High-Performance Simulations of Coherent Synchrotron Radiation on Multicore GPU and CPU Platforms

Balša Terzić, PhD

Department of Physics, Old Dominion University

Center for Accelerator Studies (CAS), Old Dominion University

2015 IPAC, Richmond, 4 May 2015

Collaborators

Center for Accelerator Science (CAS) at Old Dominion University (ODU):

Professors:

Physics: Alexander Godunov

Computer Science: Mohammad Zubair, Desh Ranjan

PhD student:

Computer Science: Kamesh Arumugam

Early advances on this project benefited from my collaboration with Rui Li (Jefferson Lab)

Outline

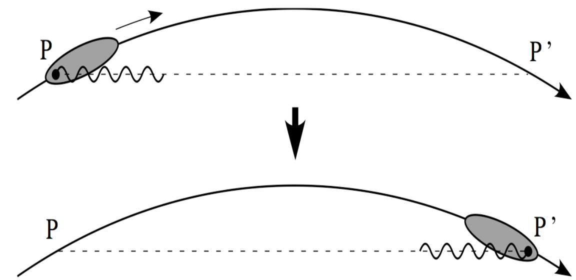
- Coherent Synchrotron Radiation (CSR)
  - Physical problem
  - Computational challenges
- New 2D Particle-In-Cell CSR Code
  - Outline of the new algorithm
  - Parallel implementation CPU/GPU clusters
  - Benchmarking against analytical results
- Still to Come
- Summary

CSR: Physical Problem

- Beam's self-interaction due to CSR can lead to a host of adverse effects
  - Increase in energy spread
  - Emittance degradation
  - Longitudinal instability (micro-bunching)
- Being able to quantitatively simulate CSR is the first step toward mitigating its adverse effects
- It is vitally important to have a trustworthy 2D CSR code

CSR: Computational Challenges

- Dynamics governed by the Lorentz force:



$$\frac{d}{dt}\left(\gamma m_e \vec{v}\right) = e\left(\vec{E} + \vec{\beta}\times\vec{B}\right)$$

LARGE CANCELLATION

$$\vec{\beta} = \frac{\vec{v}}{c}$$

$$\vec{E} = \vec{E}^{ext} + \vec{E}^{self}$$

$$\vec{B} = \vec{B}^{ext} + \vec{B}^{self}$$

- $\vec{E}^{ext}, \vec{B}^{ext}$: external EM fields
- $\vec{E}^{self}, \vec{B}^{self}$: self-interaction (CSR)

$$\vec{E}^{self} = -\vec{\nabla}\phi - \frac{1}{c}\frac{\partial\vec{A}}{\partial t}$$

$$\vec{B}^{self} = \vec{\nabla}\times\vec{A}$$

NUMERICAL NOISE DUE TO GRADIENTS

$$\begin{bmatrix} \phi(\vec{r},t) \\ \vec{A}(\vec{r},t) \end{bmatrix} = \int \begin{bmatrix} \rho(\vec{r}\,',t') \\ \vec{J}(\vec{r}\,',t') \end{bmatrix} \frac{d\vec{r}\,'}{\left|\vec{r}-\vec{r}\,'\right|}$$

retarded potentials

$$t' = t - \frac{\left|\vec{r}-\vec{r}\,'\right|}{c}$$

retarded time

ACCURATE 2D INTEGRATION

Charge density: $\rho(\vec{r},t) = \int f(\vec{r},\vec{v},t)\, d\vec{v}$

Current density: $\vec{J}(\vec{r},t) = \int \vec{v}\, f(\vec{r},\vec{v},t)\, d\vec{v}$

Beam distribution function (DF): $f(\vec{r},\vec{v},t)$

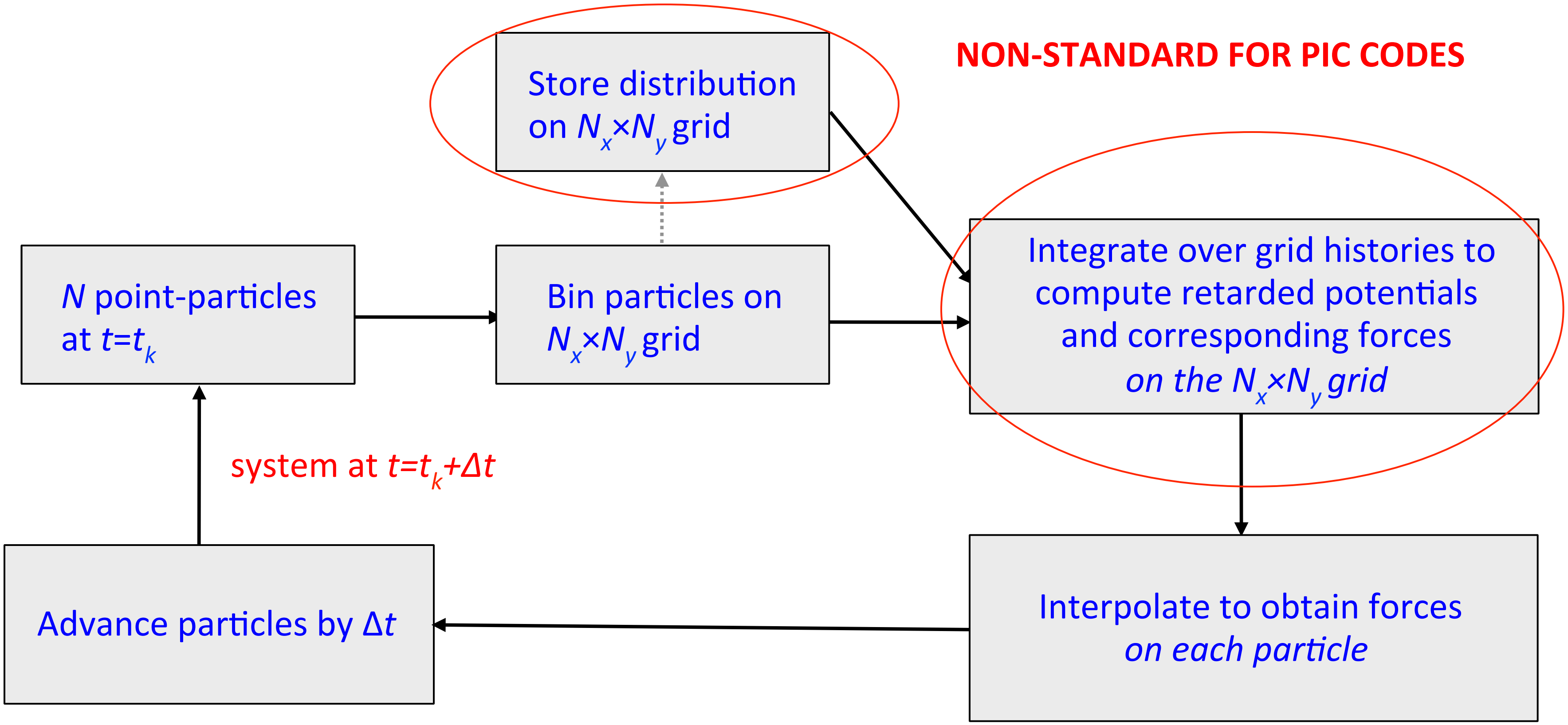
Need to track the entire history of the bunch

ENORMOUS COMPUTATIONAL AND MEMORY LOAD

CSR: Computational Challenges

- Our new code solves the main computational challenges associated with the numerical simulation of CSR effects
  - Enormous computational and memory load
    (storing and integration over beam's history)
    Parallel implementation on GPU/CPU platforms
  - Large cancellation in the Lorentz force
    Developed high-accuracy, adaptive multidimensional integrator for GPUs
  - Scaling of the beam self-interaction
    Particle-in-Cell (PIC) code
    - Self-interaction in PIC codes scales as grid resolution squared
      (Point-to-point codes: scales as number of macroparticles squared)
  - Numerical noise
    Noise removal using wavelets

