PyHEADTAIL on the GPU

... an Overview!

Adrian Oeftiger for the PyHEADTAIL dev team

CERN – PyHEADTAIL Meeting #26

26 March 2019

Running PyHEADTAIL on the GPU?

... nothing easier than running `PyHEADTAIL` on a hardware accelerating GPU (graphics processing unit):

1. start from your existing simulation script (`main.py`)
2. add 3 lines of code
3. run script on machine with NVIDIA GPU (e.g. CNAF cluster in Bologna)

Running PyHEADTAIL on the GPU?

… nothing easier than running `PyHEADTAIL` on a hardware accelerating GPU (graphics processing unit):

1. start from your existing simulation script (`main.py`)
2. add 3 lines of code
3. run script on machine with NVIDIA GPU (e.g. CNAF cluster in Bologna)

Why on the GPU?

- **speed-up**: 1 GPU vs. single CPU core
  - factor 100 faster smooth approximation space charge simulations
  - factor 10 faster instability simulations
- **simplicity**: easy to transfer script to GPU

⟶ strive for **maintainability**: 1 place of physics implementation!

Reminder for Users

usual script code:

```
bunch = (...)
one_turn_map = (...)


for turn in xrange(n_turns):
    for m in one_turn_map:
        m.track(bunch)
```

Reminder for Users

extended script code:

```
import pycuda.autoinit
from PyHEADTAIL.general.contextmanager import GPU

bunch = (...)
one_turn_map = (...)

with GPU(bunch) as context_manager:
    for turn in xrange(n_turns):
        for m in one_turn_map:
            m.track(bunch)
```

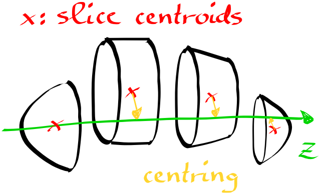
⟶ wrap "`with GPU(bunch) as cmg:`" around simulation code

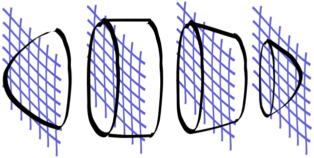
⟹ PyHEADTAIL takes care of managing CPU RAM and GPU RAM

What simulation types can I run on the GPU?

- wakefield
- space charge:
    - **frozen 3D fieldmap** with interpolation (e.g. from beam blueprint, can be adaptive/auto-updating)
    - **frozen transverse fieldmap** with longitudinal self-consistent line charge density
    - **exact Bassetti-Erskine** with slice-based actual rms sizes
    - **self-consistent 3D PIC**
    - **self-consistent 2.5D PIC**, recentring of transverse grids along slices

      $\Longrightarrow$ also implemented moving aperture fixed to recentring grids!





Overview: GPU-compatible Modules

PyHEADTAIL / PyHEADTAIL /

Create new file | Upload files | Find file | History

aoeftiger release-script: bumping version file. — Latest commit `3ac0e26` 12 days ago

..		
aperture	Move package to subdirectory	2 years ago
cobra_functions	monitors: CellMonitor uses coordinates instead of bunch object	a year ago
feedback	transverse_damper: generalising resistive damper..	4 months ago
field_maps	field_map: fixed typo in FieldMapSliceWise (zp -> dp)	2 years ago
general	contextmanager: need to define error message above import of pmath	2 years ago
gpu	All pycuda.array.empty_like calls fixed in gpu_wrap	4 months ago
impedances	wake_kicks.py: typo in comment	a month ago
machines	synchrotron: allow higher-order chroma	2 years ago
monitors	monitors: CellMonitor uses coordinates instead of bunch object	a year ago
multipoles	Move package to subdirectory	2 years ago
particles	generators: some changes in internal variable naming	2 months ago
radiation	Move package to subdirectory	2 years ago
rfq	Move package to subdirectory	2 years ago
spacecharge	adaptive pic: take sigmas from beam	2 years ago
testing	unittests: removing parallelplates* deprecation warnings	4 months ago
trackers	long_tracking: added RFBox for square well potential	a month ago
__init__.py	__init__.py: versioning corrected	2 years ago
_version.py	release-script: bumping version file.	12 days ago

Overview: GPU-compatible Modules

PyHEADTAIL / PyHEADTAIL /

Create new file | Upload files | Find file | History

aoeftiger release-script: bumping version file. Latest commit 3ac0e26 12 days ago

PyHEADTAIL

⚠: (partly) only on GPU ✓: on GPU ✗: not on GPU

Overview of modules:

- ✓ aperture
- ✓ cobra_functions
- ✓ feedback
- ⚠ field_maps
- ✓ general
- ✓ gpu
- ✓ impedances
- ✓ machines
- ✓ monitors
- ✓ multipoles
- ✓ particles
- ✗ radiation
- ✓ rfq
- ⚠ spacecharge
- ⚠ testing
- ✓ trackers

_version.py release-script: bumping version file. 12 days ago

PyPIClib

The particle-in-cell part of the PyHEADTAIL space charge module depends on PyPIClib[1]:

- mainly implemented on GPU:
  - free space (open boundary) FFT with integrated Green's function
    - ⟶ 2D, 3D
    - ⟶ 2.5D: parallel transverse 2D grids along longitudinal plane (slices)
  - linear algebra sparse matrix solver with Dirichlet boundary condition
- `PyHEADTAIL.spacecharge.pypic_spacecharge.SpaceChargePIC` calls:
  - ⟶ `pypic.poissonsolver.is_25D` (if `True`, also `pypic.mesh.dz`)
  - ⟶ `pypic.mesh.dimension` (assert 3D)
  - ⟶ `pypic.pic_solve`

