Install & Run:

操作系统：deepin2015

Github 地址：https://github.com/wphu/Smilei_SEF

要求：c++编译器，mpi 库，hdf5 库，python2.7, SuperLU4.3

需要注意：编译 hdf5 需要 mpicc 编译器编译并行版本，configure 如下：

CC=mpicc ./configure –enable-parallel –prefix=/home/huwanpeng/opt/hdf5-intel-mpi，但有时候需要-fPIC： CC=mpicc ./configure -enable-parallel CFLAGS=-fPIC -prefix=/share/apps/hdf5-intel-mpi （相应的 mpi 也需要加上-fPIC 重新编译）；

另外 mpi 库和 hdf5 用到的编译必须是相同的，gnu 或者 intel 都行。

环境变量设置（相应环境变量设置成当前系统的）：

```
#mpi 路径
PATH=/home/wp/opt/mpich-3.2/bin:$PATH

#makefile 用 SMILEICXX 指定的 mpi 编译器来编译，如果没有设置就用 makefile 中设置的值
export SMILEICXX=mpicxx

#mpi 版本 hdf5 库的根目录
export HDF5_ROOT_DIR=/opt/hdf5-mpich3

#这个时动态链接库，编译器时不需要，但是 code 运行时需要
export
LD_LIBRARY_PATH=/home/wp/opt/mpich-3.2/lib:/opt/hdf5-mpich3/lib:$LD_LIBRARY_PATH

#二维解泊松方程的时候需要用到 superlu 库
export SuperLU_DIR=/home/wp/codes/science/SuperLU_4.3
```

Code 运行及查看结果(smilei_SEF-v2.0/example/tst1d_simple 目录下)：

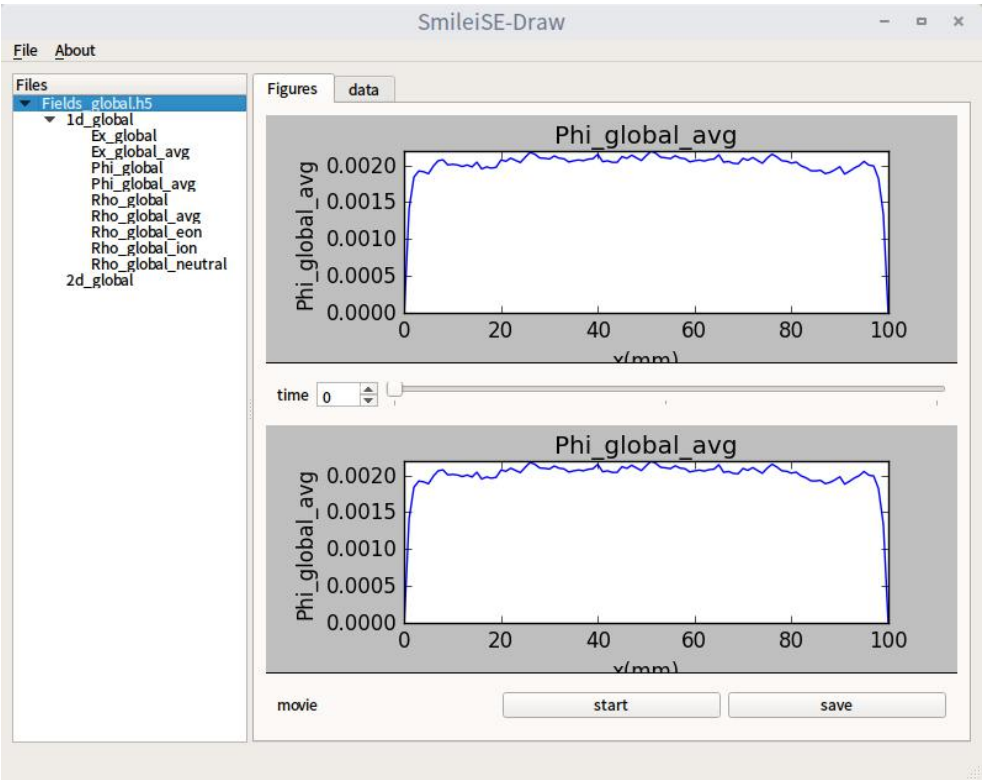
运行：./run.sh

查看结果：./draw_gui.sh

#draw_gui.sh 调用 example/post_process 目录下的 python 脚本，并读入 example/tst1d_simple 目录下的.h5 文件，然后进行绘图。

Post-process

The code output hdf5 data file, file name is like “Field_global.h5”. The python scripts like “draw_gui.py” in example/post_process, can be used to draw the picture, however, the scripts are simple, and specified for this code. The main python libraries include: PyQt5, h5py, matplotlib. Result is shown in below figure:



Problem: when redraw animation, python give below error, but the program still can ran.

