PAUL SCHERRER INSTITUT

PSI

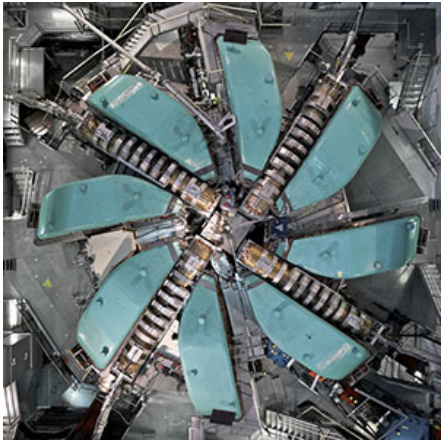
Matthias Frey :: PhD student LSM-AMAS :: Paul Scherrer Institut

Adaptive mesh refinement Poisson solver for neighbouring bunch simulations

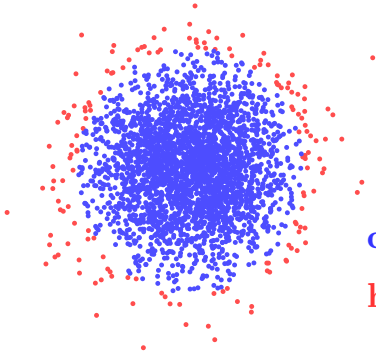
11/21/2019 :: FFA'19

Thesis advisor: Prof. Dr. Klaus S. Kirch
Thesis supervisor: Dr. Andreas Adelmann

Motivation - Precise Simulations of Multi Bunches in High Intensity Cyclotrons



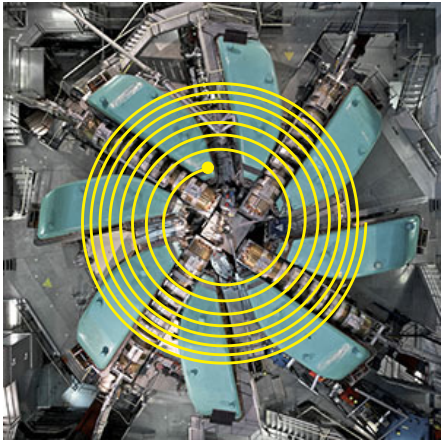
PSI Ring Cyclotron
(`https://www.psi.ch/ltp/facilities`)



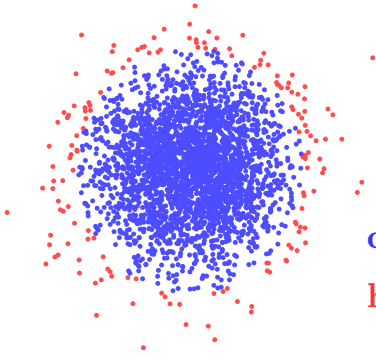
core $(< 3\sigma)$

halo $(> 3\sigma)$

Motivation - Precise Simulations of Multi Bunches in High Intensity Cyclotrons



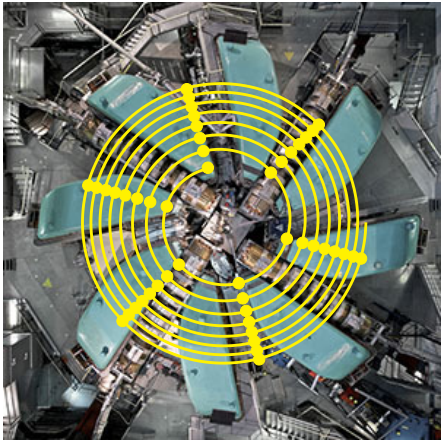
PSI Ring Cyclotron
(`https://www.psi.ch/ltp/facilities`)



