KEMP: Korea University's ElectroMagnetic Propagator

Nano Optics Lab.

Myung-Su Seok

Introduction of KEMP

1. FDTD simulation software
   FDTD (finite-difference time-domain) : Numerical analysis technique used for modeling computational electrodynamics

2. using CPU & GPU as computing device
   (KEMP is designed to support various computing devices)
   programming language for core engine: C (CPU), CUDA (NVIDIA GPU), OpenCL(AMD GPU, Intel MIC)
   Optimization of CPU, NVIDIA GPU : completed
   AMD GPU : available
   Intel MIC : to be supported later

3. Specialized to massive computation
   Enabled Multi-node computing by using MPI(Massage Passing Interface)
   Optimized parallel overhead by overlapping computation method

4. Python package
   KEMP can be used by writing and executing python script
   GUI : to be supported later

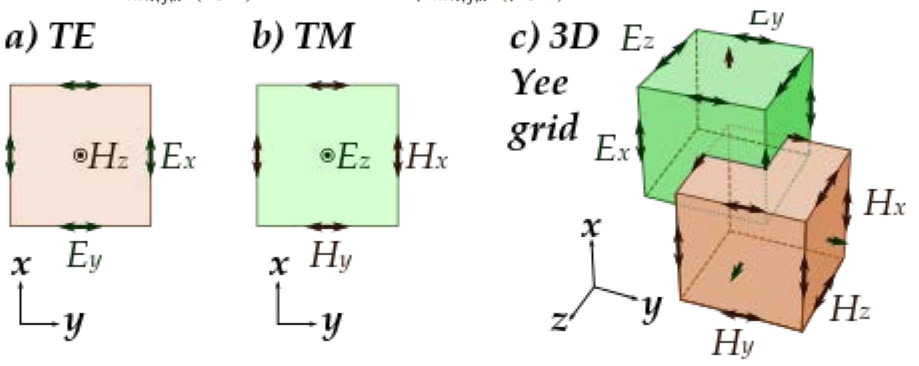
$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t},$$

$$\nabla \times \mathbf{H} = \epsilon \frac{\partial \mathbf{E}}{\partial t},$$

$$E_x|^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}} = E_x|^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}} + CE_x|_{i,j+\frac{1}{2},k+\frac{1}{2}}\left(H_z|^{n}_{i,j+1,k+\frac{1}{2}} - H_z|^{n}_{i,j,k+\frac{1}{2}} - H_y|^{n}_{i,j+\frac{1}{2},k+1} + H_y|^{n}_{i,j+\frac{1}{2},k}\right),$$

$$H_x|^{n+1}_{i+\frac{1}{2},j,k} = H_x|^{n}_{i+\frac{1}{2},j,k} + CH_x|_{i+\frac{1}{2},j,k}\left(E_z|^{n+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k} - E_z|^{n+\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2},k} - E_y|^{n+\frac{1}{2}}_{i+\frac{1}{2},j,k+\frac{1}{2}} + E_y|^{n+\frac{1}{2}}_{i+\frac{1}{2},j,k-\frac{1}{2}}\right),$$

$$CE_x|_{i,j,k} = \frac{1}{\epsilon_{rx}|_{i,j,k}}\left(\frac{\Delta_t}{\epsilon_0 \Delta}\right), \qquad CH_x|_{i,j,k} = \frac{1}{\mu_{rx}|_{i,j,k}}\left(\frac{\Delta_t}{\mu_0 \Delta}\right).$$



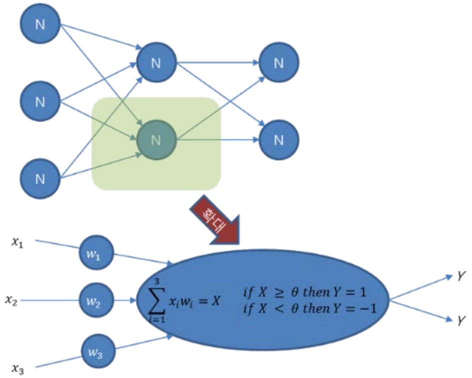
GPGPU: General-Purpose computing on Graphic Processing Units

GPU : Graphic Processing Unit

Designed for graphic computation

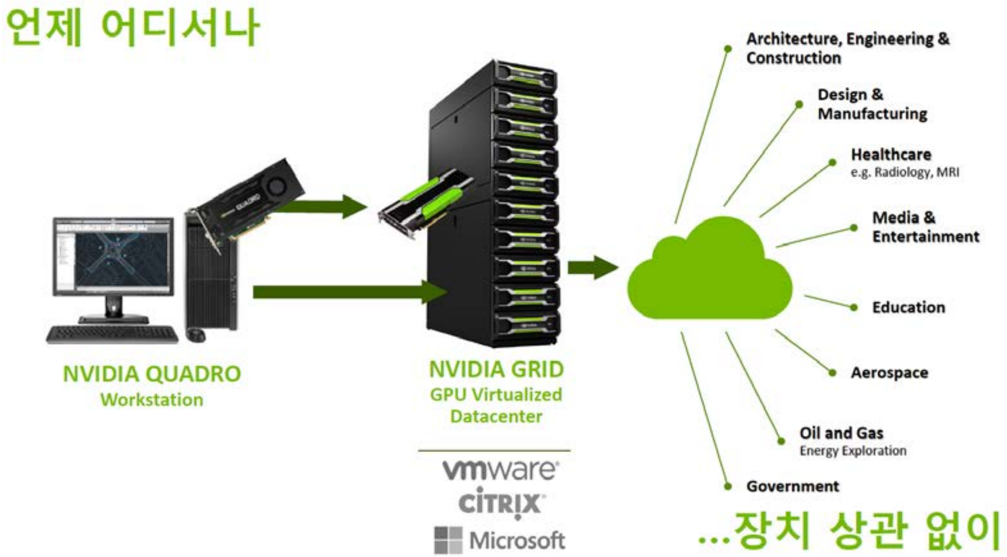
(High-resolution video, game)