Exception TypeError: TypeError("'instancemethod' object is not connected",) in <bound method TimerQT.__del__ of <matplotlib.backends.backend_qt5.TimerQT object at 0x7fcb2c13da90>> ignored

#> Other software for reading and drawing h5 file see 附录 2

Centos install PyQt5: http://www.jianshu.com/p/abf910e78771

Change plan:

1、暂时废弃 Diagnostic MovWindow 和 SmileiIO 三个模块；

2、差值（计算局部电场）和投射（计算电荷密度）改为 1 阶；

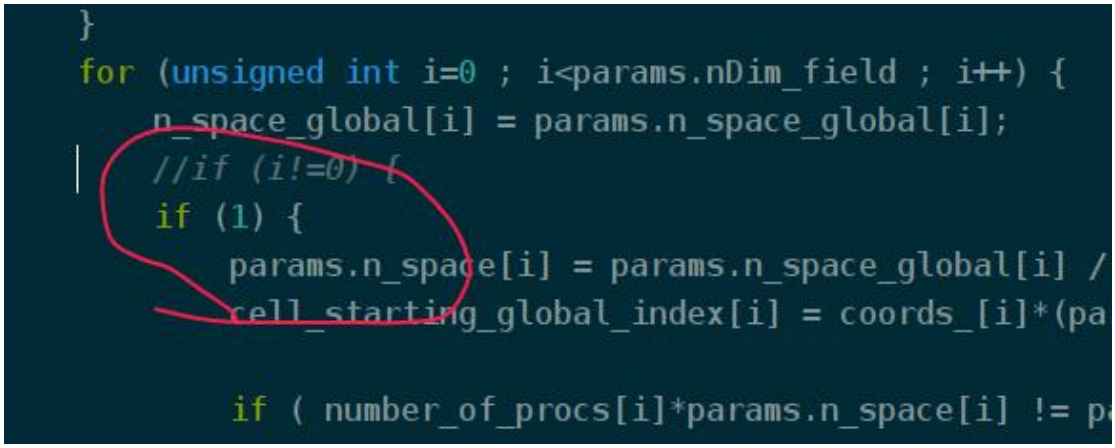
3、求解电场：首先将电荷密度收集到 Master 节点，然后 SuperLU 求解电势和电场，然后将电场分发到各个节点；

4、考虑怎样注入粒子。

Known bugs:

1、二维并行 mpi 区域划分只能是 [1 x n]， 而不能是 [m x n]，即 m 不能是 1；

解决：SmileiMPI_Cart2D.cpp 的 createTopology 函数里如下改动：

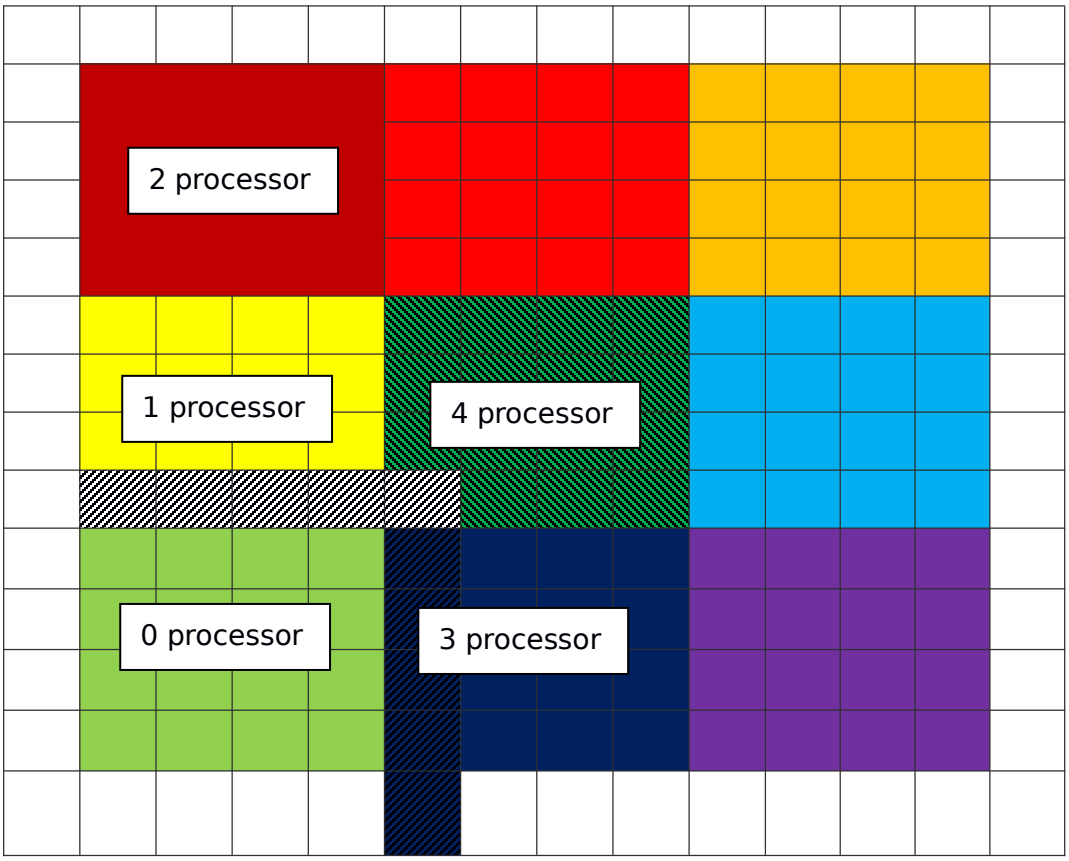


1、跨节点并行时，一些粒子的 weight 会变成 nan，目前发现应该时 exchangeParticles 导致的；

解决：SmileiMPI_Cart2D.cpp 的 exchangeParticles 函数里将 mpi 接收的非阻塞性改成阻塞性，即 MPI_Irecv 改成 MPI_Recv，并且相应的参数做修改，rrequest 改成 rstat。且 MPI_Recv 后面要加 barrier()。现在应该没啥问题，但还不确定会不会出别的问题或需进一步修改。

2、

Parallelization



1. SmileMPI_Cart2D.h

dims_global_gather[2]: intermediate variable,
dims_global_gather[0]*dims_global_gather[1] is the total number of grid points for gathering or scattering charge density and electric field.

dims_gather[smilei_sz, 2]: number of grid points in 2 direction of all processors(smileri_sz).

2. PicParams.h

n_space[2]: number of grids (not grid points) in the current processor.

n_space_global[2]: number of grids in the global region.

3. Python/pyinit.py

If you want to add one class like species and collisions in the input file(.py) (for example, now I want to add PSI module), you should add the responding class in the pyinit.py file to let the code recgonizes the content in the input file. Like below:

```
class PSI(SmileiComponent):
    """PSI parameters"""
    species1 = None
    species2 = None
    PSI_type = None
```

About weight of macro particles

Weight in the code: Wc = n / Npic, where n is number density, and Npic is macro particle number in one cell.

Weight in reality: Wr = n * dx * dy *dz / Npic, where dx dy dz is cell lenght. For 1D dy = dz = 1 (m), for 2D, dz = 1 (m).

So we can deduce that: Wr = Wc * dx * dy * dz, which will be used in calculating the paritcle flux and heat flux.

Calculate particle flux: pFlux = nP * Wr / (dS * dt), nP is particle number absorbed by the wall in the time duration dt, and dS is the wall surface area.

We can get: pFlux = nP * Wc * dx * dy *dz / (dy * dz *dt) = nP * Wc * dx / dt.

We can also get the heat flux: hFlux = nE * Wc * dx / dt, where eE is accumulated energy.

DSMC

1. Collision sampling techniques

1.1 The NTC method is species independent, and can be applied to all the molecules as a single group. However, a practical problem arises when there is a very large variation in the molecular masses. This is because the collisions that involve the light molecules will have high values of the relative speed and, since these will set the value of $(\sigma c)_{max}$, the acceptance rate for the heavy molecules will be low, and the overall selection process will be inefficient. Which means: this is OK for light molecules, but is bad for heavy molecules.