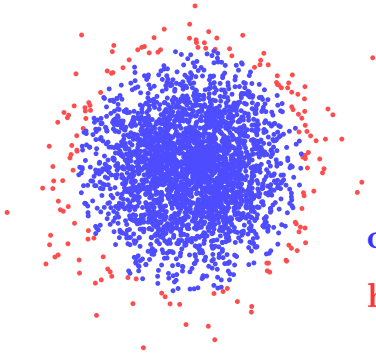
core $(< 3\sigma)$

halo $(> 3\sigma)$

Motivation - Precise Simulations of Multi Bunches in High Intensity Cyclotrons



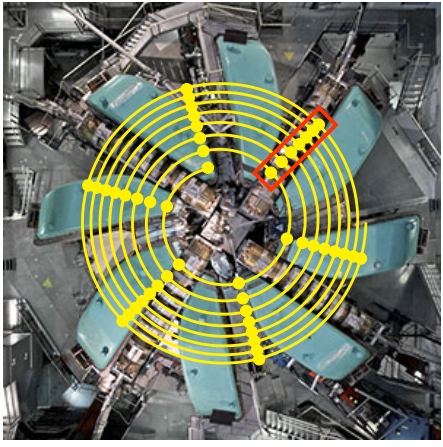
PSI Ring Cyclotron
(`https://www.psi.ch/ltp/facilities`)



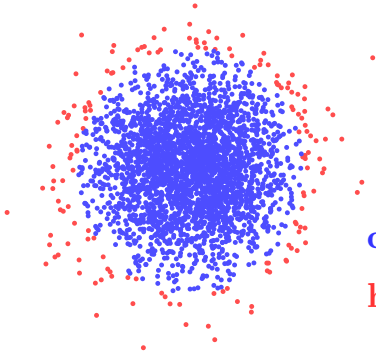
core $(< 3\sigma)$

halo $(> 3\sigma)$

Motivation - Precise Simulations of Multi Bunches in High Intensity Cyclotrons



PSI Ring Cyclotron
(`https://www.psi.ch/ltp/facilities`)

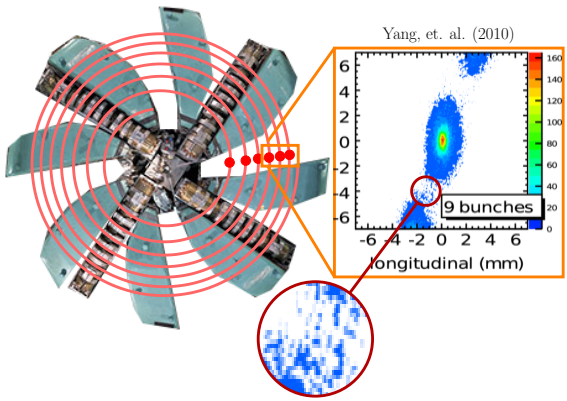


core $(< 3\sigma)$

halo $(> 3\sigma)$

Goal: Understand Halo Creation and Evolution

- Halo particles become losses
  - $\Rightarrow$ machine activation
  - $\Rightarrow$ machine intensity limitation
- Large scale N-body problems of $\mathcal{O}(\mathbf{10^9 ... 10^{10}})$ **particles** coupled with Maxwell's equations
- Particle-In-Cell (PIC) models with $\mathcal{O}(\mathbf{10^8 ... 10^9})$ **grid points**



Characterization of Halo

- **Spatial-halo parameter**, $i = x, y, z$

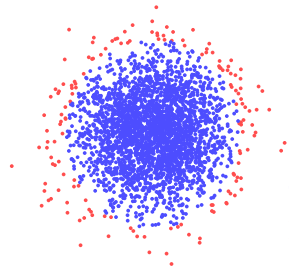
$$h_i = \frac{\left\langle q_i^4 \right\rangle}{\left\langle q_i^2 \right\rangle^2} - \frac{15}{7}$$

- **Phase-space halo parameter**, $i = x, y, z$

$$H_i = \frac{\sqrt{3}}{2} \frac{\sqrt{I_4^i}}{I_2^i} - \frac{15}{7}$$

$$I_2^i = \left\langle q_i^2 \right\rangle \left\langle p_i^2 \right\rangle - \left\langle q_i p_i \right\rangle^2$$

$$I_4^i = \left\langle q_i^4 \right\rangle \left\langle p_i^4 \right\rangle + 3 \left\langle q_i^2 p_i^2 \right\rangle^2 - 4 \left\langle q_i p_i^3 \right\rangle \left\langle q_i^3 p_i \right\rangle$$

