GPU Parallelisation of Particle-In-Cell Algorithm

Beam Physics Simulations

Giovanni Iadarola, Giovanni Rumolo, and

Adrian Oeftiger (Ph.D. student in

BE-ABP-HSC section / Space Charge Working Group)

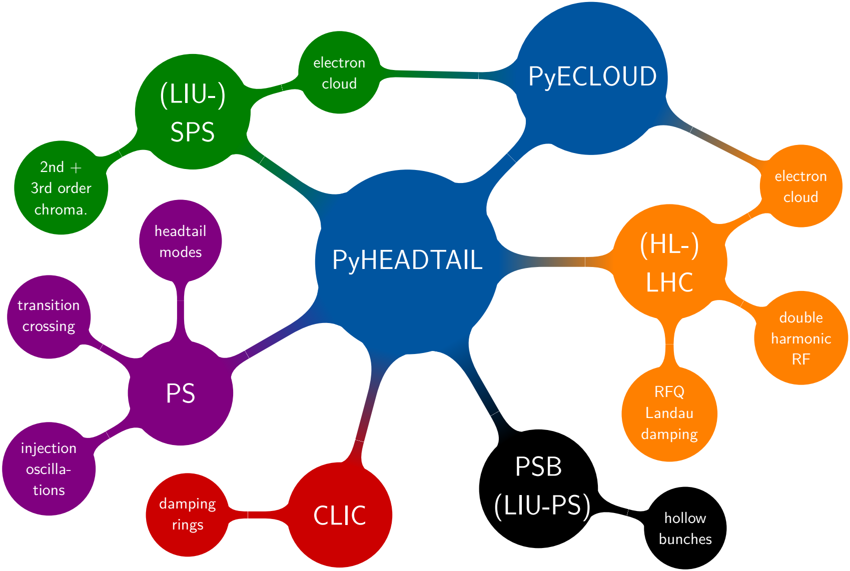
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

CERN

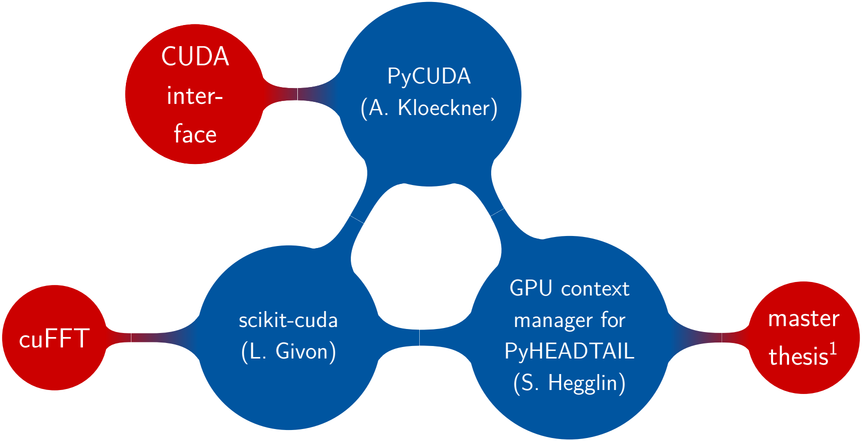
GPU Computing Meeting #2, CERN

11. March 2016

PyHEADTAIL / PyECLOUD Studies (Reminder)



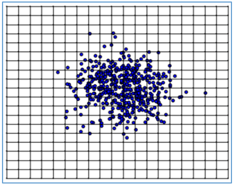
PyHEADTAIL GPU Ingredients (Reminder)

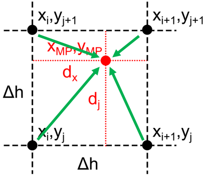


1) http://indico.cern.ch/event/471081/

Particle-in-cell Algorithm (Reminder)

1. **particles to mesh**: deposit all macro-particle charges onto (regularly distributed) mesh nodes
2. **solve** discretised Poisson equation on the mesh, options:
   - Hockney's algorithm $\Longrightarrow$ 'cheap' FFT algorithm
   - direct solving, e.g. via sparse matrices
   - iterative solving (Jacobi, SOR, Conjugate Gradient, ...)
3. **gradient** of potential yields electric fields
4. **mesh to particles**: interpolate mesh fields to particles





Hockney's Algorithm (Reminder)