-> Specialized High parallel computation


N
N
N
N
N
N
N
N
확대
x_1
x_2
x_3
w_1
w_2
w_3
$\sum_{i=1}^{3} x_i w_i = X$
$if\ X \geq \theta\ then\ Y = 1$
$if\ X < \theta\ then\ Y = -1$
Y
Y

Examples of GPGPU

1. Deep learning
2. GPU accelerated science
3. Cloud (Grid computing) / Virtualization



FDTD acceleration using GPU

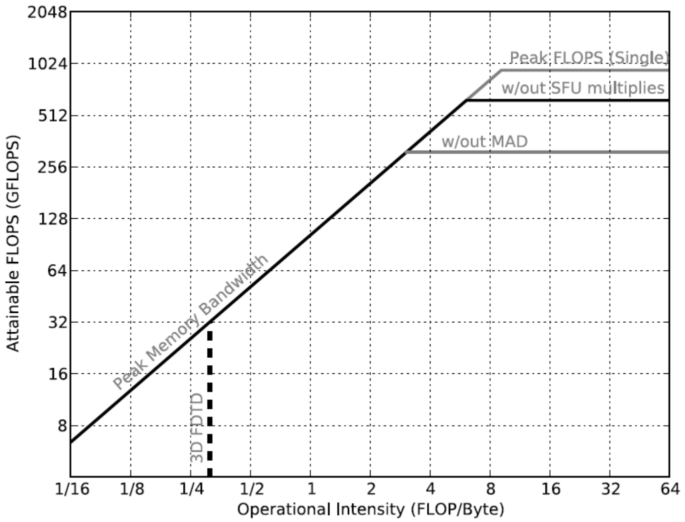
Ki-Hwan Kim, et al.,
Performance analysis and optimization of
three-dimensional FDTD on GPU
using roofline model
Computer Physics Communications 182 (2011)

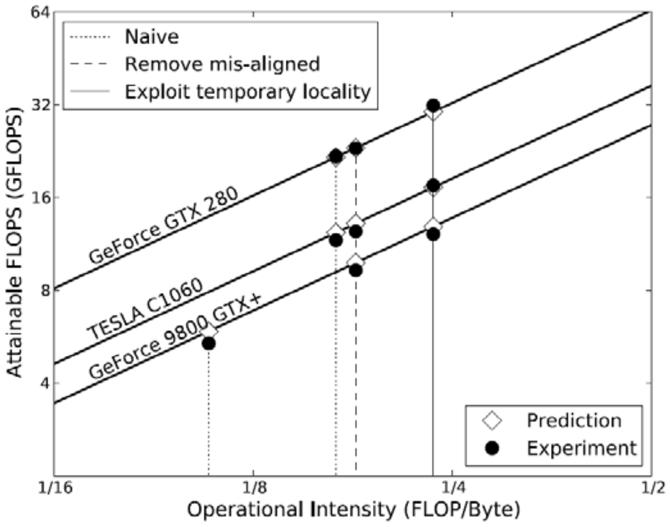
Summary

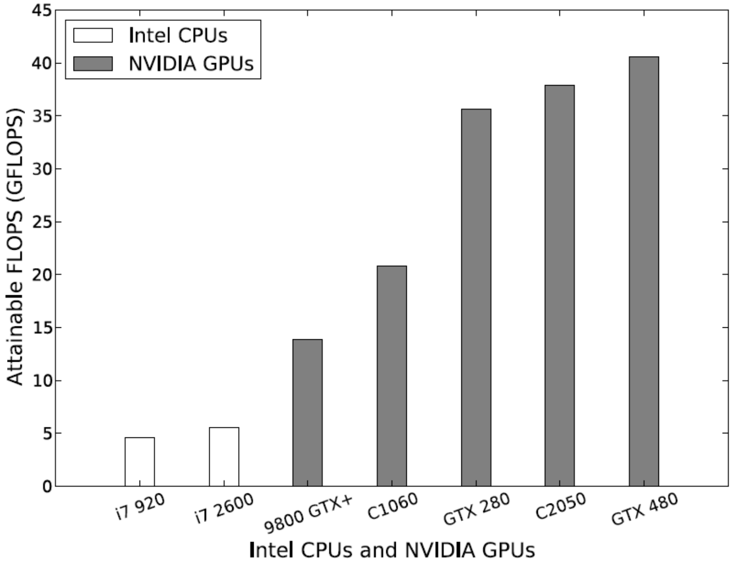
1. High parallelism of FDTD algorithm
-> Enhancement of computation speed