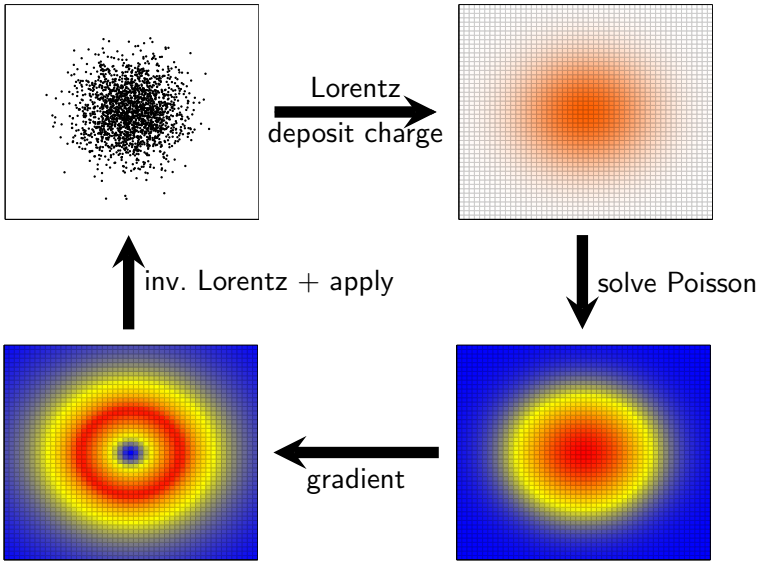


core ($< 3\sigma$)

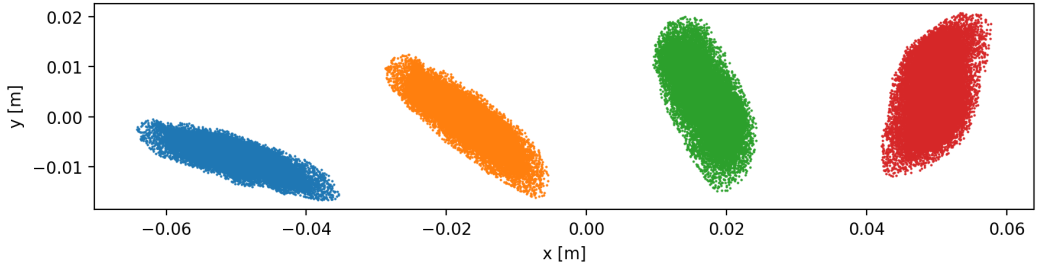
halo ($> 3\sigma$)

C. K. Allen and T. P. Wangler, Phys. Rev. ST Accel. Beams, Vol. 5, Issue 12 (2002)

Electrostatic PIC in OPAL



Electrostatic PIC with Neighbouring Bunches in OPAL



- Energy bin k of particle evaluated from momentum $\beta\gamma$:

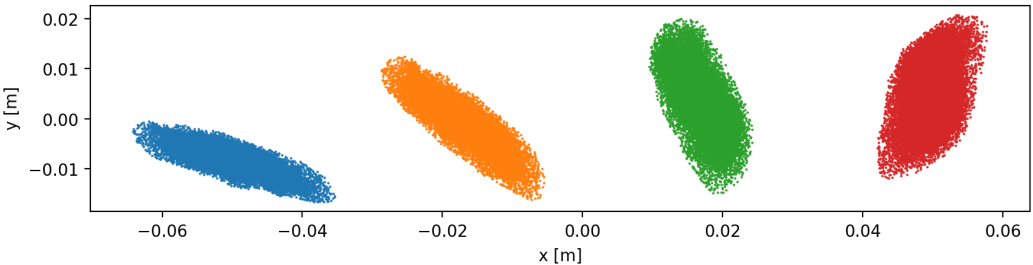
$$k = \left\lfloor \frac{\sinh^{-1}\left(\beta\gamma\right) - \sinh^{-1}\left(\min_{i=\{1,N\}}\left(\beta\gamma\right)_i\right)}{\eta} \right\rfloor$$

derived from: $\Delta(\beta\gamma) = (\beta\gamma)_{k+1} - (\beta\gamma)_k \approx \eta\gamma_k = \eta\sqrt{1+(\beta\gamma)_k^2} \xrightarrow{\eta \ll 1} \frac{d(\beta\gamma)(k)}{dk} \approx \eta\sqrt{1+(\beta\gamma)_k^2}$

