GPU-Accelerated 3-D Electromagnetic Particle-in-Cell Implementations in VORPAL

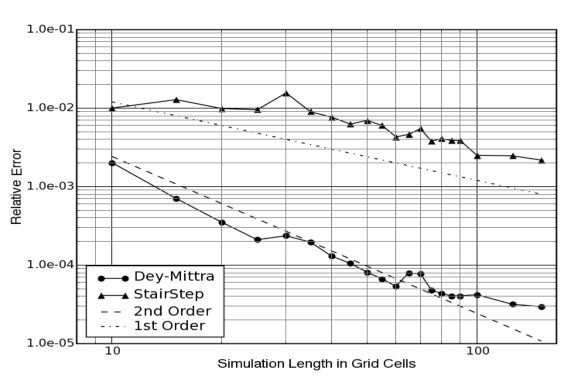
K. Amyx (Tech-X Corporation)

Abstract

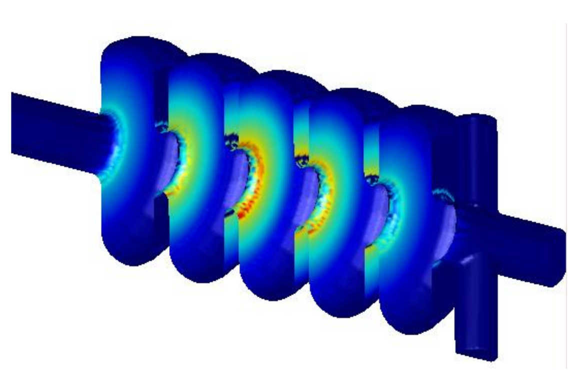
We present recent developments in implementing 3D GPU-accelerated eletromagnetic particle-in-cell particle updates in the plasma physics framework VORPAL. The primary challenge in PIC methods on GPUs is thread contention during the current deposition stage: we resolve these thread contentions by sorting particles into 'tiles' of many cells each time step. Multiple thread blocks may be assigned to each tile, and each block accumulates the contribution to the deposition field from a moderate number of particles via an optimized unsegmented Esirkepov 1st-order scheme. These buffers are then written back to global field mesh via atomic operations. We have observed performance increases of 20-25x over CPU-based VORPAL implementations for fully self-consistent double-precision electromagnetic PIC simulations using Tesla C2070 GPUs, corresponding to update times of 25 ns per particle (for electrostatic simulations) and 50 ns per particle (for electromagnetic simulations). We have seen little degradation in performance between hot and cold plasmas, or between uniform plasmas and dense plumes.

VORPAL

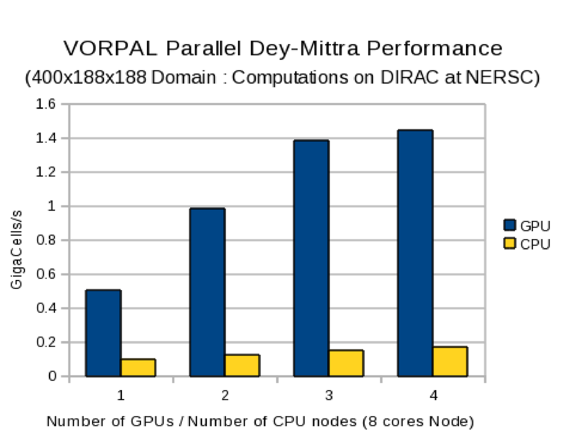
VORPAL is a massively-parallel, highly-flexible plasma and EM modeling framework (http://vorpal.txcorp.com). VORPAL currently supports GPU-acceleration of Finite Difference Time Domain (FDTD) methods, including Dey-Mittra algorithms for 2nd-order accuracy for complex cut-cell geometries. FDTD is a highly-scalable, explicit algorithm for modeling time-dependent EM problems. The Dey-Mittra algorithm is an extension of FDTD to enable 2nd order accuracy for complex cut-cell geometries. This enables highly accurate, yet time efficient simulations of devices like RF cavities (IEEE Microwave and Guided Wave Letters, 7 (9), 1997).



