M. Frey, A. Adelmann :: SNF project 200021_159936 :: Paul Scherrer Institut

Computer Architecture Independent Adaptive Geometric Multigrid Solver for AMR-PIC

21/10/2018 :: ICAP'18

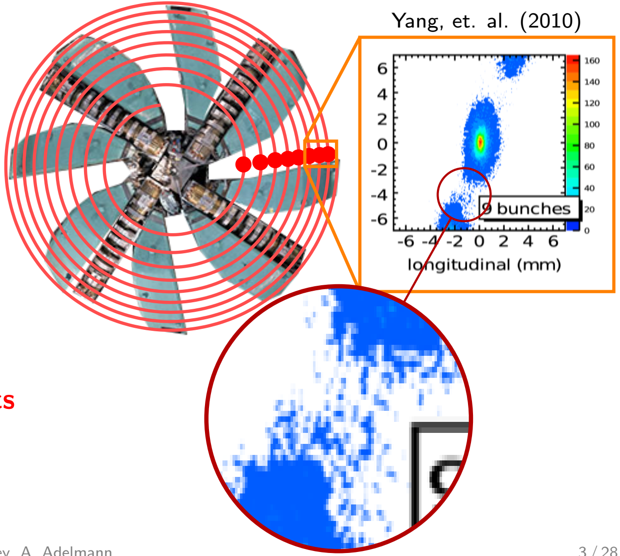
Thesis advisor: Prof. Dr. Klaus S. Kirch
Thesis supervisor: Dr. Andreas Adelmann

Outline

- Motivation
- Adaptive Geometric Multigrid
- Benchmarks
- Conclusion

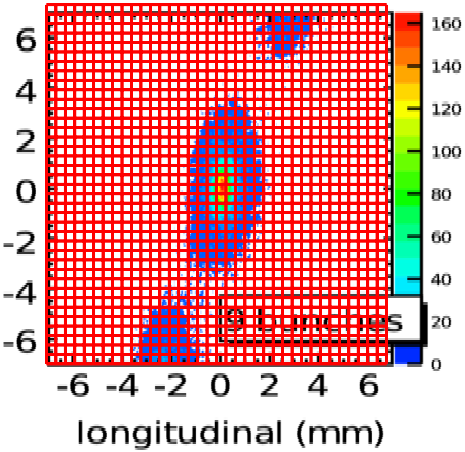
Goal: Understand Halo Creation and Evolution

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



Towards Exascale Computing

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

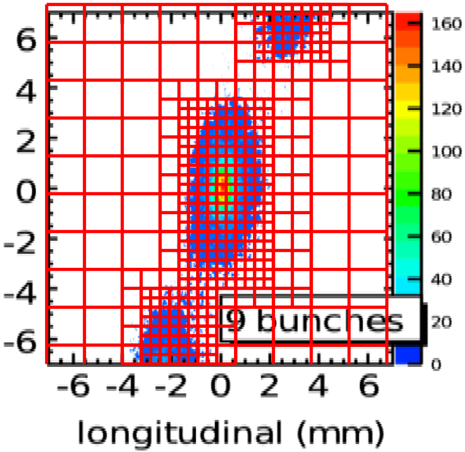


Towards Exascale Computing

- **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)



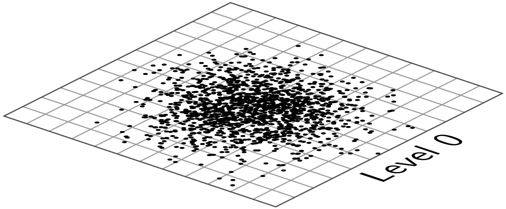
Tools: Software & Hardware

- **OPAL:** physics (https://gitlab.psi.ch/OPAL/src/wikis/home)
- **AMReX:** grids, formerly: BoxLib (https://ccse.lbl.gov/AMReX)
- **Trilinos:** Amesos2, Belos, Ifpack2, MueLu, Tpetra (https://trilinos.org/)