$\rightarrow (\beta\gamma)_k = \sinh\left(k\eta + \sinh^{-1}\left[\min_{i=\{1,N\}}(\beta\gamma)_i\right]\right)$

G. Fubiani, et al., Phys. Rev. ST Accel. Beams, Vol. 9, Issue 6 (2006)

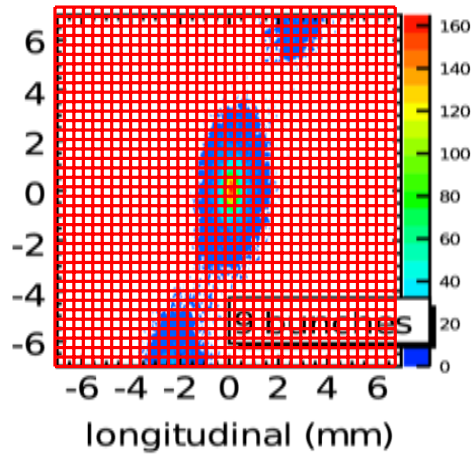
Electrostatic PIC with Neighbouring Bunches in OPAL



- Space-Charge calculation for multi-bunch mode:
  - energy-dependent particle binning
  - compute space-charge in every energy bin
  - sum up contribution of all energy bins

Towards Exascale Computing

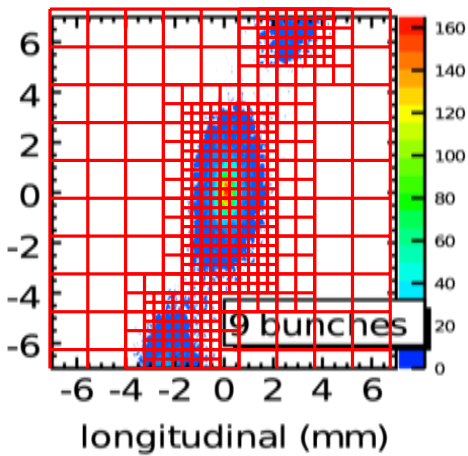
- **Naive Meshing:** Cartesian Uniform
  - Waste of memory in regions of void
  - Waste of computational power



Towards Exascale Computing

- **Naive Meshing:** Cartesian Uniform
  - Waste of memory in regions of void
  - Waste of computational power
- **Adaptive Meshing:**
  - Save memory
  - Save computational effort
  - Used in CFD, astrophysics, etc.
- **Issue of state-of-the-art solvers:**

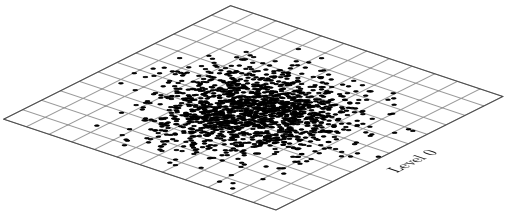
  Implementation is hardware specific (e.g. CPU, GPU)



AMReX in a Nutshell

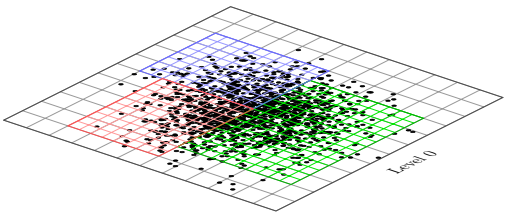
Tagging - Mark a Cell for Refinement (https://amrex-codes.github.io/)

- Coordinate space discretized by grid



AMReX in a Nutshell

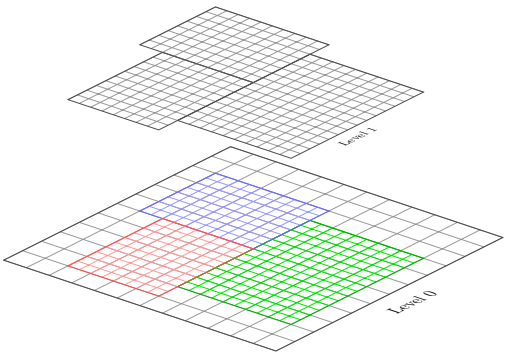
Tagging - Mark a Cell for Refinement (https://amrex-codes.github.io/)



- Coordinate space discretized by grid
- Mark cell for refinement according to some criteria:
  - charge density per cell
  - potential gradient
  - potential magnitude
  - etc.