New Code: The Big Picture

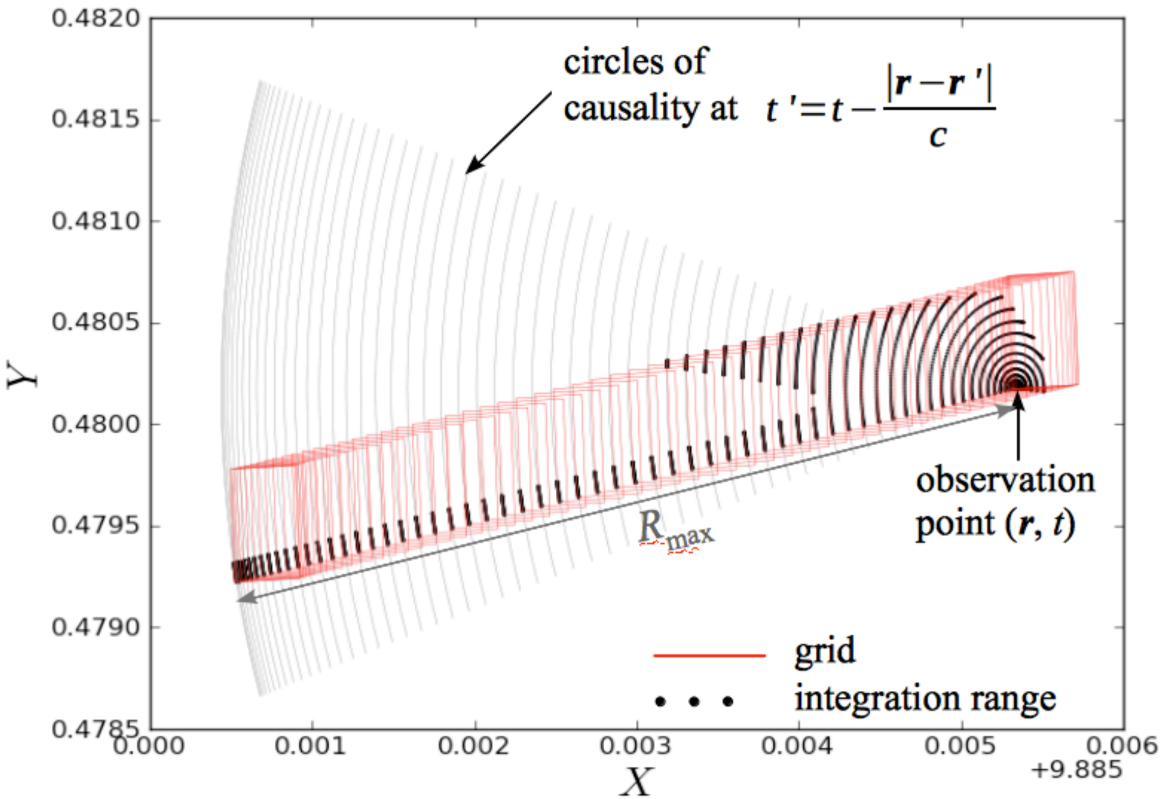


New Code: Computing Retarded Potentials

- Carry out integration over history:

$$\begin{bmatrix} \phi(\vec{r},t) \\ \vec{A}(\vec{r},t) \end{bmatrix} = \int \begin{bmatrix} \rho\left(\vec{r}',t-\frac{R'}{c}\right) \\ \vec{J}\left(\vec{r}',t-\frac{R'}{c}\right) \end{bmatrix} \frac{d\vec{r}'}{|\vec{r}-\vec{r}'|} = \sum_{i=1}^{M_{\rm int}} \int_0^{R_{\max}} \int_{\theta^i_{\min}}^{\theta^i_{\max}} \begin{bmatrix} \rho\left(\vec{r}',t-\frac{R'}{c}\right) \\ \vec{J}\left(\vec{r}',t-\frac{R'}{c}\right) \end{bmatrix} dR'd\theta'.$$

- Determine limits of integration in lab frame:



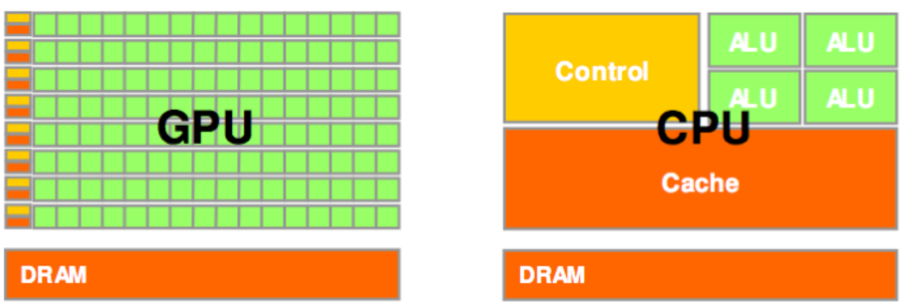
compute R_{max} and $(\vartheta_{min}{}^{i}, \vartheta_{max}{}^{i})$

For each gridpoint, <u>independently</u>, do the same integration over beam's history

Obvious candidate for parallel computation

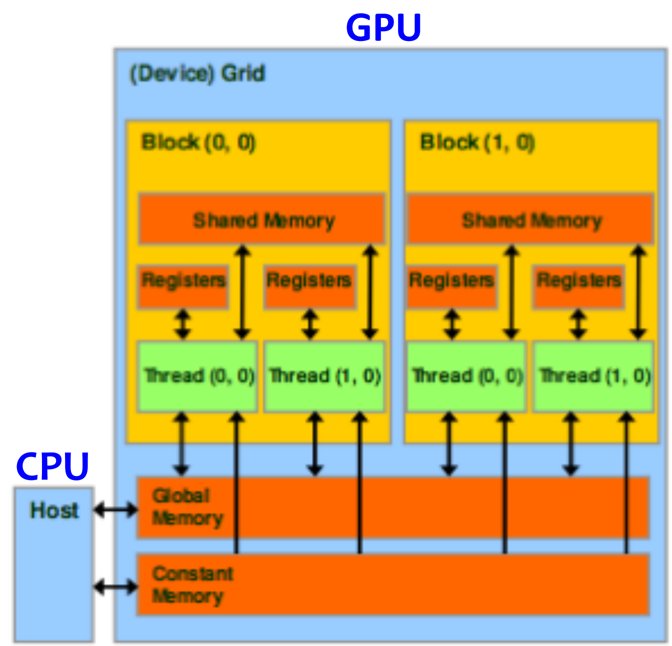
Parallel Computation on GPUs

- Parallel computation on GPUs
  - Ideally suited for algorithms with *high arithmetic operation/memory access ratio*
  - Same Instruction Multiple Data (SIMD)
  - *Several types of memories* with varying access times (global, shared, registers)
  - Uses extension to existing programming languages to handle new architecture
  - GPUs have many smaller cores (~400-500) designed for parallel execution
  - *Avoid branching and communication* between computational threads



More space for ALU, less for cache and flow control

GPU:
grid → blocks → threads



Example: NVIDIA GeForce GTX 480 GPU has 448 cores

Parallel Computation on GPUs

- Computing the retarded potentials requires integrating over the entire bunch history – *very slow*! Must parallelize.

- Integration over a grid is ideally suited for GPUs
  - No need for communication between gridpoints
  - Same *kernel* executed for all
  - Can remove all branches from the algorithm

- We designed a new adaptive multidimensional integration algorithm optimized for GPUs
  [Arumugam, Godunov, Ranjan, Terzić & Zubair 2013a,b]
  - NVIDIA's CUDA framework (extension to C++)
  - About 2 orders of magnitude speedup over a serial implementation
  - Useful beyond this project

Performance Comparison: CPU Vs. GPU

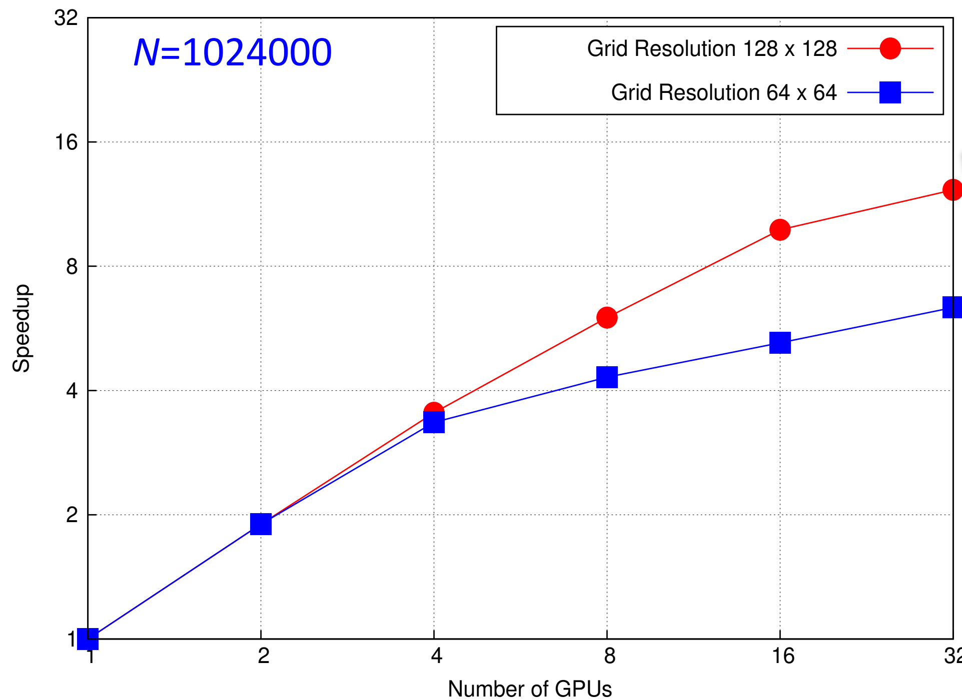
- Comparison: 1 CPU vs. 1 GPU; 8 CPUs vs. 4 GPUs (one compute node)