2nd order accuracy for Dey-Mittra algorithm in case of Spherical resonator



π-Mode of a Project X Cavity shown through |E| on cavity walls. Simulation was done on 2 NVIDIA FERMI 2070 GPUs.

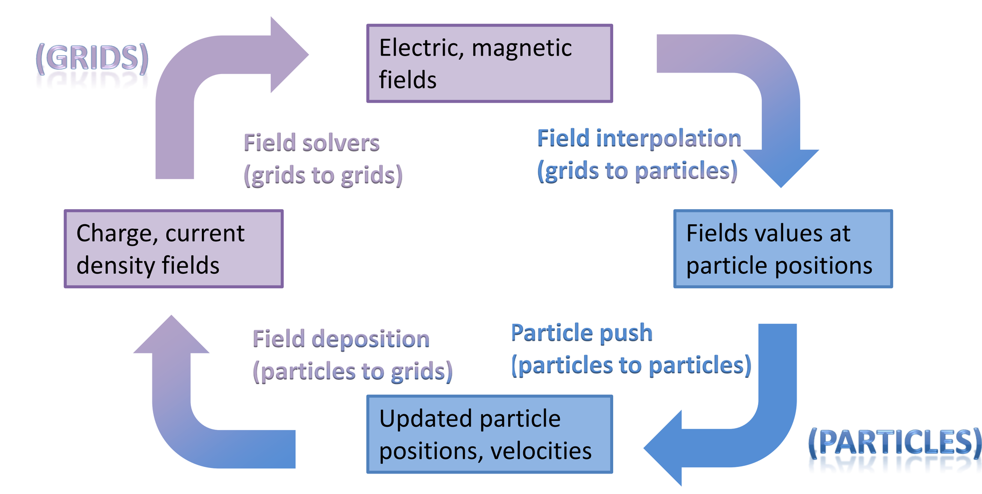


GPU-Accelerated FDTD Simulations in VORPAL show excellent scaling across multiple GPUs using a hybrid MPI-CUDA scheme.

VORPAL also supports Just-in-Time compilation of user-defined initial and boundary conditions, using the CUDA driver API to dynamically load generated kernels at runtime.

Particle-in-Cell Methods

In a PIC approach, the O(N2) problem that would lead to a full solution of Maxwell's equations is transformed into an O(N) problem by introducing discretized field meshes.



- Primary challenge on a GPU: thread contention in the deposition step
  - A 1st-order particle writes to between 12 and 48 field values in an EM simulation
  - In principle, all particles can deposit to the *same* field values

Esirkepov Current Deposition

For a particle with initial relative position (r_x,r_y,r_z) and final relative position (r_x', r_y', r_z'), such that $0 < r_x < N_x$, where N_x is the number of cells in the x-direction, the current density J_x at Yee mesh node (i,j,k) is given by

$$J_x[i,j,k] = C * SD_x[i](rx,rx') * (\tfrac{1}{2}*S_y[j](r_y)*S_z[k](r_z') + S_y[j](r_y')*S_z[k](r_z') + S_y[j](r_y)*S_z[k](r_z) + \tfrac{1}{2}*S_y[j](r_y')*S_z[k](r_z))$$

J_y and J_z are cyclic permutations. Where C is a constant based on the charge of the particle, the grid spacing, and the time step, and S is a transverse cofactor

$$S_d[i_d](r_d) = \begin{cases} (1-(r_d-i)) & \text{if } floor(r_d) = i_d \\ (r_d-o) & \text{if } ceiling(r_d) = i_d \\ 0 & \text{else} \end{cases}$$

Which can be re-written to avoid if-statements by using min and max functions

$$S_d[i,r_d] = min(max(1 - r_d + i_d, 0), max(r_d - i_d + 1, 0))$$

$SD_d[i_d](r_d,r_d')$ is the longitudinal cofactor, and can be written as an expression that depends on whether the particle crossed a cell boundary in the d-direction