Piz Daint at CSCS

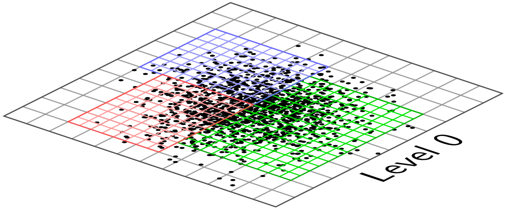
Taken from https://www.cscs.ch/computers/piz-daint/

AMReX in a Nutshell



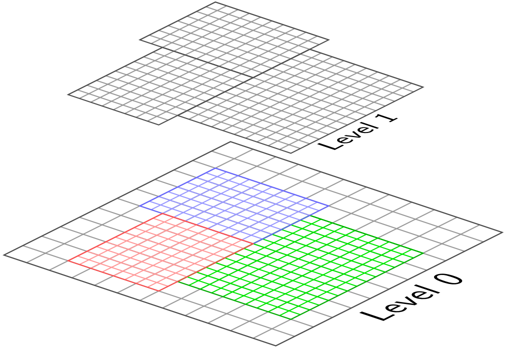
- Coordinate space discretized by grid

AMReX in a Nutshell



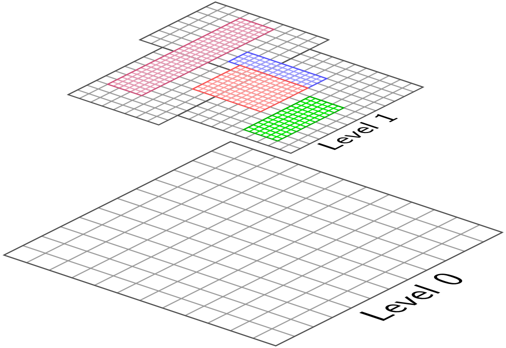
- Coordinate space discretized by grid
- Mark cell for refinement according to some criteria:
  - charge density per cell
  - potential gradient
  - potential magnitude
  - etc.

AMReX in a Nutshell



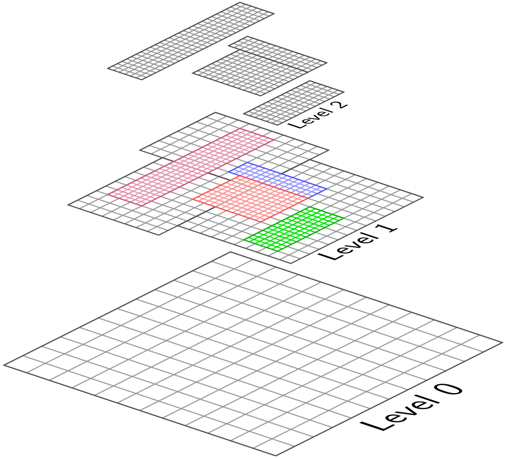
- Generate level 1

AMReX in a Nutshell



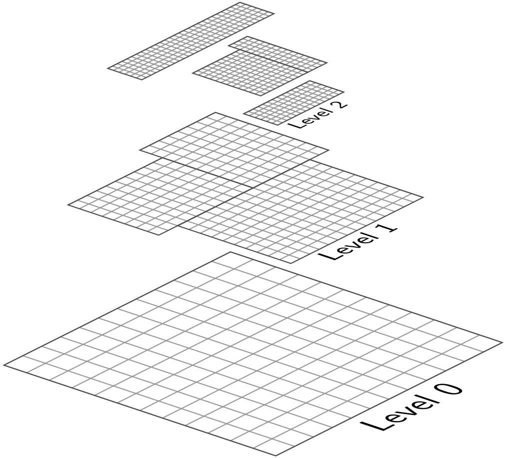
- Generate level 1
- Mark cell for refinement according to some criteria:
  - charge density per cell
  - potential gradient
  - potential magnitude
  - etc.