[1] integrated under GPU directory in PyPIC, https://github.com/PyCOMPLETE/PyPIC/ stand-alone: https://github.com/aoeftiger/PyPIClib/

Context Management

Abstracted `PyHEADTAIL` into 1 jupyter notebook:
View notebook ↗ in my github repository ↗

⟶ ready to play and improve the concept in realistic conditions

⟹ already working with NVIDIA engineers via openlab E4 project ↗ to

1. improve concept for better throughput for instability simulations (e.g. transverse trackers)
2. incorporate advanced non-linear tracking with `SixTrackLib`

Context Management

Abstracted `PyHEADTAIL` into 1 jupyter notebook:
View notebook ↗ in my github repository ↗

⟶ ready to play and improve the concept in realistic conditions

⟹ already working with NVIDIA engineers via openlab E4 project ↗ to

1. improve concept for better throughput for instability simulations (e.g. transverse trackers)
   ⟶ during April'19
2. incorporate advanced non-linear tracking with `SixTrackLib`
   ⟶ until autumn '19

Resources

Some links:

- Adrian Oeftiger talk '17: PyHEADTAIL v1.11 GPU integration ↗
- Stefan Hegglin talk '16: PyHEADTAIL GPU concepts ↗
- Simulating Collective Effects on GPUs ↗ [1]
  ⟶ foundation of GPU computing in `PyHEADTAIL`, context management
- Space Charge Modules for PyHEADTAIL ↗ [2]
  ⟶ derivation of `PyHEADTAIL`'s space charge suite, models, physics, limitations

Distributed GPU Computing

Collective beam dynamics: *memory constrained*, cross-talking threads

- memory-intensive: FCC-hh example for coupled-bunch instability simulation 10 400 bunches × 500 000 macro-p./bunch × 8 coord./macro-p. × 8 B/coord. ≈ 300 GB
- latest NVIDIA V100 cards come in 16 GB and 32 GB flavours
- have 4× V100 (16 GB) in Bologna for BE-ABP
- CERN IT bought 16× V100: available in near future in HPC cluster
- CSCS (Swiss National Supercomputing Centre) has 5704 nodes with NVIDIA P100 (16 GB) each!

MPI + GPU possible: NVIDIA instructions ↗

- with CUDA-aware MPI (openMPI v≥ 1.7): avoid transfer through CPU memory, direct GPU communication

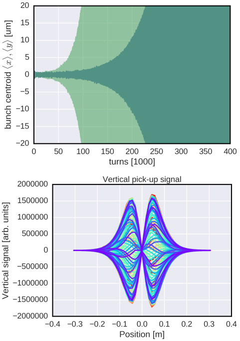
⟶ good to have NVLink connected GPU cards (up to 25 GB/sec bi-directional)

⟹ How to use MPI + GPU in `PyCUDA` ↗

PySixTrack / SixTrackLib Integration?

Space charge study in HL-LHC presented in WP2 meeting 31.10.2017 ↗

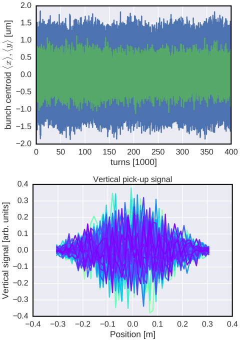
only wake field



vert. head-tail instab.,
18.6 kturns rise time

+

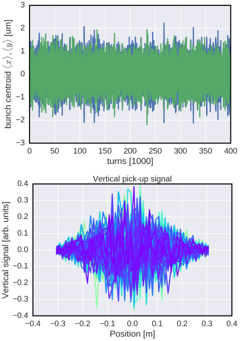
only space charge



stable

=

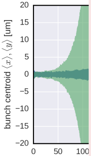
wake field + SC

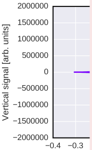


stable!

Space charge study in HL-LHC presented in WP2 meeting 31.10.2017 ↗

only





vert. hea
18.6 ktu

PIC long-term study

400 000 turns with 100 kicks per turn (smooth approximation) of particle-in-cell space charge on K40 GPU for 1×10^6 macro-particles:

2.5 sec/turn

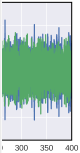
(also easily able to run the convergence cross-check with 15×10^6 macro-particles)

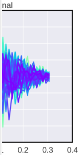
SixTrackLib tracks LHC lattice for 1×10^6 macro-particles on V100 GPU in

1 sec/turn

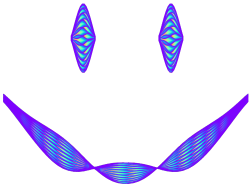
⟹ long-term self-consistent PIC space charge studies with non-linear tracking **feasible**!

SC





Thank you for your attention!



References I

[1] Stefan Eduard Hegglin, Kevin Li, and Peter Arbenz. "Simulating Collective Effects on GPUs". Presented 10 Feb 2016. Feb. 2016. URL: http://cds.cern.ch/record/2239398.

[2] Adrian Oeftiger and Stefan Eduard Hegglin. "Space charge modules for PyHEADTAIL". In: CERN-ACC-2016-0342 (July 2016), MOPR025. 6 p. URL: http://cds.cern.ch/record/2239290.