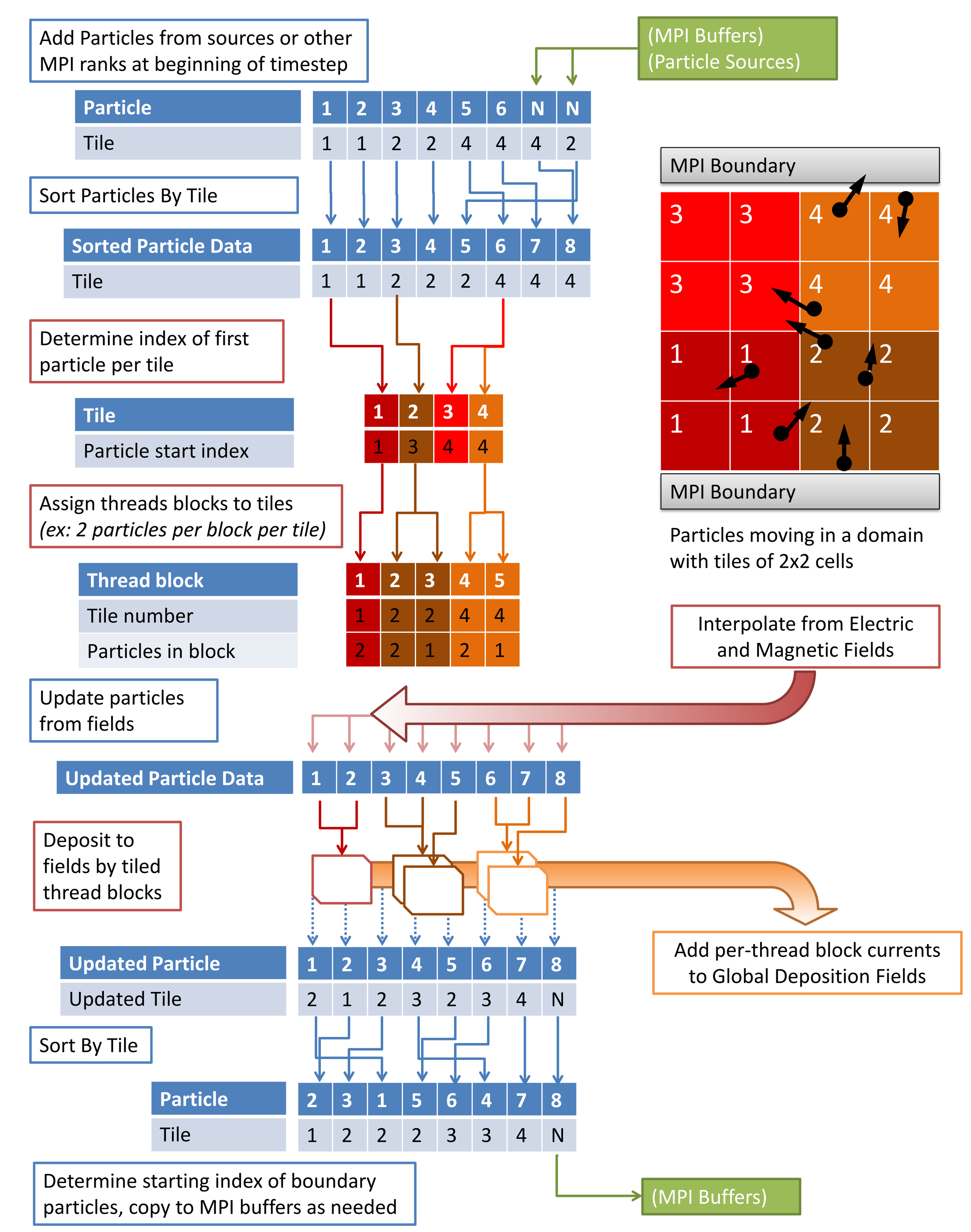
$$SD_d[i_d](r_d,r_d') = \begin{cases} (1-(r_d'-i_d)) & \text{if } (id = floor(r_d)-1) \text{ and } (\Delta d < 0) \\ (r_d-i_d) & \text{if } (i_d = floor(r_d)) \text{ and } (\Delta d < 0) \\ (r_d'-r_d) & \text{if } (i_d = floor(r_d)) \text{ and } (\Delta d = 0) \\ (1-(r_d-i_d)) & \text{if } (i_d = floor(r_d)) \text{ and } (\Delta d > 0) \\ -(r_d'-i_d) & \text{if } (i_d = floor(r_d)+1) \text{ and } (\Delta d > 0) \\ 0 & \text{else} \end{cases}$$