Number of Particles (N)	Grid Resolution	Multicore CPU implementation: Single Core Time (sec.)	8 cores Time (sec.)	8 cores Speedup	GPU implementation on a standalone system with: Single GPU Time (sec.)	Single GPU Speedup	4 GPUs Time (sec.)	4 GPUs Speedup
102400	32×32	73.5	11.1	6.6	1.5	49.0	0.7	105.0
	64×64	878.5	116.2	7.6	16.8	52.3	4.7	186.9
	128×128	13123.2	1695.3	7.7	246.8	53.2	68.4	191.9
1024000	32×32	58.1	12.7	4.6	1.2	48.4	0.6	96.8
	64×64	573.9	83.9	6.8	11.1	51.7	3.2	179.3
	128×128	7651.5	1000.9	7.6	144.1	53.1	40.1	190.8
4096000	32×32	57.8	11.9	4.9	1.3	44.5	0.6	96.3
	64×64	452.8	66.5	6.8	9.2	49.2	2.4	188.7
	128×128	5307.5	725.3	7.3	101.4	52.3	27.1	195.9

- 1 GPU over 50 x faster than 1 CPU
- Both linearly scale with multicores: 4 GPUs 25x faster than 8 CPUs
- Hybrid CPU/GPU implementation marginally better than GPUs alone
- Execution time *reduces* as the number of point-particles grows
  - More particles, less numerical noise, fewer function evaluations needed for high-accuracy integration

GPU Cluster Implementation

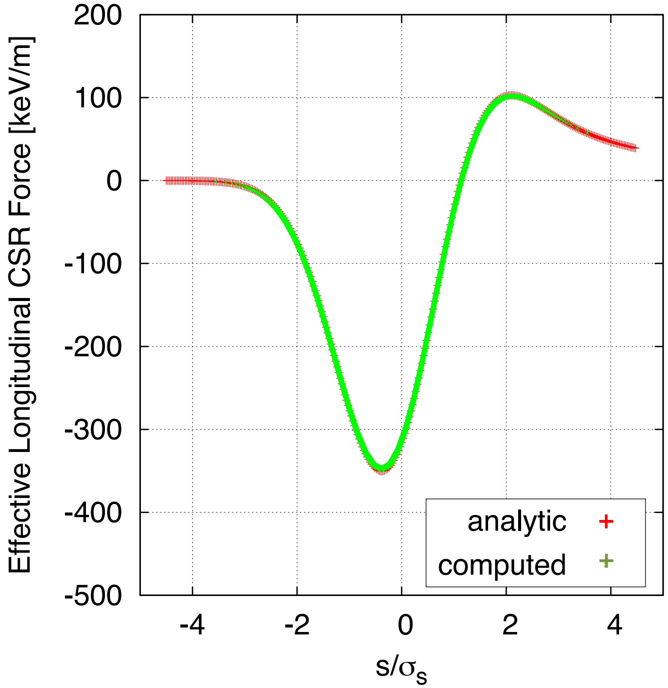
- The higher the resolution, the larger the fraction of time spent on computing integrals (and therefore the speedup)
  - We expect the scaling at larger resolutions to be nearly linear
  - 1 step of the simulation on a 128x128 grid and 32 GPUs: ~ 10 s



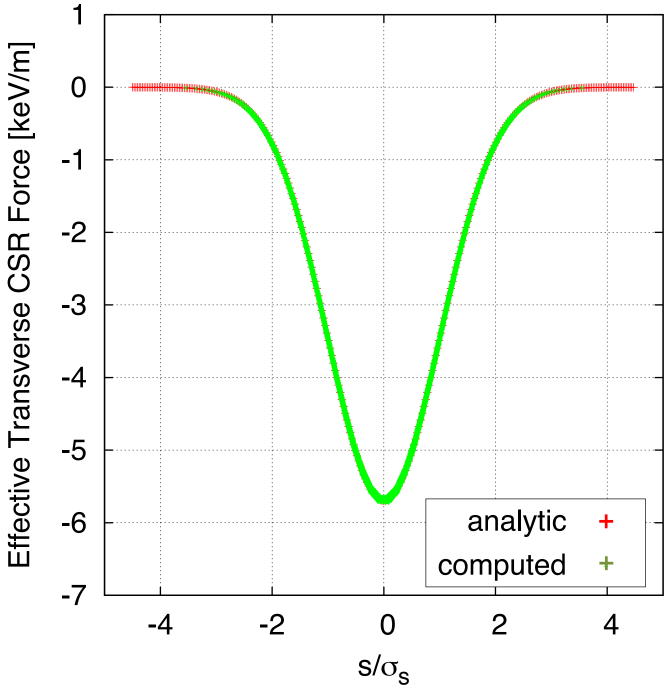
Benchmarking Against Analytic 1D Results

- Analytic steady state solution available for a rigid line Gaussian bunch [Derbenev & Shiltsev 1996, SLAC-Pub 7181]