Solution: the method can be applied separately to the species and, for a collision between a species p and a species q molecule, $(\sigma c)_{max}$ becomes $\{(\sigma c)_{max}\}_{pq}$ (equation 11.6 in Bird' book). The efficiency improvement is best for heavy-heavy, then heavy-light, worse for light-light.

DSMC is more like MC, not Binary collision for coulomb.

1.2 In the equation 11.3, number of pair selections per time step is obtained by multiplying by $N^2/2$. However, in most cases, N is a fluctuating quantity and, so N2 should be replaced by the product of the instantaneous value and a time or ensemble averaged value. In the code, the average N is averaged all the time from beginning to now.

附录 1：原 SMILEI code User Guide:

在 centos6.3 和 deepin2015a1 中都成功编译并运行。

官网：http://www.maisondelasimulation.fr/projects/Smilei/html/installation.html

要求：c++编译器，mpi 库，hdf5 库，python2.7

需要注意：编译 hdf5 需要 mpicc 编译器编译并行版本，configure 如下：CC=mpicc ./configure –enable-parallel –prefix=/home/huwanpeng/opt/hdf5-intel-mpi；另外 mpi 库和 hdf5 用到的编译必须是相同的，gnu 或者 intel 都行。

环境变量设置：

#mpi 路径
PATH=/home/wp/opt/mpich-3.2/bin:$PATH

#makefile 用 SMILEICXX 指定的 mpi 编译器来编译，如果没有设置就用 makefile 中设置的值
export SMILEICXX=mpicxx

#mpi 版本 hdf5 库的根目录
export HDF5_ROOT_DIR=/opt/hdf5-mpich3

#
export LD_LIBRARY_PATH=/home/wp/opt/mpich-3.2/lib:/opt/hdf5-mpich3/lib:$LD_LIBRARY_PATH
export SuperLU_DIR=/home/wp/codes/science/SuperLU_4.3

运行例子命令(smilei-v2.0 目录下)：
mkdir tst1d_6_particle_diagnostic
cp benchmarks/tst1d_6_particle_diagnostic.py tst1d_6_particle_diagnostic/
mpiexec -np 2 src/smilei tst1d_6_particle_diagnostic/tst1d_6_particle_diagnostic.py

有些例子会出现下面错误：

原因是：dim=2d3v 应该写成 dim=’2d3v’

==========

函数及参数说明

==========

1 virtual double computeNRJ(unsigned int shift, unsigned int istart[3][2], unsigned int bufsize[3][2]) = 0;

//Method used to compute the field energy (assuming electromagnetic type)

//纯虚函数，

2 void Field2D::shift_x(unsigned int delta)

//Method to shift field in space

//暂时不知道具体是干啥的

附录 2：查看 hdf5 文件或绘图可能用到的软件:

The code output hdf5 data file, file name is like “Field_global.h5”. The python scripts like “draw1d_gui.py” , “draw1d.py ” can be used to draw the picture, however, the scripts are simple, and specified for this code. The are some software and libraries can view hdf5 files and draw figures, such as below:

1. Hdfview (java);
2. H5pyViewer; https://pypi.python.org/pypi/h5pyViewer
3. ViTables; http://vitables.org/Download/
4. Panoply;

http://stackoverflow.com/questions/8897195/hdf5-viewers-editors-linux

5. NeXus/HDF5 Tree View http://www.opengda.org/documentation/manuals/Diamond_SciSoft_Python_Guide/8.18/nexusviewer.html

附录 3：程序调试工具的使用：

http://blog.csdn.net/yanghao23/article/details/7514587
http://www.cnblogs.com/wangkangluo1/archive/2011/07/20/2111248.html

1. 在编译和链接时，CFLAGS += -g -pg -Wall -D__DEBUG -O0，优化要用-O0，不然错误位置和 code 对应不上；在正式运行时，CFLAGS += -O3，-O3 对程序进行优化，运行速度会提高很多。

2. 如果是 mpi 并行程序，运行命令是：mpiexec -n 10 valgrind　./yourProgram

3. 如果出错太多，超过 1000 个错误后，之后的错误不再显示，如果需要显示全，需要加入--error-limit=no 在 valgrind 后面。