(Where Δd is the change in the particle's cell index in the d direction)

Which can similarly be re-written to avoid if-statements

$$SD_d[i_d,r_d,r_d'] = sign(r_d'-r_d)*(min(1 + i_d, max(i_d, max(r_d,r_d'))) - max(i_d, min(i_d + 1, min(r_d,r_d'))))$$

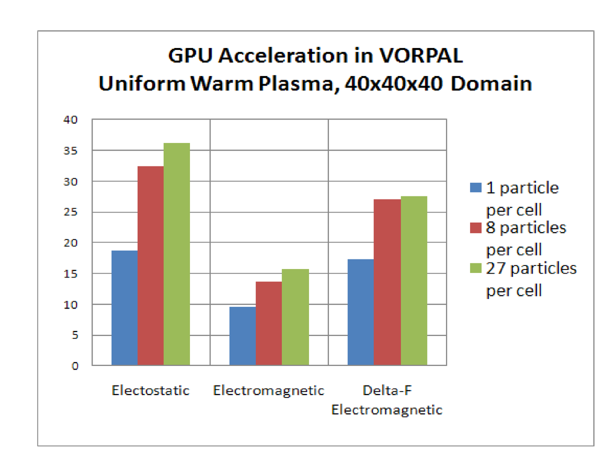
GPU Update Sequence

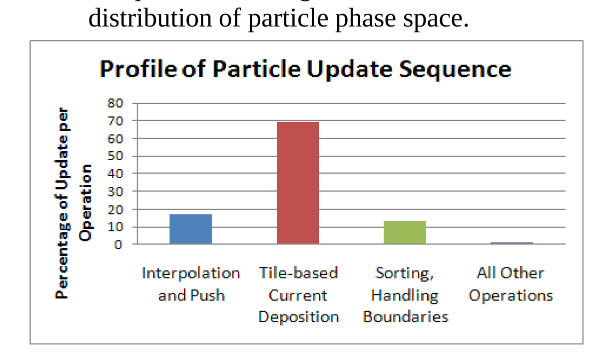


Performance

This scheme is implemented for:

- 3D Electostatic PIC
  - 8 deposits, 6 interpolations per particle
- 3D Electromagnetic PIC
  - 12-48 deposits , 6 interpolations per particle
- 3D Delta-F Electromagnetic PIC
  - 12-48 deposits, 12 interpolations per particle
  - Requires evaluating a Maxwellian distribution of particle phase space.

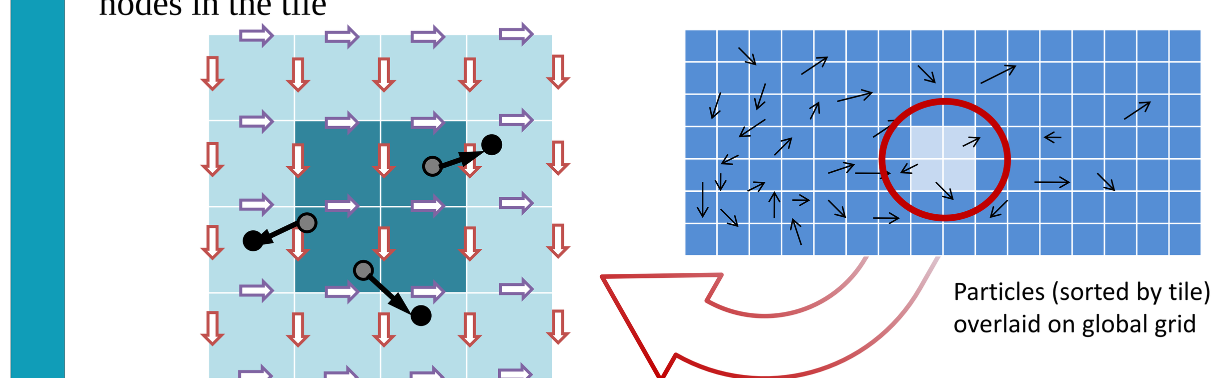




A profile of the code indicates that the majority of time is spent in the deposition step, and not in the particle push or sorts.

