Smilei

training workshop
March 2019
Maison de la Simulation

Interface, outputs, post-processing

Frédéric Pérez

The minimum knowledge to use Smilei

1. Compile
2. Write an input file (a.k.a. *namelist*)
3. Run the program
4. Read & post-process the outputs

Smilei already does much of the work for you

The minimum knowledge to use Smilei

1. Compile
2. Write an input file (a.k.a. *namelist*)
3. Run the program
4. Read & post-process the outputs

Requirements

- C++11 compiler (with openMP)
- MPI (with MPI_THREAD_MULTIPLE)
- HDF5
- Python 2.7+

Help is given on our website

Compiling Smilei

```
$ make
```

```
$ make -j 8      # compile with 8 processors
```

```
$ make config=debug
```

```
$ make machine=irene_skl
```

Send us your machine's configuration !

The minimum knowledge to use Smilei

1. Compile
2. Write an input file (a.k.a. *namelist*)
3. Run the program
4. Read & post-process the outputs

The input file is written in python

```
Main(
    geometry = "1Dcartesian",
    timestep = 0.009,
    simulation_time = 1.,
    cell_length = [0.01],
    grid_length  = [4.],
...
)

Species(
    name = 'ion',
    particles_per_cell = 100,
    mass = 2000,
    charge = 1.0,
    number_density = trapezoidal(1., xplateau=2.),
...
)

DiagFields(
    every = 100,
)
```

Normalized units
(c, m_e, $m_e c^2$, etc.)

The python input provides flexibility

- Calculate simulation parameters at runtime

```
omega0_SI   = 2. * math.pi * 3e8 / 1.06e-6
duration_SI = 300.e-15
Main(   simulation_time = duration_SI * omega0_SI,   ... )
```

- Loops

```
for s in [“ion1”, “ion2”, “ion3”, “ion4”]:
    DiagParticleBinning(   species = [s],   ... )
```

- Plasma & laser profiles are given as *functions*

```
def myprofile(x,y):
    return x * exp(-(y-y0)**2)
Species(   number_density = myprofile,   ... )
```

- Any python code accepted

  Import modules, read external files, run other scripts, etc.

The minimum knowledge to use Smilei

1. Compile
2. Write an input file (a.k.a. *namelist*)
3. Run the program
4. Read & post-process the outputs

Test and run

```
$ smilei_test myinput.py
```

(fast & convenient way to debug the input)

```
$ mpirun -n 4 smilei myinput.py
```

The minimum knowledge to use Smilei

1. Compile
2. Write an input file (a.k.a. *namelist*)
3. Run the program
4. Read & post-process the outputs

Outputs

- Standard output: *lots of info, warnings, errors, etc.*
  (the “log”)

- Diagnostics
  - `Scalar` global simulation quantities
  - `Fields` direct output of the fields arrays
  - `Probe` fields interpolated on regular grids
  - `ParticleBinning` versatile averaged particle data
  - `Screen` time-integrated particles passing through surface
  - `TrackParticles` particle trajectories
  - `Performances` timers, memory, load balancing

- Checkpoints (≡ dumps and restarts)

  HDF5 files in the `checkpoints` folder for restarting the simulation

Included python post-processing

- The repository includes a *python* module `happi`

```
$ make happi
$ ipython
In [1]: import happi
```

- Get simulation parameters
- Data manipulation
- Obtain raw or processed data
- Plots in 1D, 2D (animated)
- Convert to VTK format (for VisIt or Paraview) → 3D

Limitations: no parallel processing; limited in memory

The minimum knowledge to use Smilei

1. Compile
2. Write an input file (a.k.a. *namelist*)
3. Run the program
4. Read & post-process the outputs

Questions ?