LONGITUDINAL



TRANSVERSE



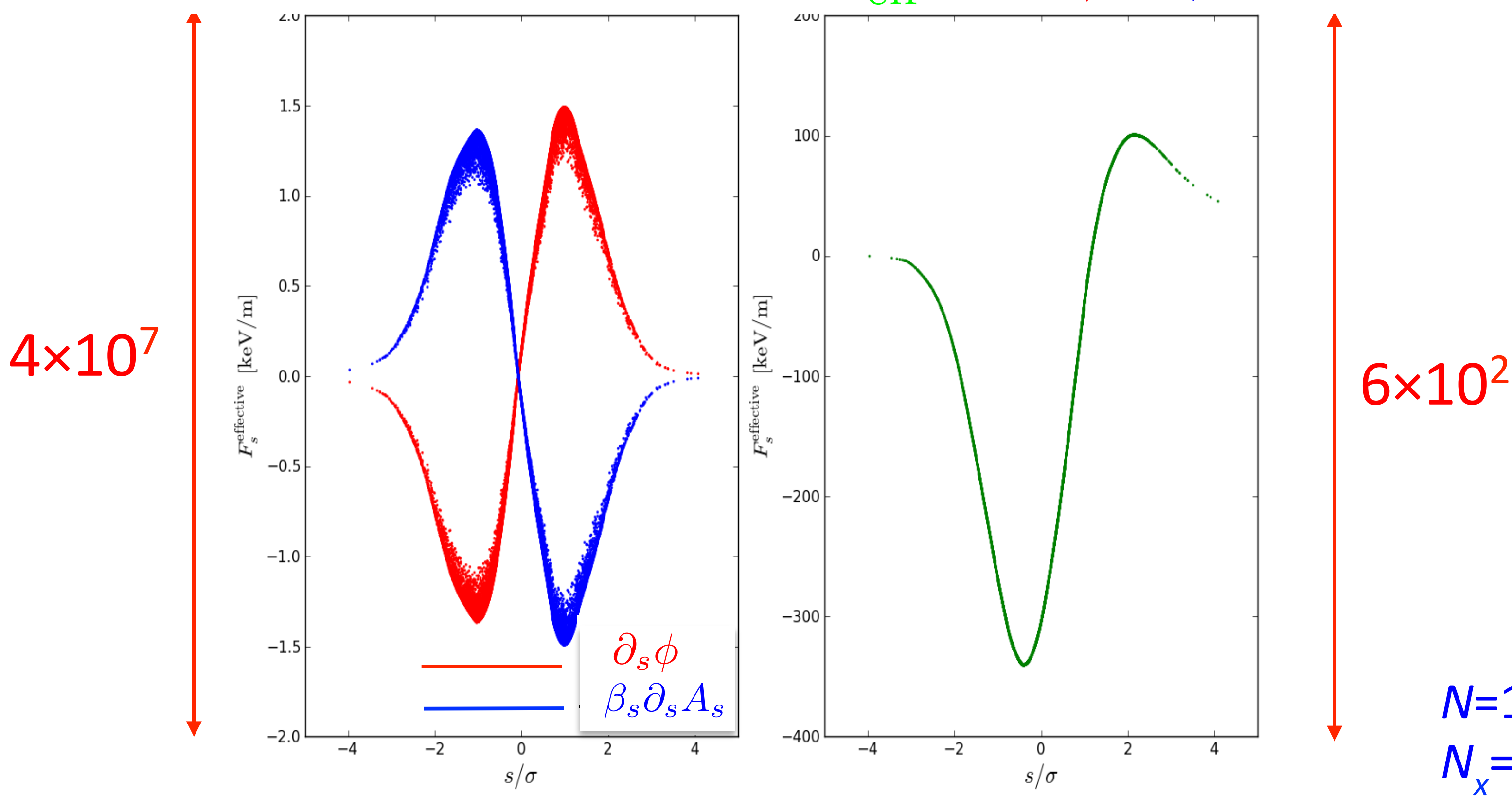
N=512000
N_x=N_y=64

- Excellent agreement between analytic and computed solutions provides a **proof of concept for the new code**

Large Cancellation in the Lorentz Force

- Traditionally difficult to track large quantities which mostly cancel out:

Effective Longitudinal Force: $F^s_{\mathrm{eff}} = \partial_s \phi - \beta_s \partial_s A_s$



N=128000
N_x=N_y=32

- High accuracy of the implementation able to track accurately these cancellations over 5 orders of magnitude

Efforts Currently Underway

- Compare to 2D semi-analytical results (chirped bunch)
  [Li 2008, PR STAB 11, 024401]

- Compare to other 2D codes (for instance Bassi *et al*. 2009)

- Simulate a test chicane

- Further Afield:
  - Various boundary conditions
  - Shielding
  - Use wavelets to remove numerical noise (increase efficiency and accuracy)
  - Explore the need and feasibility of generalizing the code from 2D to 3D

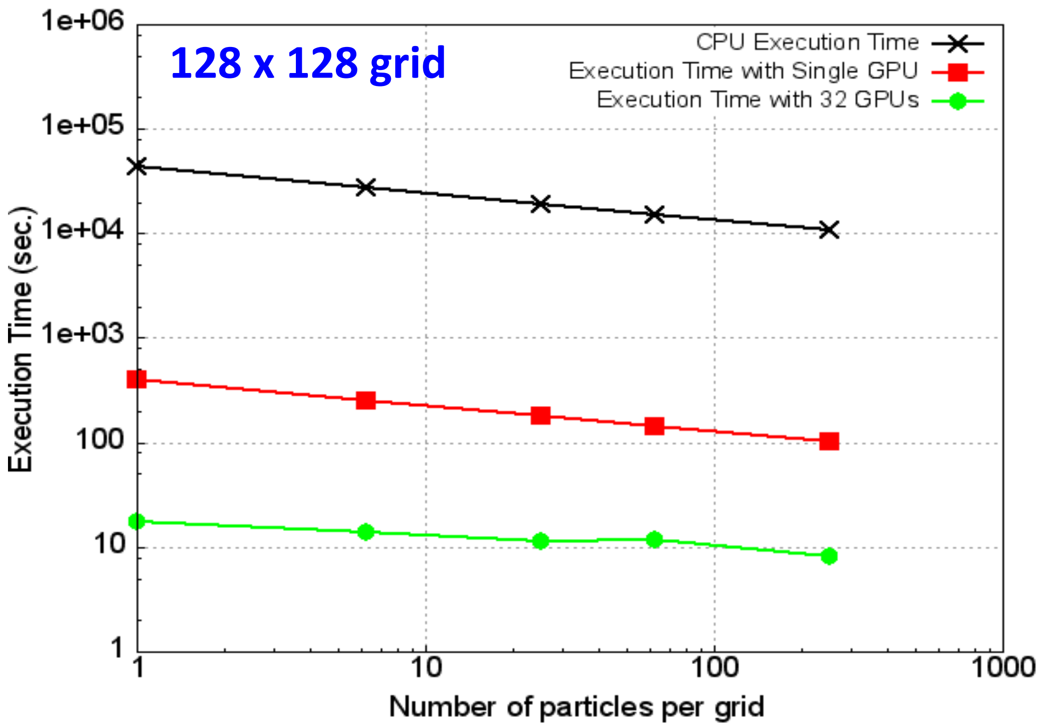
Summary

- Presented the new 2D PIC code:
  - Resolves traditional computational difficulties by optimizing our algorithm on a GPU platform
  - Proof of concept: excellent agreement with analytical 1D results
- Outlined outstanding issues that will soon be implemented
- Closing in on our goal
  - Accurate and efficient code which faithfully simulates CSR effects

Backup Slides

Importance of Numerical Noise

- Signal-to-noise ratio in PIC simulations scales as $N_{ppc}^{1/2}$
  [Terzić, Pogorelov & Bohn 2007, PR STAB 10, 034021]
  - Then the numerical noise scales as $N_{ppc}^{-1/2}$ (N_{ppc}: avg. # of particles per cell)



Execution time for integral evaluation also scales as $N_{ppc}^{-1/2}$

Less numerical noise = more accurate and faster simulations

[Terzić, Pogorelov & Bohn 2007, PR STAB 10, 034021; Terzić & Bassi 2011, PR STAB 14, 070701]

Wavelet Denoising and Compression

- When the signal is known, one can compute *Signal-to-Noise Ratio* (*SNR*):

$$SNR = \sqrt{N_{ppc}}$$

N_{ppc}: avg. # of particles per cell $\quad N_{ppc} = N/N_{\text{cells}}$

$$SNR = \sqrt{\frac{\sum_{i=1}^{N_{grid}} \bar{q}_i^2}{\sum_{i=1}^{N_{grid}} \left(q_i - \bar{q}_i\right)^2}}$$

$\bar{q}_i \quad exact$

$q_i \quad grid$

2D superimposed Gaussians on 256×256 grid

COMPACT: only 0.12% of coeffs

WAVELET THRESHOLDING

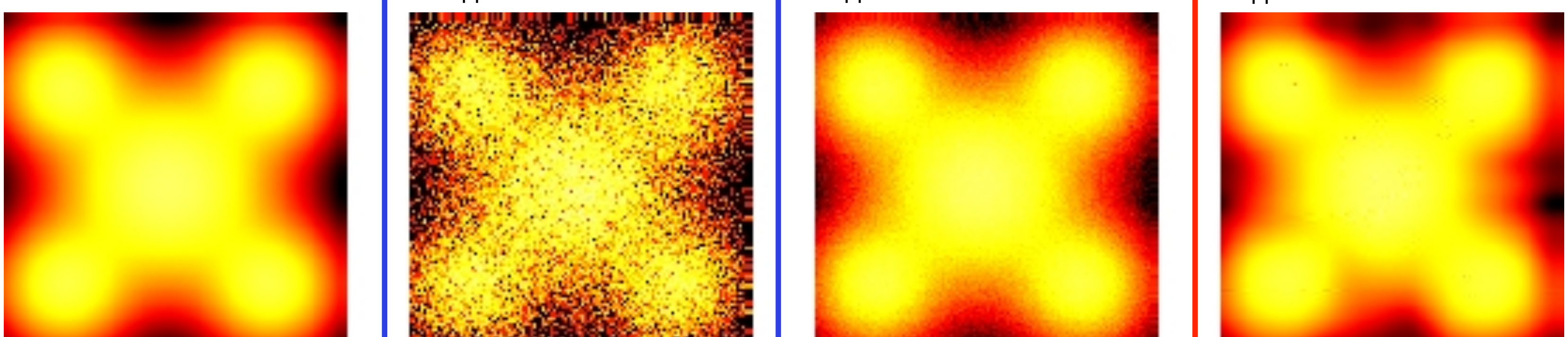
DENOISED

ANALYTICAL

N_{ppc}=3 $\;SNR$=2.02

N_{ppc}=205 $\;SNR$=16.89

N_{ppc}=3 $\;SNR$=16.83



Wavelet denoising yields a representation which is:

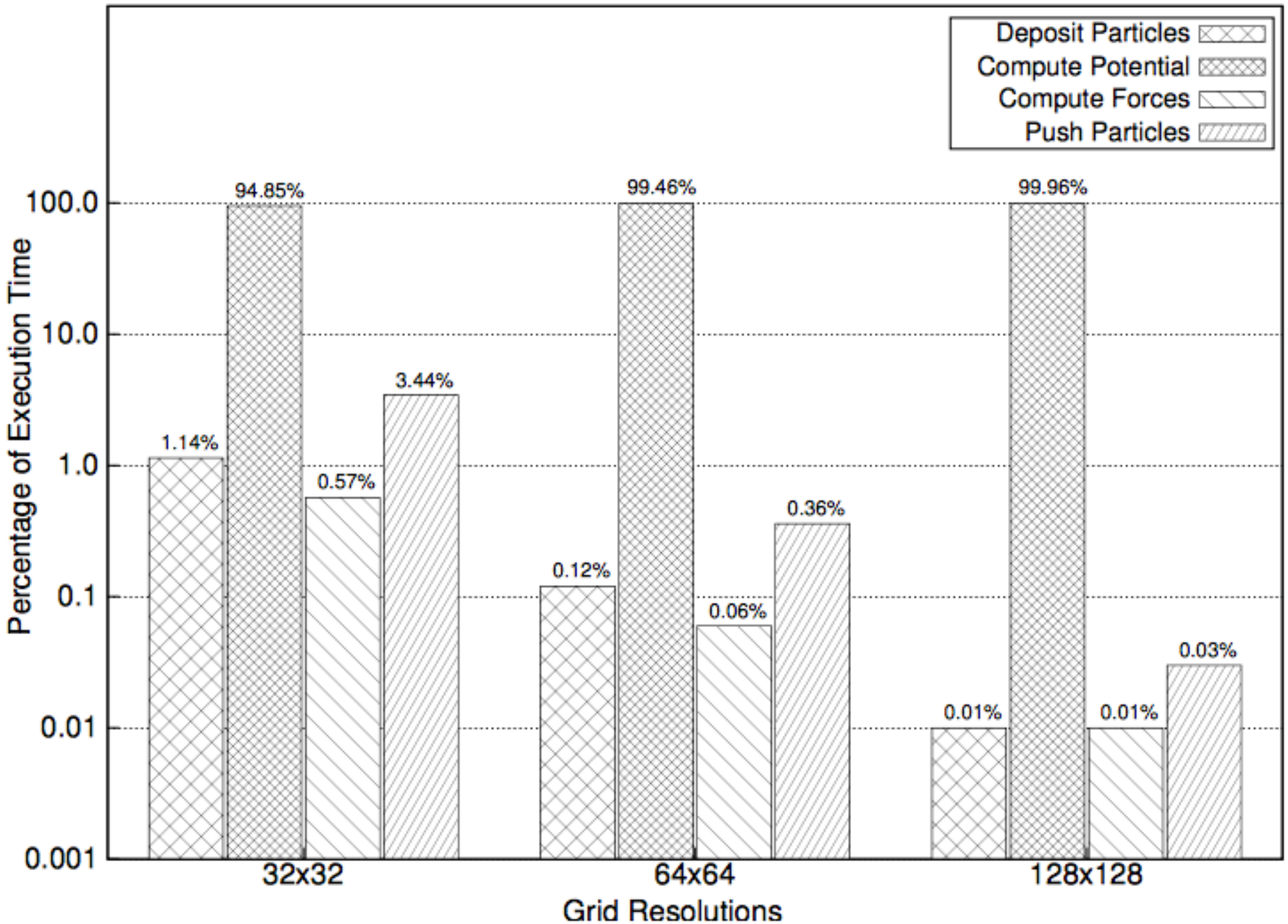
- Appreciably more accurate than non-denoised representation
- Sparse (if clever, we can translate this sparsity into computational efficiency)

Performance Comparison: GPU Vs. Hybrid CPU/GPU

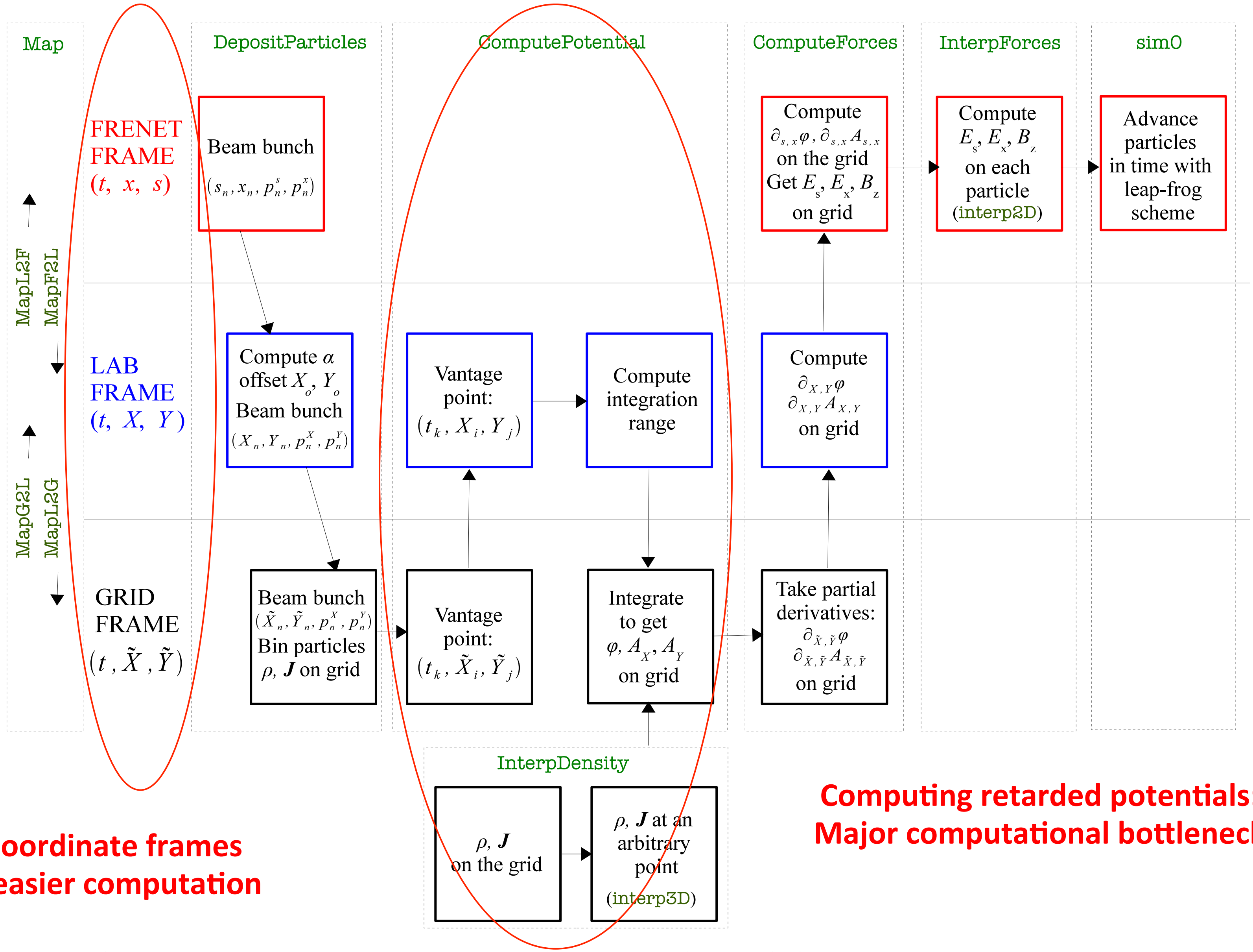
- Comparison: 1 CPU vs. 1 GPU; 8 CPUs vs. 4 GPUs (one compute node)
- Hybrid CPU/GPU implementation marginally better than GPUs alone

Number of Particles (N)	Grid Resolution	GPU implementation on a standalone system with Single GPU		GPU implementation on a standalone system with 4 GPUs		Hybrid implementation on multicore CPU with 4 GPUs	
		Time (sec.)	Speedup	Time (sec.)	Speedup	Time (sec.)	Speedup
102400	32×32	1.5	49.0	0.7	105.0	0.7	105.0
	64×64	16.8	52.3	4.7	186.9	4.5	195.2
	128×128	246.8	53.2	68.4	191.9	65.8	199.4
1024000	32×32	1.2	48.4	0.6	96.8	0.6	96.8
	64×64	11.1	51.7	3.2	179.3	3.1	185.1
	128×128	144.1	53.1	40.1	190.8	38.6	198.2
4096000	32×32	1.3	44.5	0.6	96.3	0.6	96.3
	64×64	9.2	49.2	2.4	188.7	2.3	196.8
	128×128	101.4	52.3	27.1	195.9	26.1	203.4