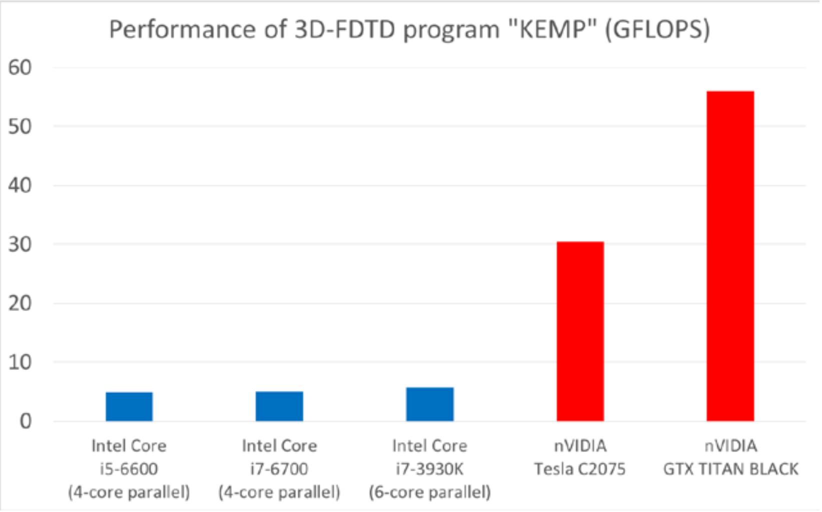
2. Prediction and verification of Theoretical Limit for FDTD computing performance using GPU

KEMP has 90% of the theoretical limit of FDTD computation performance



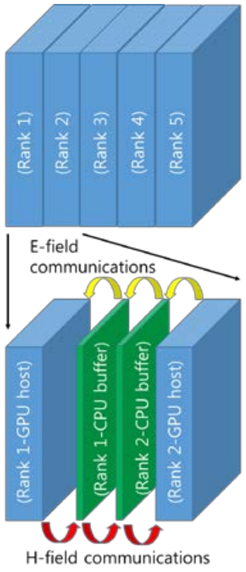


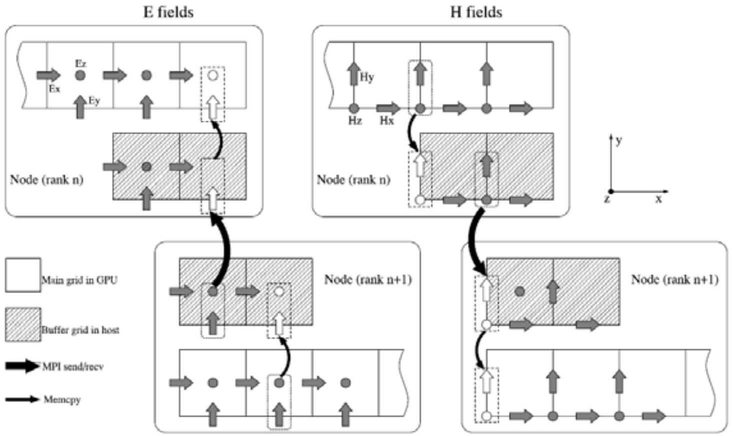


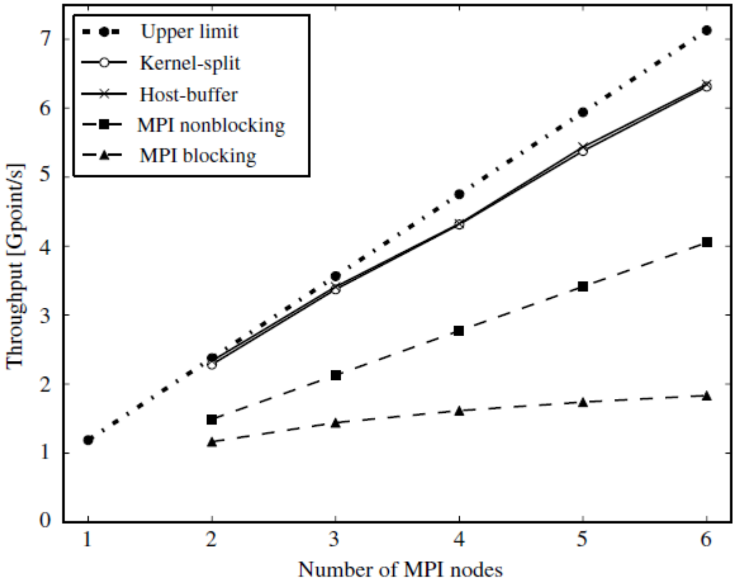


Overlapping computation method

Overlapping Computation time (GPU) & Communication time (Network)
Ki-Hwan Kim, et al.,
Overlapping computation and communication of three-dimensional FDTD on a GPU cluster
Computer Physics Communications 183 (2012)