AMReX in a Nutshell

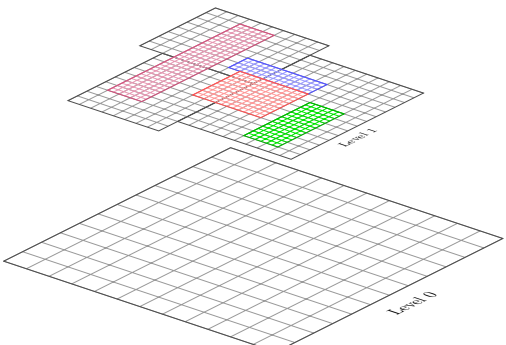
Tagging - Mark a Cell for Refinement (https://amrex-codes.github.io/)



- Generate level 1

AMReX in a Nutshell

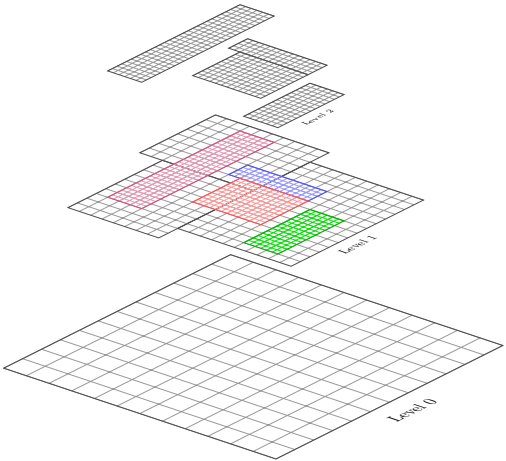
Tagging - Mark a Cell for Refinement (https://amrex-codes.github.io/)



- Generate level 1
- Mark cell for refinement according to some criteria:
  - charge density per cell
  - potential gradient
  - potential magnitude
  - etc.

AMReX in a Nutshell

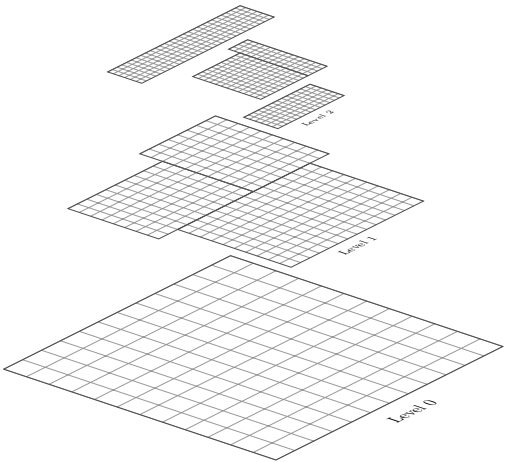
Tagging - Mark a Cell for Refinement (https://amrex-codes.github.io/)



- Generate level 2

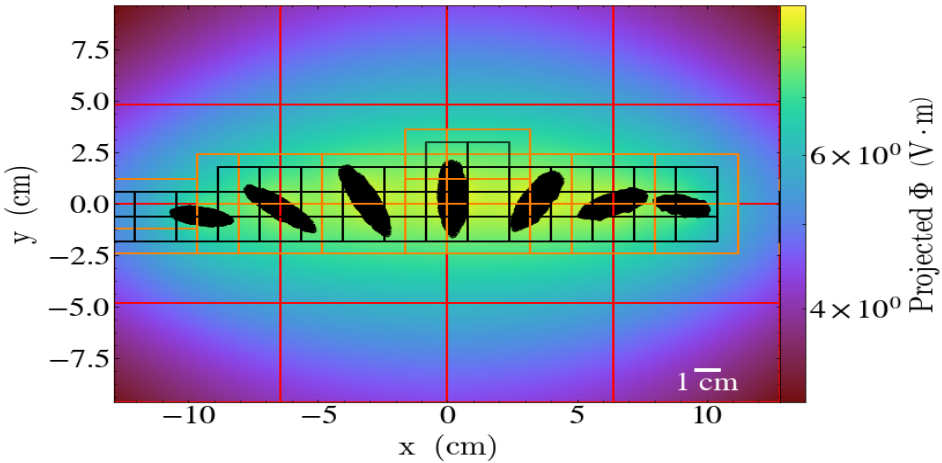
AMReX in a Nutshell

Tagging - Mark a Cell for Refinement (https://amrex-codes.github.io/)