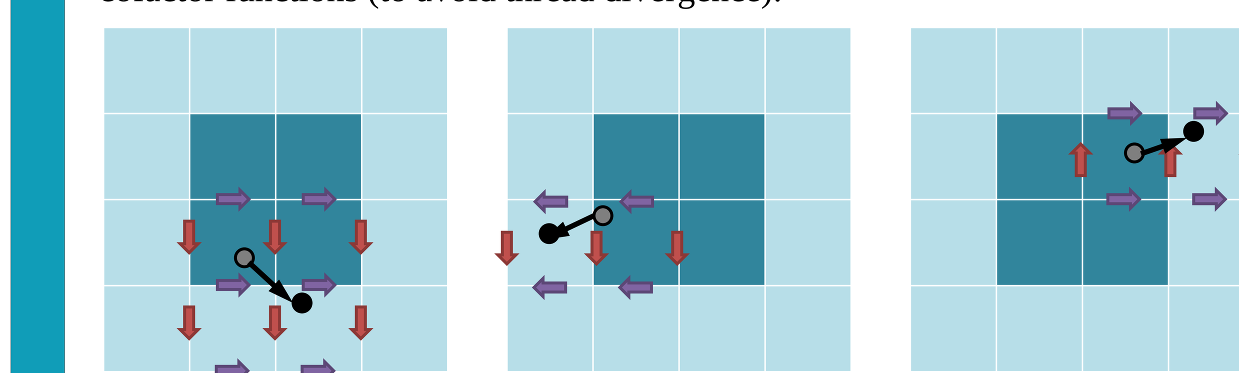
Deposition Kernel

Step 1: For each thread block, store tile data and relevant particles in shared memory. Each thread (x,y) will accumulate current in registers for all (x,y,z) nodes in the tile



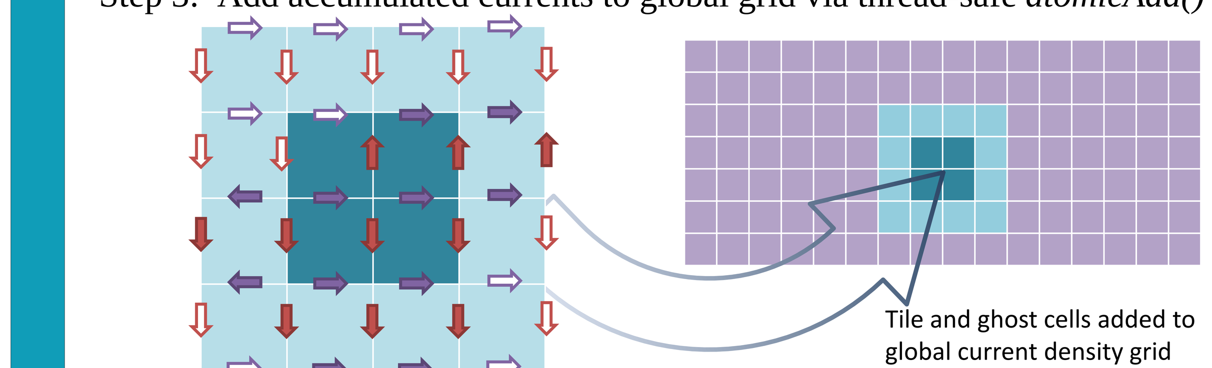
Particles buffered in shared memory. J_x and J_y stored in registers

Step 2: Iterate over buffered particles. For each particle, thread (x,y) computes deposition for relevant (x,y,z) nodes via floating-point-heavy Esirkepov cofactor functions (to avoid thread divergence).