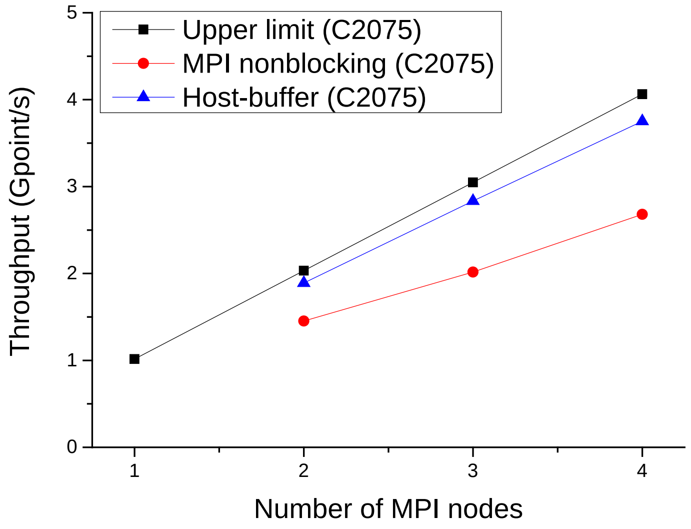


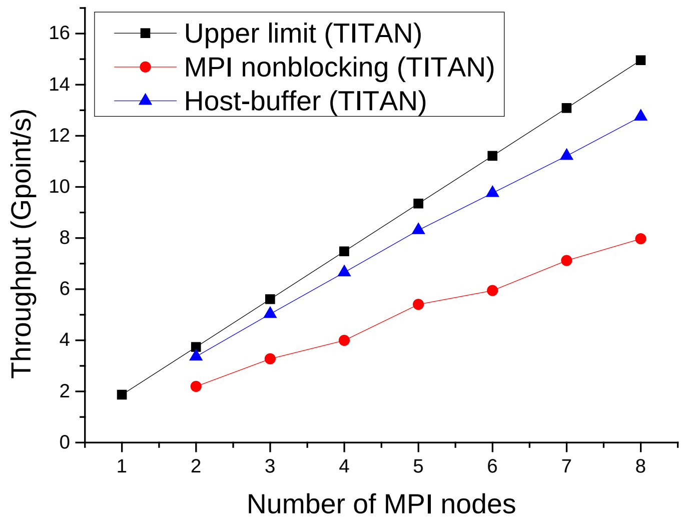


Summary

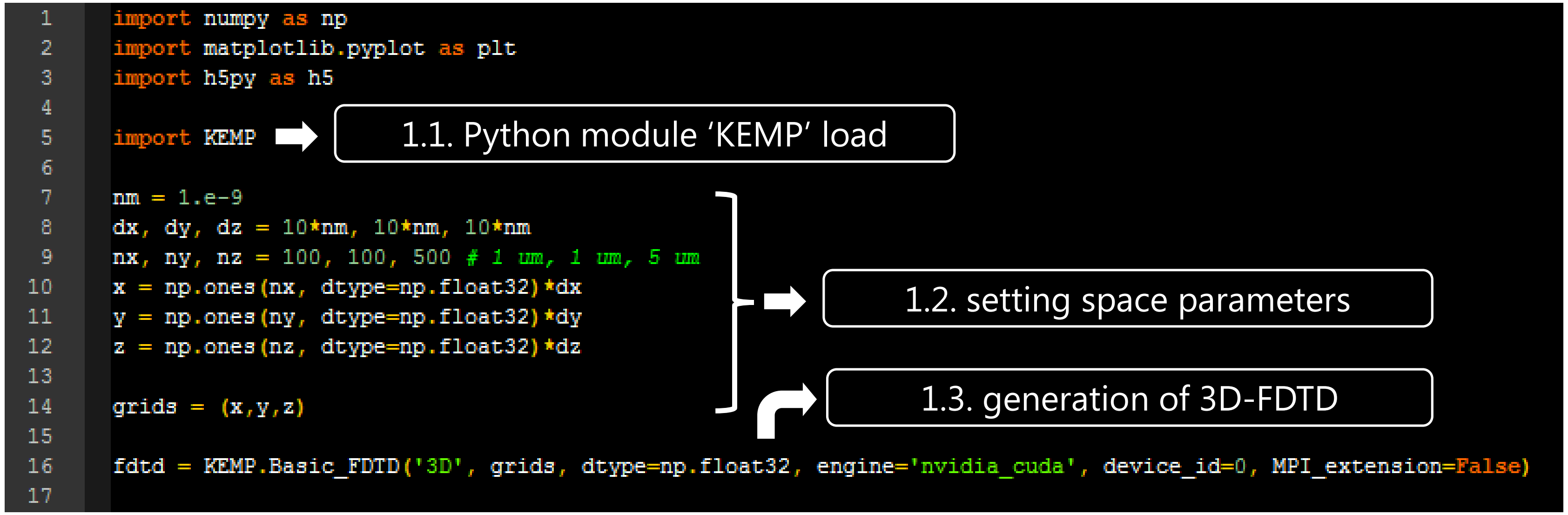
1. boundary communication time main-region computation
-> High overhead from parallel computation with multi-node using GPUs
2. communication and computation can be executed at the same time
-> Theoretical limit of parallel computation can be achieved

KEMP has 85% of the theoretical limit of FDTD computation performance





User interface
1. 3D-FDTD space



```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import h5py as h5
4
5   import KEMP
6
7   nm = 1.e-9
8   dx, dy, dz = 10*nm, 10*nm, 10*nm
9   nx, ny, nz = 100, 100, 500 # 1 um, 1 um, 5 um
10  x = np.ones(nx, dtype=np.float32)*dx
11  y = np.ones(ny, dtype=np.float32)*dy
12  z = np.ones(nz, dtype=np.float32)*dz
13
14  grids = (x,y,z)
15
16  fdtd = KEMP.Basic_FDTD('3D', grids, dtype=np.float32, engine='nvidia_cuda', device_id=0, MPI_extension=False)
17
```