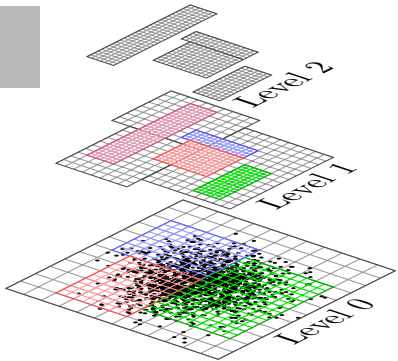


- Generate level 2
- Maximum level reached (user-defined)

Motivation - Adaptive Mesh Refinement (AMR)



How to solve Poisson's Equation with AMR?



Dan F. Martin and Keith L. Cartwright. Solving poisson's equation using adaptive mesh refinement. Technical Report UCB/ERL M96/66, Univ. Calif. Berkeley, 1996.

- **Poisson's equation:**

$$\Delta\phi(x,y,z) = -\frac{\rho}{\varepsilon_0}$$
$$\phi(\infty) = 0$$

  with charge density ρ, vacuum permittivity ε_0 and potential ϕ.

- **Difficulty:** Continuity conservation of ϕ!

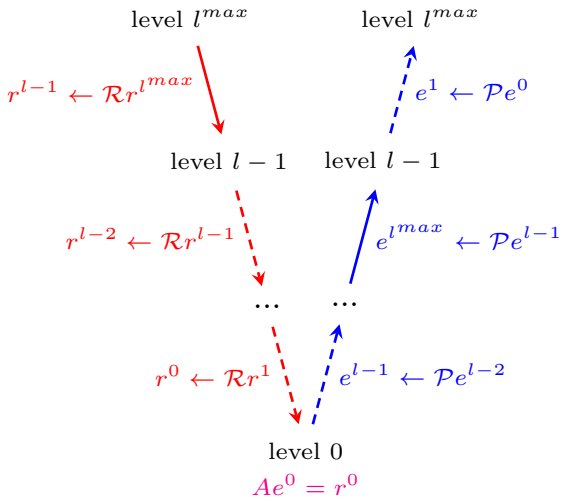
- **Solution:** elliptic matching condition, i.e.

  Dirichlet + Neumann condition at coarse-fine interfaces

AMR Poisson Solver

V-cycle (residual correction formulation) Algorithm

1: **function** RELAX(l)
2: **if** $l = l^{max}$ **then**
3: $r^l \leftarrow \rho^l - \mathcal{L}\phi^l$ ($\mathcal{L}$: Laplace operator)
4: **end if**
5: **if** $l > 0$ **then**
6: $\phi^{l,save} \leftarrow \phi^l$
7: $e^{l-1} \leftarrow 0$
8: SMOOTH(e^l, r^l)
9: $\phi^l \leftarrow \phi^l + e^l$
10: $r^{l-1} \leftarrow \mathcal{R}r^l$ ▷ restrict residual
11: RELAX($l-1$)
12: $e^l \leftarrow \mathcal{P}e^{l-1}$ ▷ prolongate error
13: $r^l \leftarrow r^l - \mathcal{L}e^l$
14: $\delta e^l \leftarrow 0$
15: SMOOTH($\delta e^l, r^l$)
16: $e^l \leftarrow e^l + \delta e^l$
17: $\phi^l \leftarrow \phi^{l,save} + e^l$
18: **else**
19: solve $Ae^0 = r^0$
20: $\phi^0 \leftarrow \phi^0 + e^0$
21: **end if**
22: **end function**



Implementation Overview

- Implemented fully in Trilinos with **2nd generation packages**, i.e.
  - Tpetra (matrix / vector data structure)
  - Ifpack2 (smoothers e.g. Gauss-Seidel, Jacobi)
  - MueLu, Amesos2, Belos (linear solvers)