Breakdown of Computations



New Code: Computation of CSR Effects

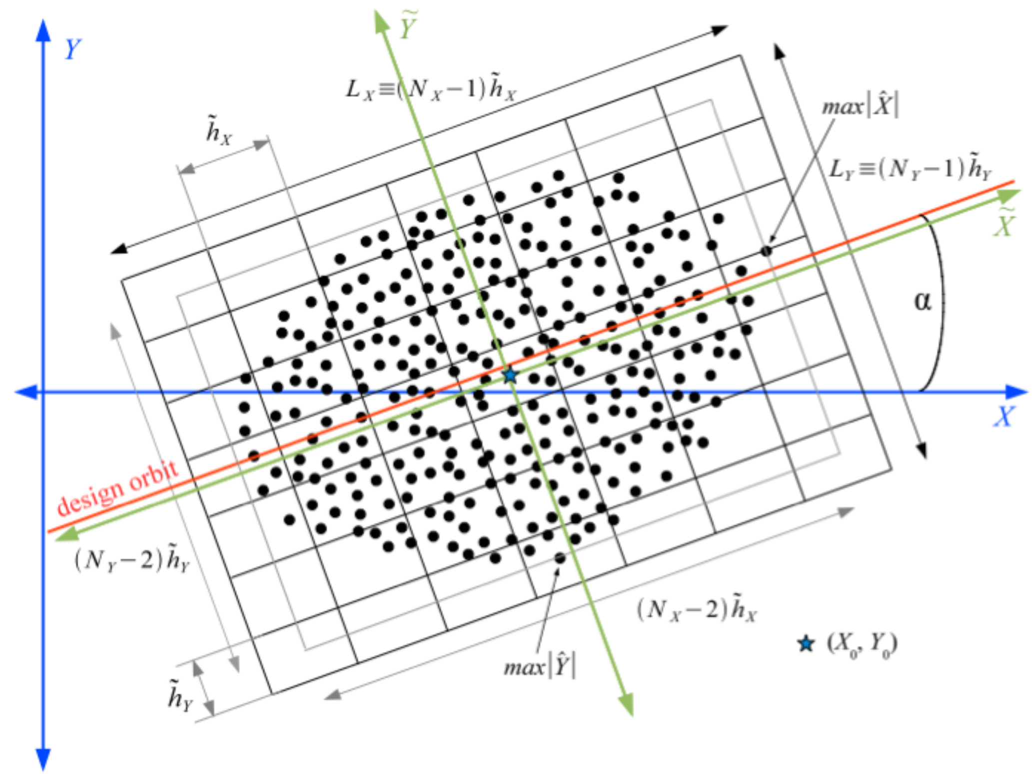


Computing retarded potentials: Major computational bottleneck

3 coordinate frames for easier computation

New Code: Particle-In-Cell

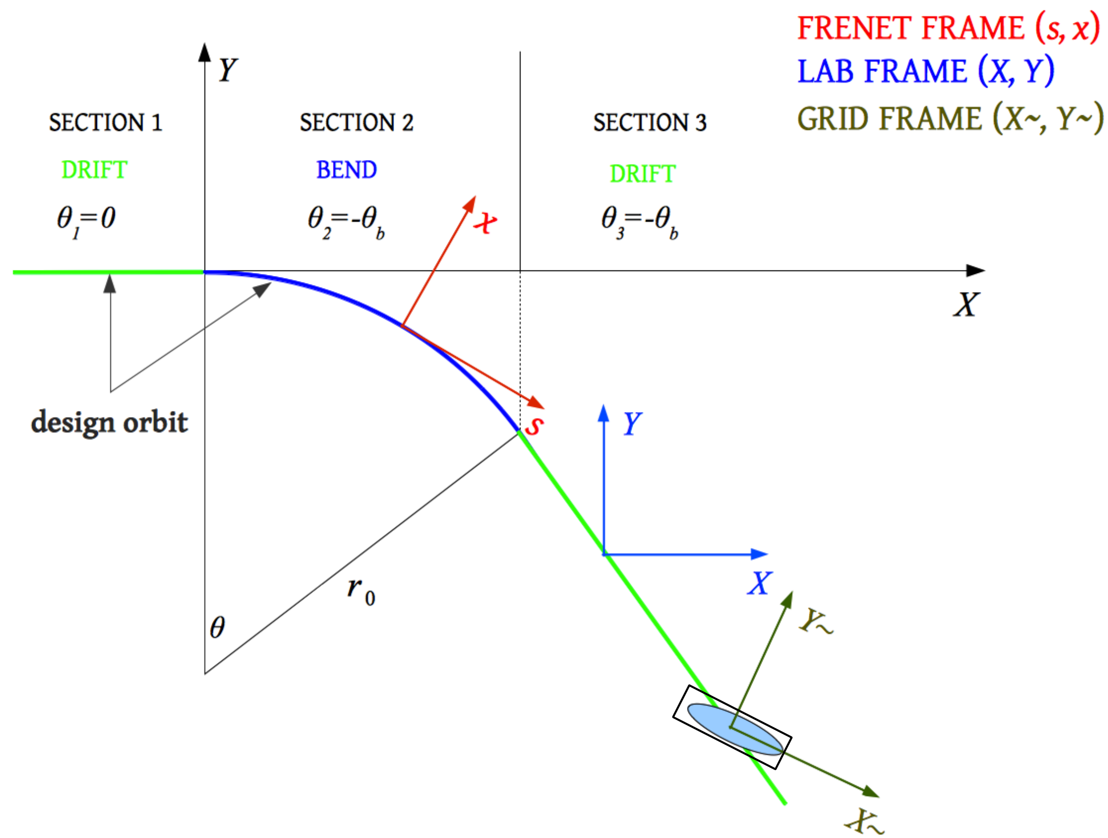
- Grid resolution is specified *a priori* (fixed grid)
  - N_X : # of gridpoints in X
  - N_Y : # of gridpoints in Y
  - $N_{grid}=N_X \times N_Y$ total gridpts
  - Grid: $\left[X_{ij}, Y_{ij}\right]_{j=1,N_y}^{i=1,N_x}$
  - Inclination angle α
  - Point-particles deposited on the grid via deposition scheme



- Grid is determined so as to tightly envelope all particles
  Minimizing number of empty cells $\Rightarrow$ optimizing spatial resolution

New Code: Frames of Reference

- Choosing a correct coordinate system is of crucial importance
- To simplify calculations use 3 frames of reference:
  - Frenet frame (*s*, *x*)
    s – along design orbit
    x – deviation normal to direction of motion
    - Particle push
  - Lab frame (*X*, *Y*)
    - Integration range
    - Integration of retarded potentials
  - Grid frame (*X~*,*Y~*)
    Scaled & rotated lab frame always [-0.5,0.5] × [-0.5,0.5]
    - Particle deposition
    - Grid interpolation
    - History of the beam



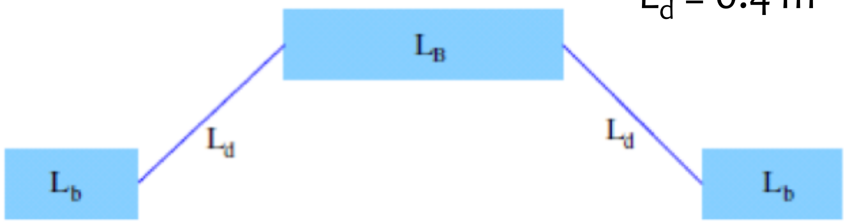
Semi-Analytic 2D Results: 1D Model Breaks Down