Poisson's equation (*without* boundary conditions)

$$\Delta\phi(\vec{x}) = \rho(\vec{x})$$

can be solved via the Green's function method

$$G: \ \Delta G(\vec{x}) = \delta(\vec{x}) \quad .$$

Trick: mirroring $G(\vec{x})$ for each plane $\implies$ periodicity!
Formal solution with convoluted Green's function

$$\varphi(\vec{x}) = \int d^3y \ \rho(\vec{x})\, G(\vec{x},\vec{y})$$

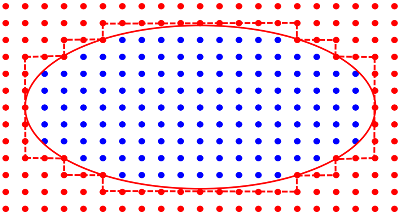
can be expressed as Fourier transform ($\implies$ **FFT**!),

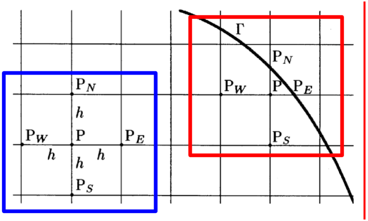
$$\varphi(\vec{x}) = \mathscr{F}\{\mathscr{F}\{\rho\}\mathscr{F}\{G\}\} \quad .$$

What If... Interior Boundary Conditions?

Shortley-Weller algorithm:

- next-order boundary interpolation w.r.t. step function (extrapolating to outer grid point)
- vanishing potential at *grid edge* intersection with boundary





⟹ implemented in PyECLOUD (on the CPU!)

⟹ reduces electric field artefacts close to vacuum chamber, important for electron clouds building up from wall impact

Peter Messmer: Some Suggested Improvements I

particle-to-mesh deposition

- we implemented sorted charge deposition, 3x speed-up over double precision atomics

$\rightsquigarrow$ single precision atomics have smaller error significance than introduced by mesh discretisation

$\Longrightarrow$ (hardware supported) single precision atomics faster (!)

particle-to-mesh and mesh-to-particle interpolation

- we implemented custom kernels for cloud-in-cell interpolation

$\rightsquigarrow$ textures offer interpolation (single-precision again!)

$\Longrightarrow$ (read-only) texture memory faster

Peter Messmer: Some Suggested Improvements II

Poisson solver for non-trivial interior boundary conditions

- we implemented direct solving algorithm using cuSOLVER

$\rightsquigarrow$ Poisson matrix direct inversion not well-suited for parallelisation

$\Longrightarrow$ try iterative solver e.g. with Conjugate Gradient algorithm via cuSPARSE

Iterative Solver: Physics Scenario

$\rightsquigarrow$ 12. January: presentation on direct space charge (open boundary conditions! allows Hockney algorithm / FFT!)

electron cloud in LHC:

- quasi-2D problem with interior boundary condition (perfectly conducting vacuum chamber walls)
- need to transversely resolve beam *and* whole chamber

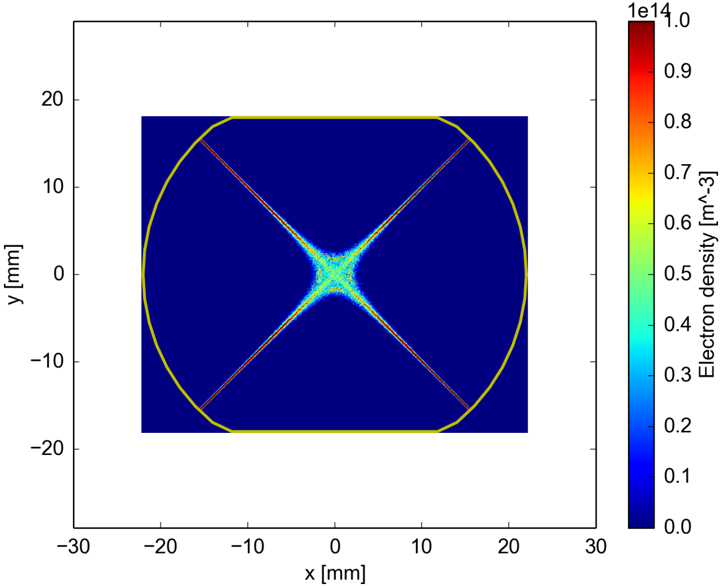
$\longrightarrow$ usual simulations at LHC injection (450 GeV): large beam

$\longrightarrow$ LHC top energy (6.5 GeV): beam size shrinks by factor 4

- many successive sub time steps to integrate motion of electrons between interaction with beam slices
- $\mathcal{O}(100)$ turns for instability analysis

$\Longrightarrow$ GPU: top energy study feasible with fast iterative solver?

Electron Cloud in LHC



Thank you for your attention!

Acknowledgements to PyHEADTAIL / PyPIC / PyECLOUD teams:

Hannes Bartosik, Stefan Hegglin, Giovanni Iadarola, Kevin Li,
Lotta Mether, Annalisa Romano, Giovanni Rumolo, Michael Schenk

`https://github.com/PyCOMPLETE/`