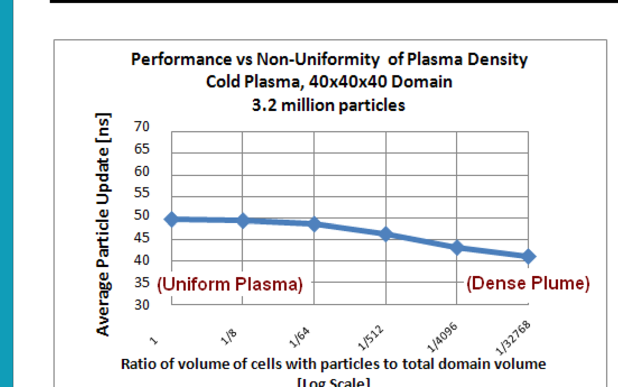


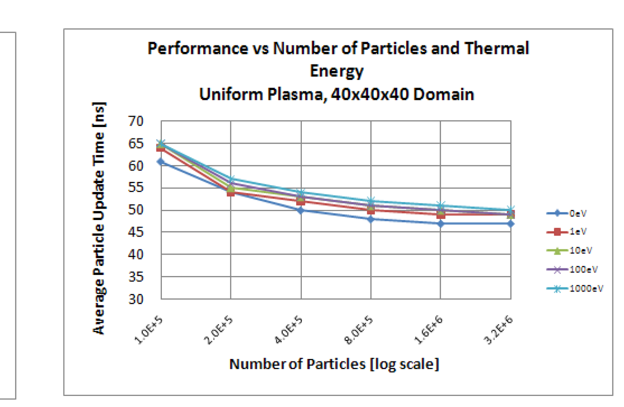
Accumulating J_x and J_y values for each particle. Some thread warps may be inactive for a given particle, and will start processing another particle

Step 3: Add accumulated currents to global grid via thread-safe *atomicAdd()*



Performance and Future Work





- 3D Performance is nearly equal for cold and hot plasmas. Performance actually slightly increases for inhomogeneous plasmas like particle plumes,
- Multi-GPU performance is severely limited by un-optimized MPI messaging scheme designed for conventional CPU simulations
  - Primary area of ongoing development
- Additional future work: support for 2nd-order Esirkepov schemes and complex cut-cell boundaries

NVIDIA GPU Technology Conference
May 14th – May 17th, 2012

Related Work
(1) Xianglong Kong, Michael C. Huang, Chuang Ren, Viktor K. Decyk, Particle-in-cell simulations with charge-conserving current deposition on graphic processing units, Journal of Computational Physics, Volume 230, Issue 4, 20 February 2011, Pages 1676-1685, ISSN 0021-9991, 10.1016/j.jcp.2010.11.032.
Villasenor-Buneman current deposition for 2D PIC implementation within the OSIRIS framework
(2) (2) Abreu, P.; Fonseca, R.A.; Pereira, J.M.; Silva, L.O.; , "PIC Codes in New Processors: A Full Relativistic PIC Code in CUDA-Enabled Hardware With Direct Visualization," Plasma Science, IEEE Transactions on , vol.39, no.2, pp.675-685, Feb. 2011
Esirkepov current deposition for a 2D GPU PIC code
(3) Burau, H., Widera, R., HÃ¶nig, W., Juckeland, G., Debus, A., Kluge, T., Schramm, U., Cowan, T.E., Sauerbrey, R., Bussmann, M. PIConGPU: A fully relativistic particle-in-cell code for a GPU cluster (2010) IEEE Transactions on Plasma Science, 38 (10 PART 2), art. no. 5556015, pp. 2831-2839.
Relativistic PIC for a GPU cluster
(4) Stantchev, G., Dorland, W., Gumerov, N. Fast parallel Particle-To-Grid interpolation for plasma PIC simulations on the GPU (2008) Journal of Parallel and Distributed Computing, 68 (10), pp. 1339-1349.
O(N) complexity charge deposition for PIC on GPU using an in-place particle sorting algorithm

Work supported by Tech-X Corporation