AMReX in a Nutshell



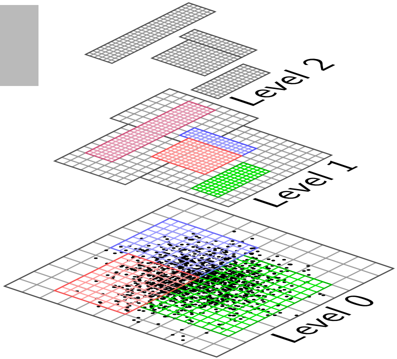
- Generate level 2

AMReX in a Nutshell



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

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.

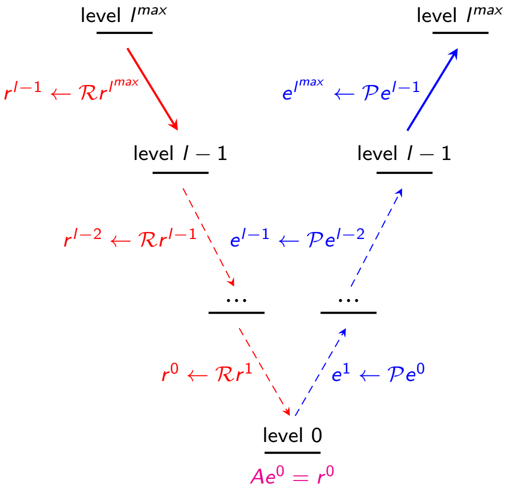
- **Difficulty:** Continuity conservation of ϕ!
- **Solution:** elliptic matching condition, i.e.

  Dirichlet + Neumann condition at coarse-fine interfaces

AMR Multigrid Poisson Solver

V-cycle (residual correction formulation) Algorithm

function RELAX(l)
  if $l = l^{max}$ **then**
    $r^l \leftarrow \rho^l - \mathcal{L}\phi^l$ ($\mathcal{L}$: Laplace operator)
  end if
  if $l > 0$ **then**
    $\phi^{l,save} \leftarrow \phi^l$
    $e^{l-1} \leftarrow 0$
    SMOOTH(e^l, r^l)
    $\phi^l \leftarrow \phi^l + e^l$
    $r^{l-1} \leftarrow \mathcal{R}r^l$ ▷ restrict residual
    RELAX($l-1$)
    $e^l \leftarrow \mathcal{P}e^{l-1}$ ▷ prolongate error
    $r^l \leftarrow r^l - \mathcal{L}e^l$
    $\delta e^l \leftarrow 0$
    SMOOTH($\delta e^l, r^l$)
    $e^l \leftarrow e^l + \delta e^l$
    $\phi^l \leftarrow \phi^{l,save} + e^l$
  else
    solve $Ae^0 = r^0$
    $\phi^0 \leftarrow \phi^0 + e^0$
  end if
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)

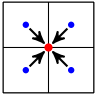
Implementation Overview

- Implemented fully in Trilinos with **2nd generation packages**, i.e.
  - Tpetra (matrix / vector data structure) $\rightarrow$ **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

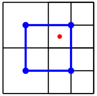
Data and Operations per Level

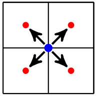
- **8 CRS-matrices**
  (prolongation, restriction, (composite) Poisson, boundaries, ...)
- **4 vectors**
  (RHS, LHS, residual, error)
- **Restriction:** simple averaging ($N = 4$ in 2D, $N = 8$ in 3D)

$$\phi^c = \frac{1}{N}\sum_{i=0}^{N-1}\phi_i^f,$$



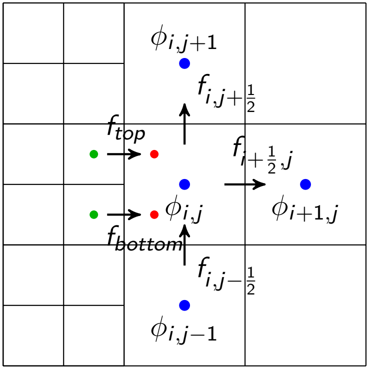
- **Prolongation:** Trilinear interp. or piecewise const. interp