1.1. Python module ‘KEMP’ load

1.2. setting space parameters

1.3. generation of 3D-FDTD

1.1. Python module ‘KEMP’ load

KEMP can be loaded simply by input “import KEMP” (line 5)

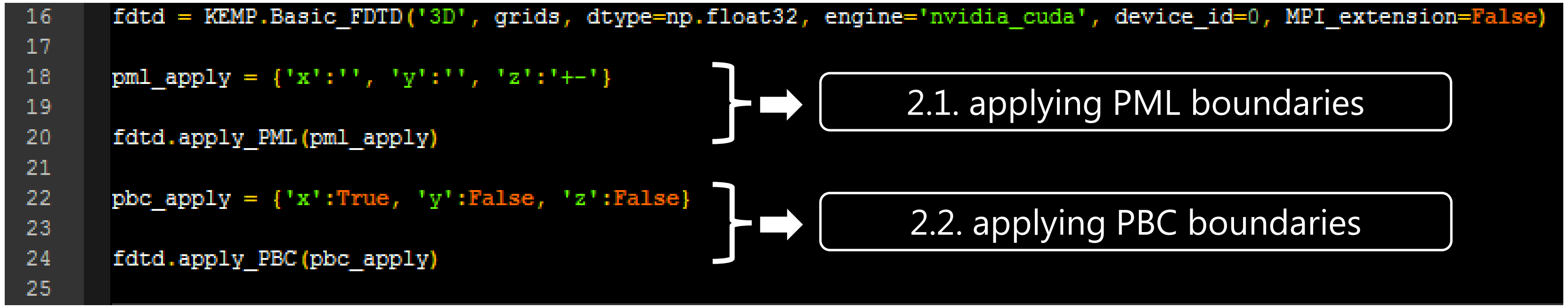
1.2. Setting FDTD space parameters

To set a 3D-FDTD space, minimal unit of spatial length and discritized space grid have to be set up

1.3. generation of 3D-FDTD environment

Generate 3D-FDTD environment using information of space grid and field data type (np.float32 means single-precision float, KEMP supports single and double precision float and complex numbers: numpy.float32, numpy.float64, numpy.complex64, numpy.complex128)

User interface
2. Boundary condition of 3D-FDTD



```
fdtd = KEMP.Basic_FDTD('3D', grids, dtype=np.float32, engine='nvidia_cuda', device_id=0, MPI_extension=False)

pml_apply = {'x':'', 'y':'', 'z':'+-'}

fdtd.apply_PML(pml_apply)

pbc_apply = {'x':True, 'y':False, 'z':False}

fdtd.apply_PBC(pbc_apply)

```

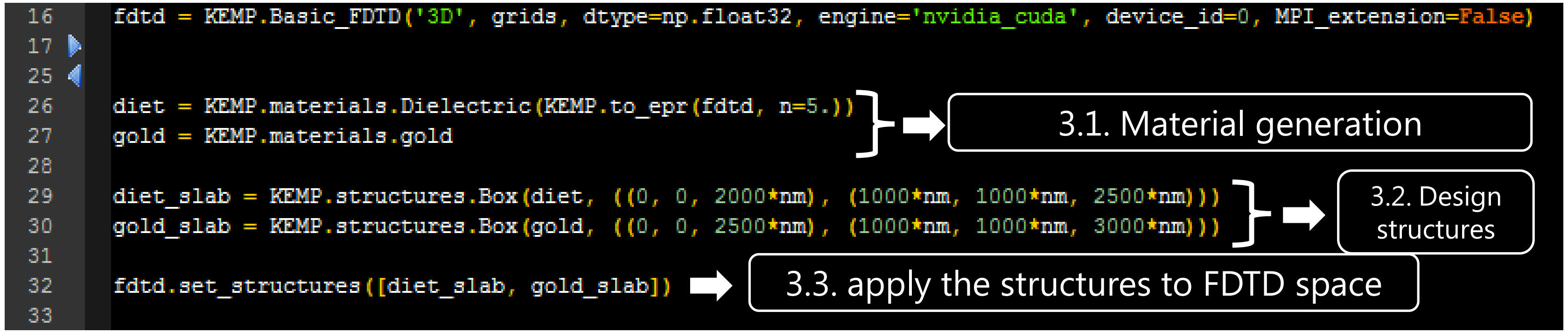
2.1. PML(Perfectly Matched Layer)

Perfectly matched layer is the boundary condition perfectly absorbing incident EM-waves without any reflection. We can set the closed 3D-FDTD space to infinitely open space by setting PML the FDTD space boundaries. PML can be applied to each 3 axes and 2 directions, total 6 boundaries.

2.2. PBC(Periodic Boundary Condition)

To calculate light-matter interaction of infinitely periodic material structure, KEMP support periodic boundary condition. PBC can be applied to axes independently. If the field data type is complex number, PBC is expanded to Bloch boundary condition (BBC) automatically.