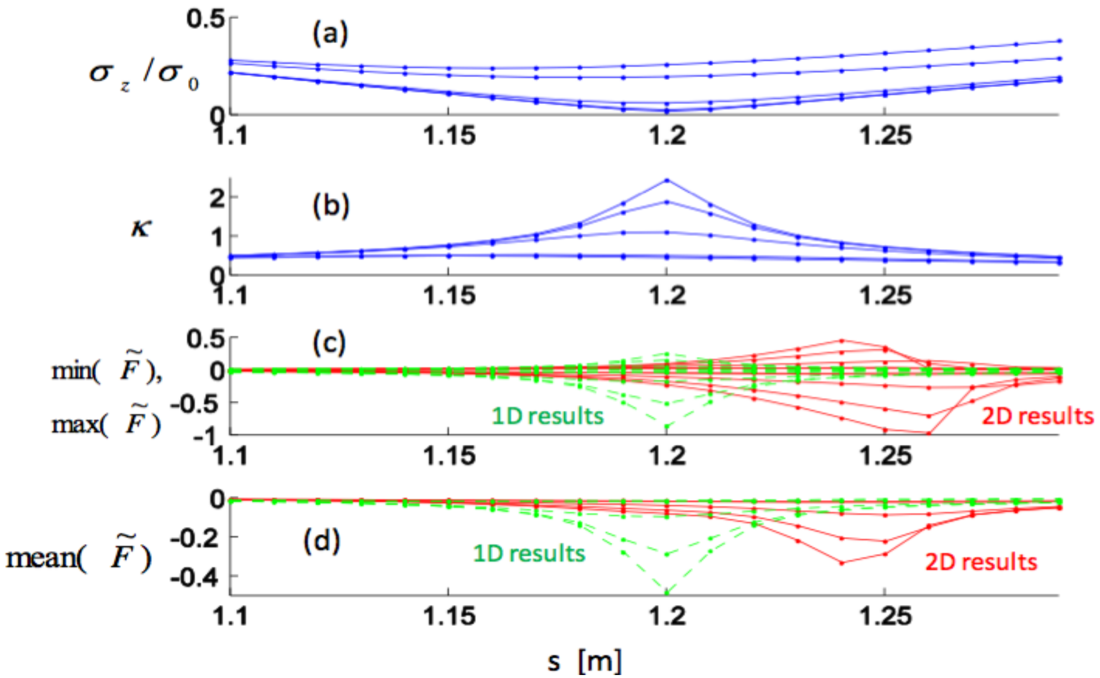
- Analytic steady state solution is justified for $\kappa = \dfrac{\sigma_x}{\left(R\sigma_z^2\right)^{1/3}} << 1$ [Derbenev & Shiltsev 1996]
- Li, Legg, Terzić, Bisognano & Bosch 2011:

Model bunch compressor (chicane)

E = 70 MeV
σ_{z0}= 0.5 mm
u = -10.56 m^{-1} energy chirp

L_b = 0.3 m
L_B = 0.6 m
L_d = 0.4 m





1D & 2D disagree in:

Magnitude of CSR force
Location of maximum force

⇒ **1D CSR model is inadequate**

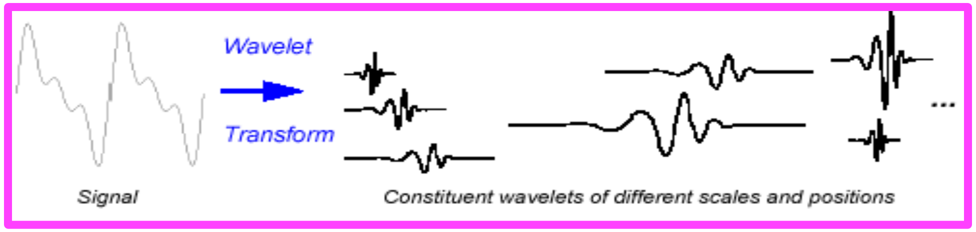
Preliminary simulations show good agreement between 2D semi-analytic results and results obtained with our code

Wavelets

- Orthogonal basis of functions composed of scaled and translated versions of the same localized *mother wavelet ψ(x) and the scaling function ϕ(x)*:

$$\psi_i^k(x) = 2^{k/2}\psi(2^k x - i), \quad k, i \in Z$$

$$f(x) = s_0^0 \phi_0^0(x) + \sum_k \sum_i d_i^k \psi_i^k(x),$$



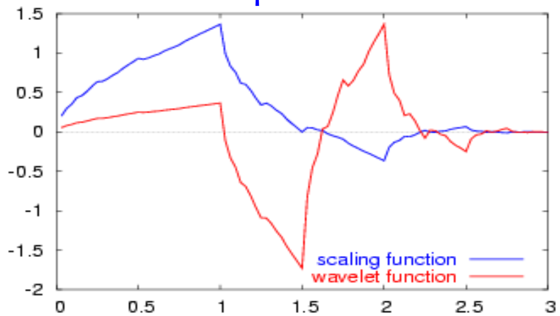
- Each new resolution level *k* is orthogonal to the previous levels

- *Compact support*: finite domain over which nonzero

- In order to attain orthogonality of different scales, their shapes are strange
  - Suitable to represent irregularly shaped functions

- For discrete signals (gridded quantities), fast Discrete Wavelet Transform (DFT) is an *O(MN)* operation, *M* size of the wavelet filter, *N* signal size

Daubachies 4th order wavelet



Advantages of Wavelet Formulation

- Wavelet basis functions have compact support ⇒ signal localized in space
  Wavelet basis functions have increasing resolution levels
  ⇒ signal localized in frequency
  ⇒ *Simultaneous localization in space and frequency* (FFT only frequency)

- Wavelet basis functions correlate well with various signal types
  (including signals with singularities, cusps and other irregularities)
  ⇒ *Compact and accurate representation of data (compression)*

- Wavelet transform *preserves hierarchy of scales*

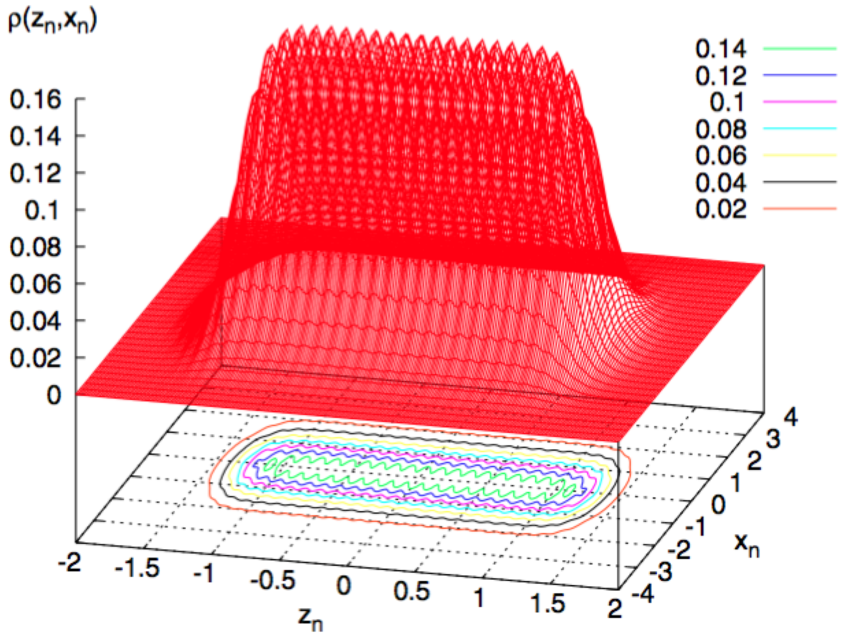
- In wavelet space, discretized operators (Laplacian) are also sparse and have an efficient preconditioner ⇒ *Solving some PDEs is faster and more accurate*

- Provide a natural setting for numerical noise removal ⇒ *Wavelet denoising*
  Wavelet thresholding: If $|w_{ij}|<T$, set $w_{ij}=0$.