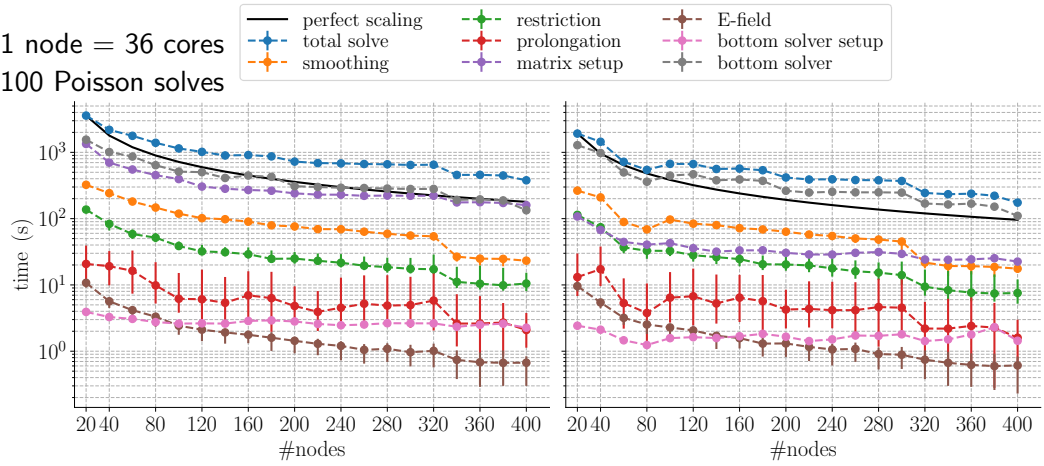
Implementation Overview

- Implemented fully in Trilinos with **2nd generation packages**, i.e.
  - Tpetra (matrix / vector data structure) → **Kokkos**
  - Ifpack2 (smoothers e.g. Gauss-Seidel, Jacobi)
  - MueLu, Amesos2, Belos (linear solvers)
- Kokkos allows **portable code** between hardware architectures **without changing your code!**
  - GPU
  - OpenMP / PThreads / serial

AMR Poisson Solver Strong Scaling

(speedup) $S_p = T_1/T_p \geq 0$ (T_p = time with p processes/cores)