User interface
3. Material structures



```
fdtd = KEMP.Basic_FDTD('3D', grids, dtype=np.float32, engine='nvidia_cuda', device_id=0, MPI_extension=False)


diet = KEMP.materials.Dielectric(KEMP.to_epr(fdtd, n=5.))
gold = KEMP.materials.gold

diet_slab = KEMP.structures.Box(diet, ((0, 0, 2000*nm), (1000*nm, 1000*nm, 2500*nm)))
gold_slab = KEMP.structures.Box(gold, ((0, 0, 2500*nm), (1000*nm, 1000*nm, 3000*nm)))

fdtd.set_structures([diet_slab, gold_slab])

```

3.1. Material generation

3.2. Design structures

3.3. apply the structures to FDTD space

3.1. Material generation

KEMP supports various electromagnetic materials : lossy dielectric (complex ε), lossy dimagnetic (complex ε), lossy Dielectromagnetic(ε, μ), electric dispersive meterials(Drude, CP, …). (magnetic dispersive material and electromagnetic dispersive material will be supported later)

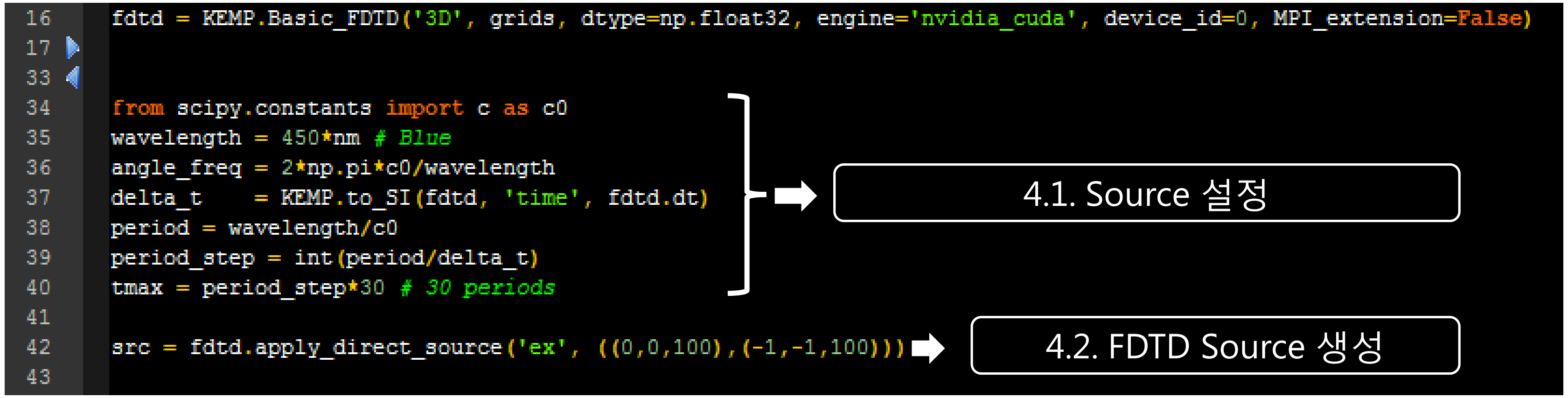
3.2. Design of material structures

We can design the shape of materials at section 3.1. KEMP supports 3D space structures(box, elliptic cylinder, ellipsoid, pyramids).

3.3. Applying prepared structures to 3D-FDTD space.

We can set material structures order and apply to FDTD structures by FDTD_space.set_structures(list_of_structures)

User interface
4. Wave source condition



```
fdtd = KEMP.Basic_FDTD('3D', grids, dtype=np.float32, engine='nvidia_cuda', device_id=0, MPI_extension=False)


from scipy.constants import c as c0
wavelength = 450*nm # Blue
angle_freq = 2*np.pi*c0/wavelength
delta_t     = KEMP.to_SI(fdtd, 'time', fdtd.dt)
period = wavelength/c0
period_step = int(period/delta_t)
tmax = period_step*30 # 30 periods

src = fdtd.apply_direct_source('ex', ((0,0,100),(-1,-1,100)))

