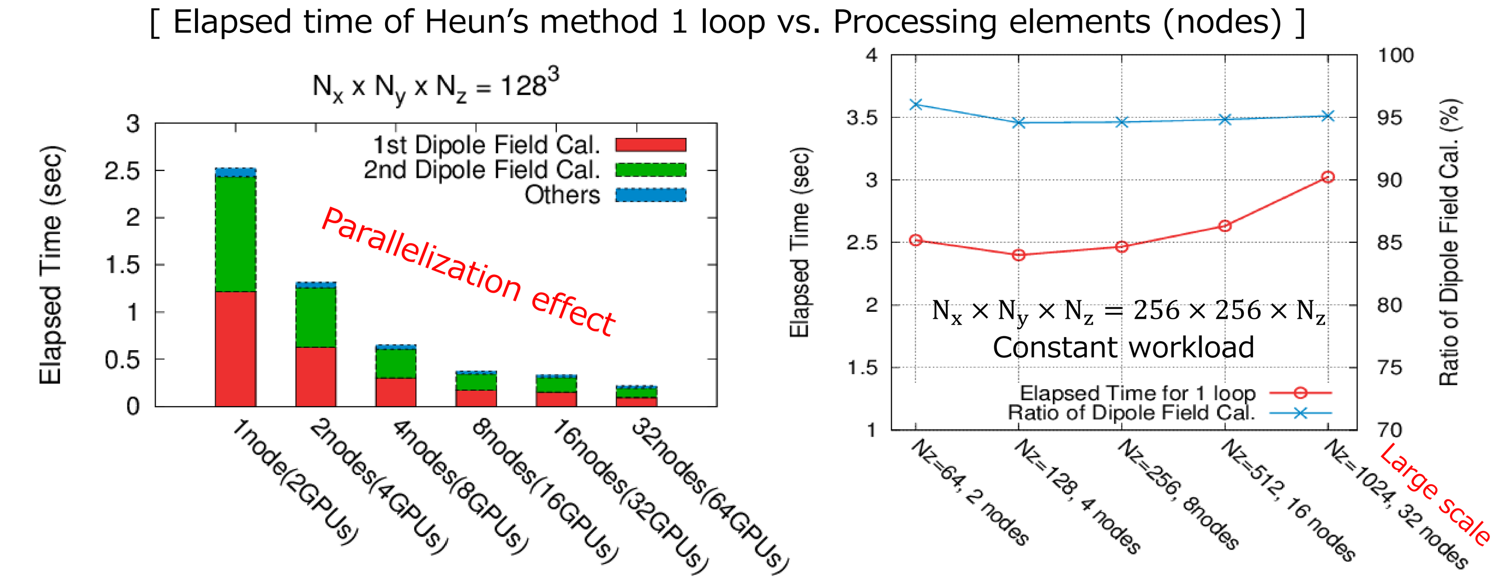
Simulations of magnetization dynamics in several magnetic system verify that μMag models with various couplings are implemented correctly.

Example:
Simulation of a spin-torque oscillator (STO) interacting with a nanomagnet toward the development of next-generation magnetic recording architecture[5,6].



Performance Measurement

[Elapsed time of Heun's method 1 loop vs. Processing elements (nodes)]



Benchmark Environment

IBM System x iDataPlex dx360 M4 computing server
・2 x Intel® Xeon® Processor E5-2690 @ 2.9 GHz / node
・192 GB RAM / node
・2 x NVIDIA® Tesla® K20X GPU / node
・Mellanox InfiniBand SX6536

Software Environment
・RHEL6.3
・NVIDIA CUDA 5.0
・Intel C 13.1
・Intel MPI Library 4.1 for linux

- Significant parallelization effect is validated, but the communication overhead degrades the strong scaling at large processing elements.
- The dipole field calculation occupies 95% of total computing time.
- Large-scale simulations are available.

We have developed a scalable Multi-GPU μMag-simulation code. The implemented code enables us to perform large-scale simulations far beyond single-GPU limitations.

References

[1] M. J. Donahue and D. G. Porter, OOMMF Object Oriented MicroMagnetic Framework, software NIST (2004).
[2] "MicroMagnum." http://magnum.physnet.uni-hamburg.de.
[3] "GPMagnet": L. Lopez-Diaz et al., J. Phys. D: Appl. Phys. **45**, 323001 (2012).
[4] "muMax3": A. Vansteenkiste et al., arXiv:1406.7635.
[5] K. Kudo, T. Nagasawa, K. Mizushima, H. Suto, and R. Sato, Appl. Phys. Express **3**, 043002 (2010).
[6] H. Suto, T. Nagasawa, K. Kudo, K. Mizushima, and R. Sato, Nanotechnology **25**, 245501 (2014).

Acknowlegements

This work is partially supported by Strategic Promotion of Innovative Research and Development from Japan Science and Technology Agency, JST. K.K. would like to thank Afumado Fatourafuman for useful discussion on the programming.