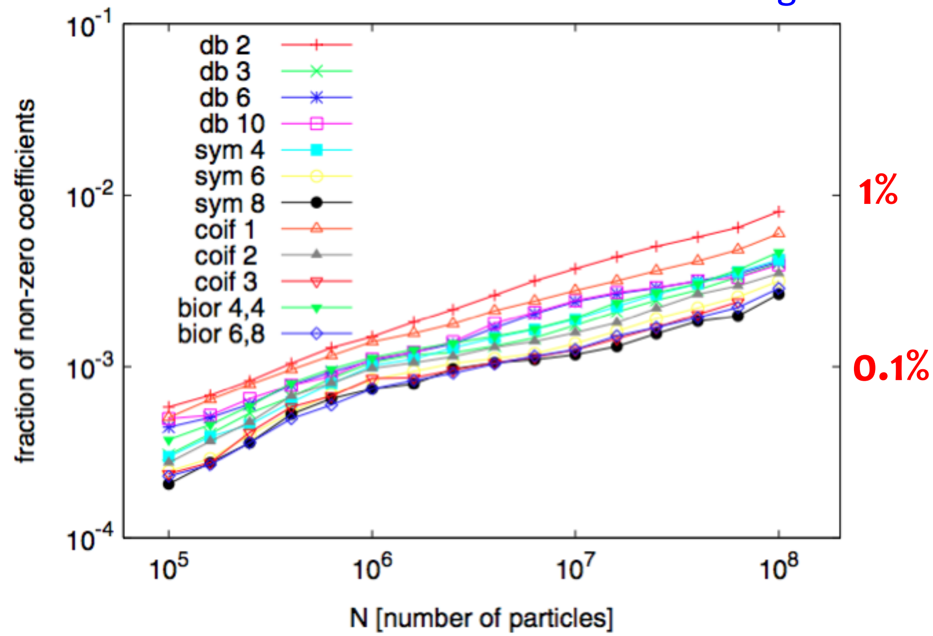
  [Terzić, Pogorelov & Bohn 2007, PR STAB 10, 034201]
  [Terzić & Bassi 2011, PR STAB 14, 070701]

Wavelet Compression

Modulated flat-top particle distribution



Fraction of non-zero coefficients retained after wavelet thresholding



[From Terzić & Bassi 2011, PR STAB 14, 070701]

CSR: Point-to-Point Approach

- **Point-to-Point approach (2D):** [Li 1998]

$$f(\vec{r},\vec{v},t) = q\sum_{i=1}^{N} n_m(\vec{r}-\vec{r}_0^{(i)}(t))\,\delta(\vec{v}-\vec{v}_0^{(i)}(t))$$ DF

$$\rho(\vec{r},t) = q\sum_{i=1}^{N} n_m(\vec{r}-\vec{r}_0^{(i)}(t))$$ Charge density

$$\vec{J}(\vec{r},t) = q\sum_{i=1}^{N} \vec{\beta}_0^{(i)}(t)\, n_m(\vec{r}-\vec{r}_0^{(i)}(t))$$ Current density

$$n_m(\vec{r}-\vec{r}_0^{(i)}(t)) = \frac{1}{2\pi\sigma_m^2}\exp\left[-\frac{(x-x_0(t))^2+(y-y_0(t))^2}{2\sigma_m^2}\right]$$ Gaussian macroparticle

- Charge density is sampled with *N* Gaussian-shaped 2D macroparticles (2D distribution without vertical spread)

- Each macroparticle interacts with each macroparticle throughout history

- **Expensive**: computation of retarded potentials and self fields ~ O(N^2)
  $\Rightarrow$ small number *N* $\Rightarrow$ *poor spatial resolution*
  $\Rightarrow$ difficult to see small-scale structure

- While useful in obtaining low-order moments of the beam,
  Point-to-Point approach is not optimal for studying CSR

CSR: Particle-In-Cell Approach

- **Particle-In-Cell approach with retarded potentials (2D):**

$$f(\vec{r},\vec{v},t) = q\sum_{i=1}^{N} \delta(\vec{r}-\vec{r}_0^{(i)}(t))\, \delta(\vec{v}-\vec{v}_0^{(i)}(t))$$

DF (Klimontovich)

$$\rho(\vec{x}_{\vec{k}},t) = q\sum_{i=1}^{N}\int_{-h}^{h} \delta(\vec{x}_{\vec{k}}-\vec{x}_0^{(i)}(t)+\vec{X})\, p(\vec{X})\, d\vec{X}$$

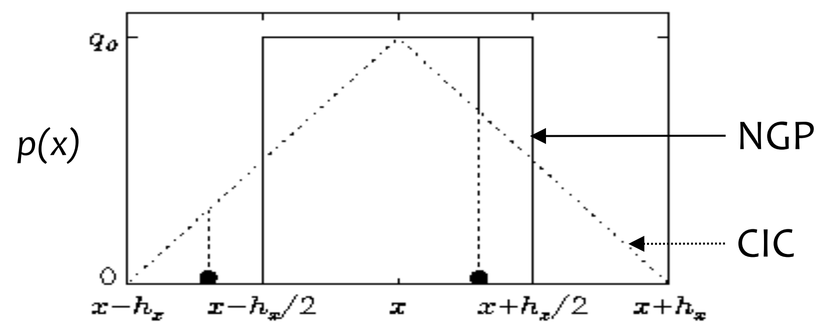
Charge density

$$\vec{J}(\vec{x}_{\vec{k}},t) = q\sum_{i=1}^{N}\vec{\beta}_0^{(i)}(t)\int_{-h}^{h} \delta(\vec{x}_{\vec{k}}-\vec{x}_0^{(i)}(t)+\vec{X})\, p(\vec{X})\, d\vec{X}$$

Current density

- Charge and current densities are sampled with *N* point-charges (δ-functions) and deposited on a finite grid $\vec{x}_{\vec{k}}$ using a deposition scheme $p(\vec{X})$
  - Two main deposition schemes
    - Nearest Grid Point (NGP)
      (constant: deposits to 1^D points)
    - Cloud-In-Cell (CIC)
      (linear: deposits to 2^D points)

    There exist higher-order schemes



x – macroparticle location
● – gridpoint location

- Particles *do not directly* interact with each other, *but only through a mean-field* of the gridded representation

CSR: P2P Vs. PIC

- Computational cost for P2P: Total cost ~ O(N^2)
  - Integration over history (yields self-forces): O(N^2) operation
- Computational cost for PIC: Total cost ~ O(N_{grid}^2)
  - Particle deposition (yields gridded charge & current densities): O(N) operation
  - Integration over history (yields retarded potentials): O(N_{grid}^2) operation
  - Finite difference (yields self-forces on the grid): O(N_{grid}) operation
  - Interpolation (yields self-forces acting on each of N particles): O(N) operation
  - Overall ~ O(N_{grid}^2)+O(N) operations
    - But in realistic simulations: $N_{grid}^2 >> N$, so the total cost is ~ O(N_{grid}^2)
  - Favorable scaling allows for larger N, and reasonable grid resolution

    $\Rightarrow$ Improved spatial resolution
- <u>Fair comparison</u>: P2P with N macroparticles and PIC with N_{grid}=N

CSR: P2P Vs. PIC

- Difference in spatial resolution: An illustrative example
  - Analytical distribution sampled with
    - $N = N_X N_Y$ macroparticles (as in P2P)
    - On a $N_X \times N_Y$ grid (as in PIC)
  - 2D grid: $N_X = N_Y = 32$

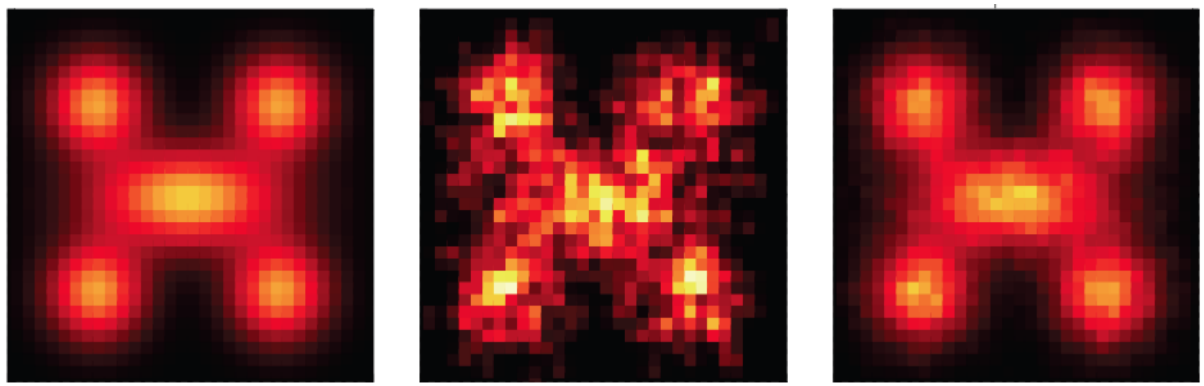
Signal-to-Noise Ratio

$$SNR = \sqrt{\frac{\sum_{i=1}^{N_{grid}} \overline{q}_i^2}{\sum_{i=1}^{N_{grid}} \left(q_i - \overline{q}_i\right)^2}}$$

$\overline{q}_i$ *exact*

q_i *grid*

EXACT | P2P N=32^2 SNR=2.53 | PIC N=50×32^2 SNR=13.89



- PIC approach provides superior spatial resolution to P2P approach
- This motivates us to use a PIC code

Outline of the P2P Algorithm