```

4.1. Source 설정

4.2. FDTD Source 생성

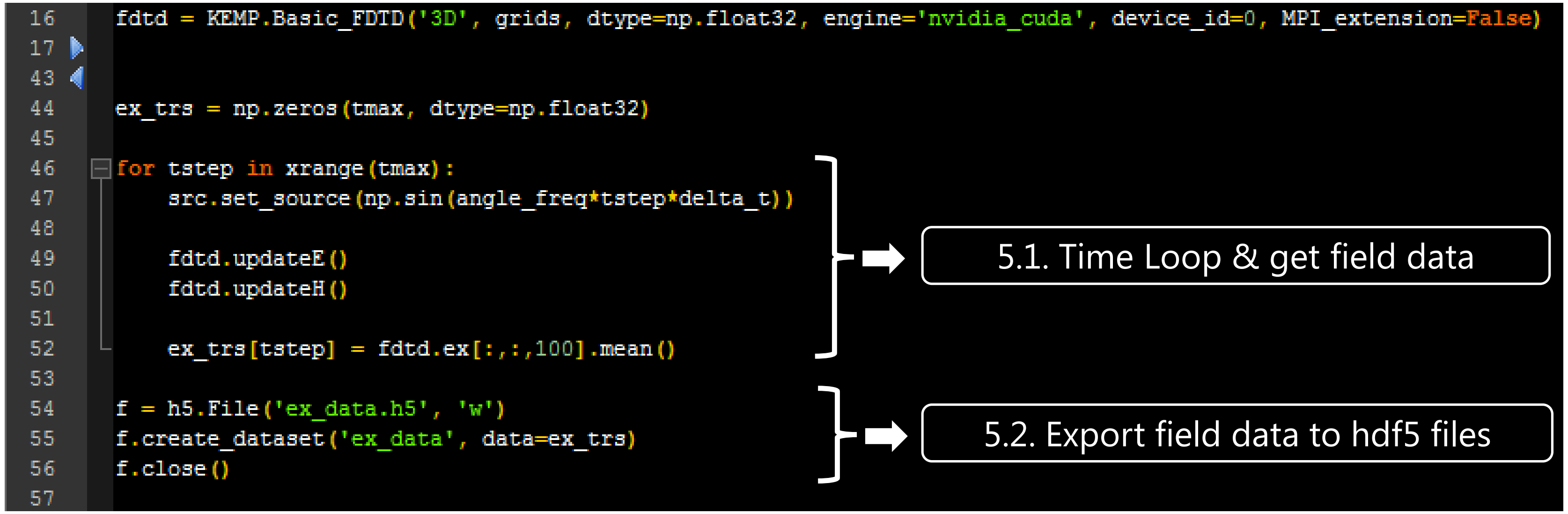
4.1. Source

Incident wave source conditions.

4.2. Setting FDTD Source

We have to set the region of being applied incident wave sources (before time loop calculation).

User interface
5. Time loop + getting field data



```
fdtd = KEMP.Basic_FDTD('3D', grids, dtype=np.float32, engine='nvidia_cuda', device_id=0, MPI_extension=False)


ex_trs = np.zeros(tmax, dtype=np.float32)

for tstep in xrange(tmax):
    src.set_source(np.sin(angle_freq*tstep*delta_t))

    fdtd.updateE()
    fdtd.updateH()

    ex_trs[tstep] = fdtd.ex[:,:,100].mean()

f = h5.File('ex_data.h5', 'w')
f.create_dataset('ex_data', data=ex_trs)
f.close()
```

5.1. Time Loop & get field data

5.2. Export field data to hdf5 files

5.1. FDTD Time Loop & get field data

In the time loop of FDTD calculation, .

(1) applying incident wave source values to field data.

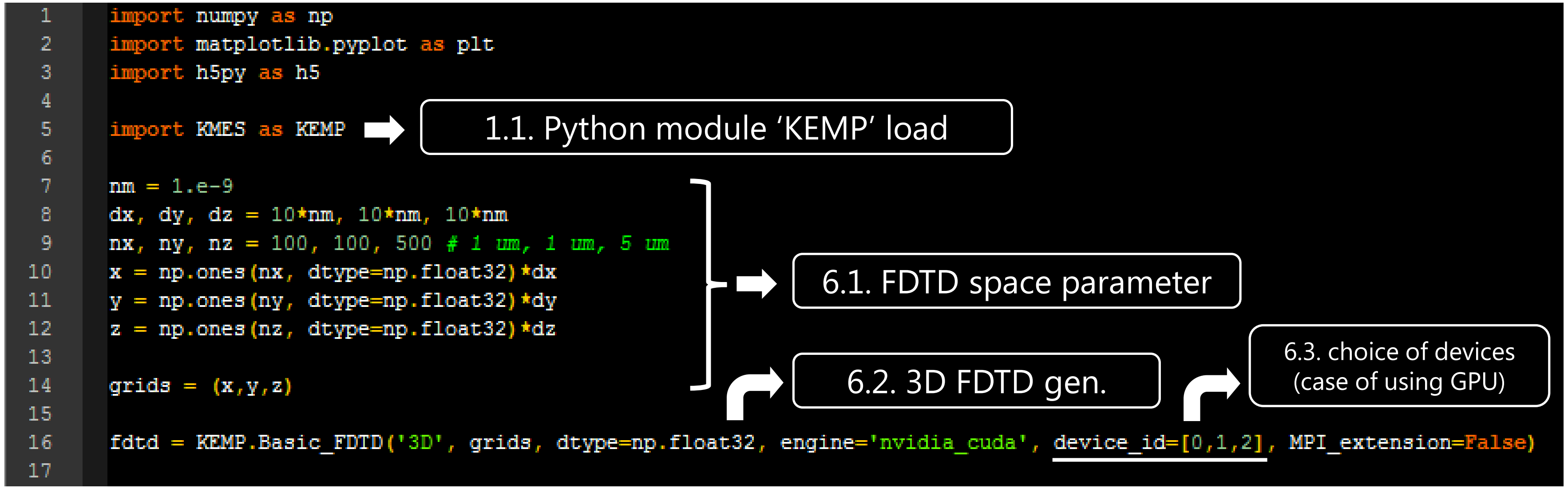
(2) EM wave propagation (fdtd.updateE, fdtd.updateH)

(3) Get time-line field data

5.2. Export FDTD field data to hdf5 files

Electromagnetic fields(fdtd.{ex, ey, ez, hx, hy, hz} operate like python numpy arrays. Therefore we can get field data of certain region. The data can be exported to Hierarchical Data Format ver.5 (hdf5) files.