1 node = 36 cores
100 Poisson solves



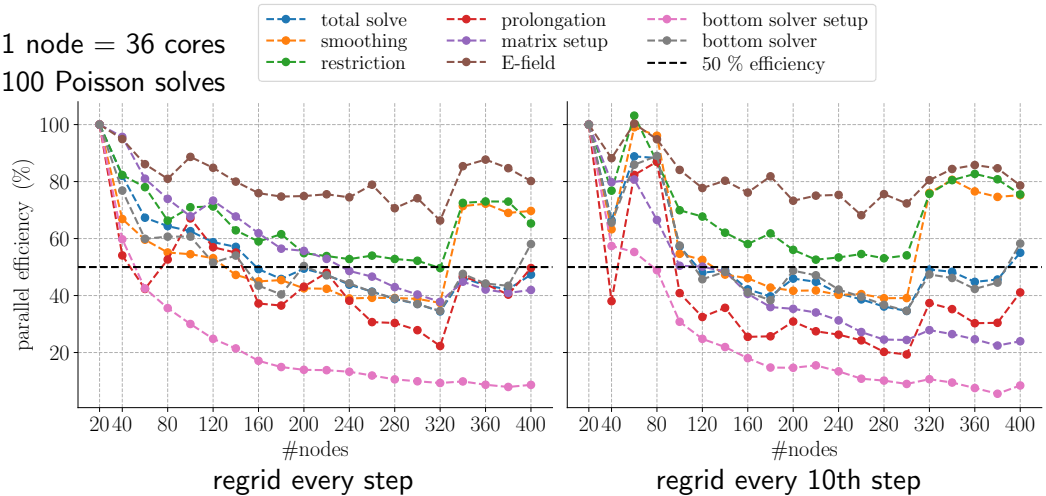
regrid every step

regrid every 10th step

AMR Poisson Solver Parallel Efficiency

(efficiency) $E_p = S_p/p \leq 1$

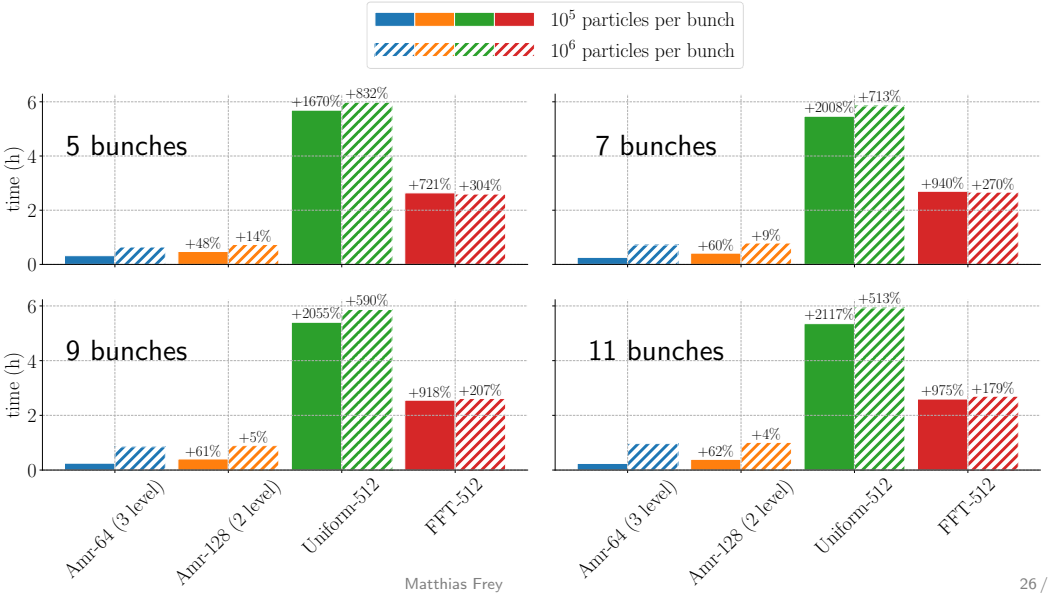
1 node = 36 cores
100 Poisson solves



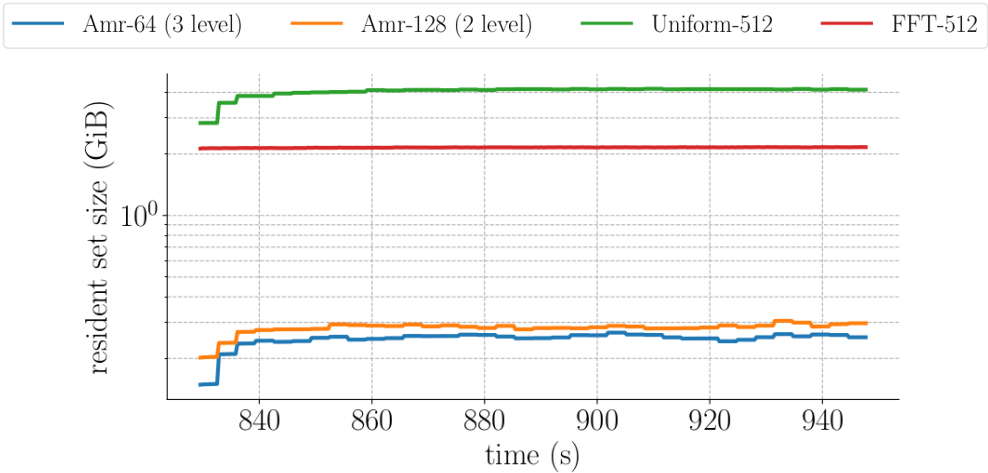
regrid every step

regrid every 10th step

Timings of AMR and FFT Particle-In-Cell Models



Memory Usage of AMR and FFT Particle-In-Cell Models



Summary

- Adaptive mesh refinement as a new OPAL PIC model for neighbouring bunch simulations
- Benefits w.r.t. neighbouring bunch simulations at same resolution:
  - less computational intensive
  - less memory intensive
- Hardware-architecture independent AMR Poisson solver
- Please see also
  M. Frey, et al., *On architecture and performance of adaptive mesh refinement in an electrostatics Particle-In-Cell code*, Computer Physics Communications, 2019, `https://doi.org/10.1016/j.cpc.2019.106912`

Thanks to

P. Arbenz (ETHZ)

A. Almgren (LBNL)

A. Myers (LBNL)

W. Zhang (LBNL)

D. Martin (LBNL)

C. Siefert (Sandia)

K. Devine (Sandia)