User interface
6. Execution of python script (Desktop PC)



```
import numpy as np
import matplotlib.pyplot as plt
import h5py as h5

import KMES as KEMP

nm = 1.e-9
dx, dy, dz = 10*nm, 10*nm, 10*nm
nx, ny, nz = 100, 100, 500 # 1 um, 1 um, 5 um
x = np.ones(nx, dtype=np.float32)*dx
y = np.ones(ny, dtype=np.float32)*dy
z = np.ones(nz, dtype=np.float32)*dz

grids = (x,y,z)

fdtd = KEMP.Basic_FDTD('3D', grids, dtype=np.float32, engine='nvidia_cuda', device_id=[0,1,2], MPI_extension=False)
```

1.1. Python module 'KEMP' load

6.1. FDTD space parameter

6.2. 3D FDTD gen.

6.3. choice of devices
(case of using GPU)

6.1. choice of computing device

We can choose computing devices by setting the engine parameter and device id. device_id is ignored in the case of using CPUs.

Because multiple GPUs can be arranged in one workstation, KEMP supports multiple GPUs by setting device_id parameter. If multiple devices are chosen, the computation performance is greatly enhanced.

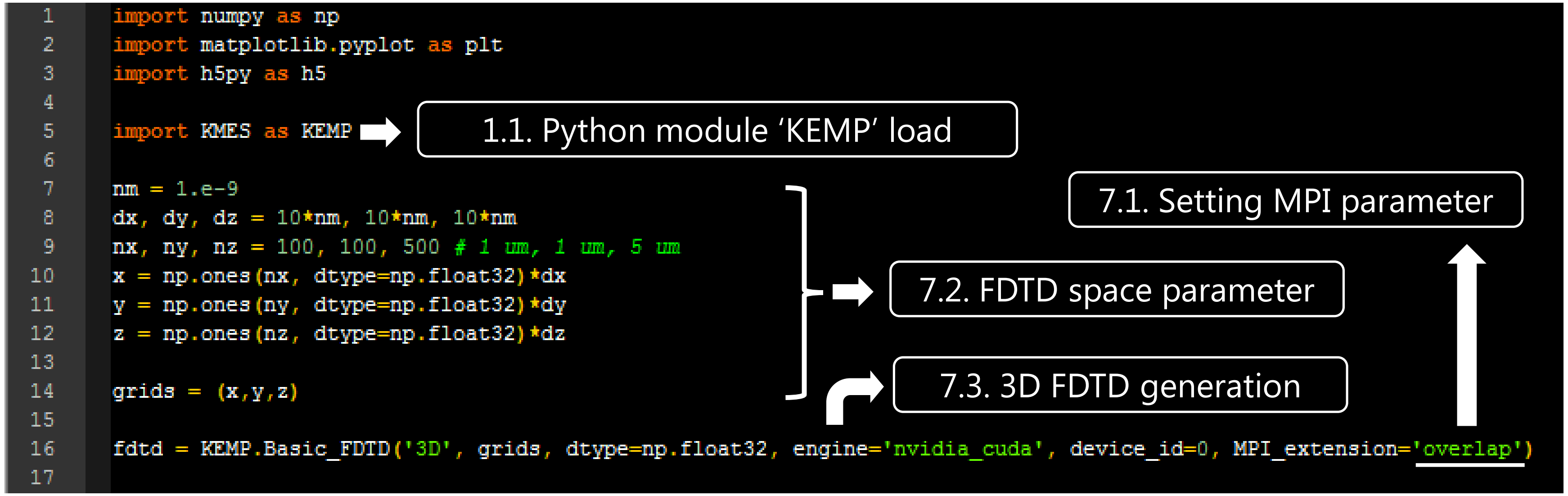
KEMP recommend to use same devices because of the devices assigned 3D-FDTD space into equal parts.

6.2. Execution of the script

We can execute the FDTD calculation by execution of python script.

$ python KEMP_example.py

User interface
7. Execution of script applying MPI interface (cluster computers)



```python
import numpy as np
import matplotlib.pyplot as plt
import h5py as h5

import KMES as KEMP

nm = 1.e-9
dx, dy, dz = 10*nm, 10*nm, 10*nm
nx, ny, nz = 100, 100, 500 # 1 um, 1 um, 5 um
x = np.ones(nx, dtype=np.float32)*dx
y = np.ones(ny, dtype=np.float32)*dy
z = np.ones(nz, dtype=np.float32)*dz

grids = (x,y,z)

fdtd = KEMP.Basic_FDTD('3D', grids, dtype=np.float32, engine='nvidia_cuda', device_id=0, MPI_extension='overlap')
```

1.1. Python module ‘KEMP’ load

7.1. Setting MPI parameter

7.2. FDTD space parameter

7.3. 3D FDTD generation

7.1. Setting MPI parameter

We can set MPI_extension parameter of 3D-FDTD space to following options. (default value is False)

1. ‘block’ : blocking communication mode is on.
2. ‘nonblock’: nonblocking communication mode is on.
   Computation speed is enhanced compared with ‘block’ option.
3. ‘overlap’: overlapping computation mode is on. (technically uses nonblocking communication mode)
   communication delay is hided in calculation time of main-region.

7.2. Execution the script

Execution by using MPI on Linux shell environment.

$ mpirun –np 3 –host y201 y202 y203 python KEMP_example.py

Information of KEMP & Download

Main page of KEMP

http://nol.korea.ac.kr/kemp.html

Tutorial & User guide

http://nol.korea.ac.kr/kempguide.html

Download

http://sourceforge.net/projects/kemp/?source=directory (Sourceforge link)