Coarse-Fine-Interface

Flux Matching Condition in 2D $\rightarrow$ 1D interface normal to boundary



$$(\Delta\phi)_{i,j} = \frac{f_{i+\frac{1}{2},j} - f^{ave}_{i-\frac{1}{2},j} + f_{i,j+\frac{1}{2}} - f_{i,j-\frac{1}{2}}}{h_c}$$

$$f^{ave}_{i-\frac{1}{2},j} = \frac{1}{2}\left(f_{top} + f_{bottom}\right),$$

$$f_{i+\frac{1}{2},j} = \frac{1}{h_c}\left(\phi_{i+1,j} - \phi_{i,j}\right),$$

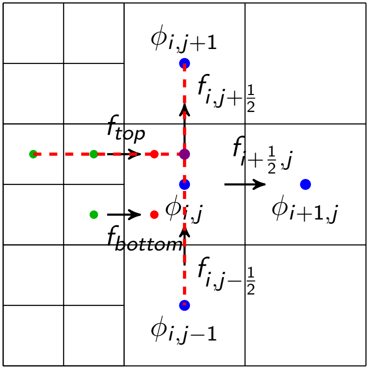
$$f_{i,j+\frac{1}{2}} = \frac{1}{h_c}\left(\phi_{i,j+1} - \phi_{i,j}\right),$$

$$f_{i,j-\frac{1}{2}} = \frac{1}{h_c}\left(\phi_{i,j} - \phi_{i,j-1}\right)$$

$$f_{top/bottom} = \frac{1}{h_f}\left(\phi^{int}_{top/bottom} - \phi_{top/bottom}\right)$$

Coarse-Fine-Interface

Flux Matching Condition in 2D $\rightarrow$ 1D interface normal to boundary



$$(\Delta\phi)_{i,j} = \frac{f_{i+\frac{1}{2},j} - f^{ave}_{i-\frac{1}{2},j} + f_{i,j+\frac{1}{2}} - f_{i,j-\frac{1}{2}}}{h_c}$$

$$f^{ave}_{i-\frac{1}{2},j} = \frac{1}{2}\left(f_{top} + f_{bottom}\right),$$

$$f_{i+\frac{1}{2},j} = \frac{1}{h_c}\left(\phi_{i+1,j} - \phi_{i,j}\right),$$

$$f_{i,j+\frac{1}{2}} = \frac{1}{h_c}\left(\phi_{i,j+1} - \phi_{i,j}\right),$$

$$f_{i,j-\frac{1}{2}} = \frac{1}{h_c}\left(\phi_{i,j} - \phi_{i,j-1}\right)$$

$$f_{top/bottom} = \frac{1}{h_f}\left(\phi^{int}_{top/bottom} - \phi_{top/bottom}\right)$$

$\Longrightarrow$ **Lagrange interpolation**

2nd Order Lagrange Interpolation in 2D

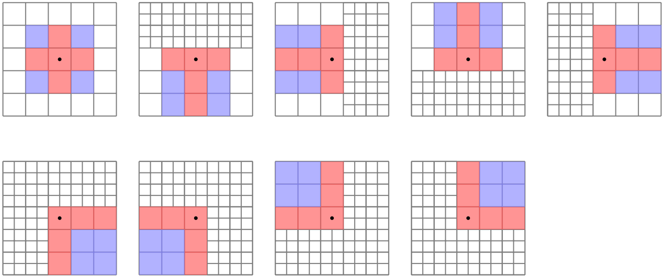
In 3D $\rightarrow$ 2D interface normal to boundary

$$g(u,v) = \sum_{i,j=0}^{2} f(u_i, v_j) L_i(u) L_j(v)$$

with

$$L_i(x) = \frac{(x - x_k)(x - x_l)}{(x_i - x_k)(x_i - x_l)} \qquad (l \neq i \neq k \neq l)$$

Cells need to be uncovered! $\Longrightarrow$ 9 possible configurations



2nd Order Lagrange Interpolation

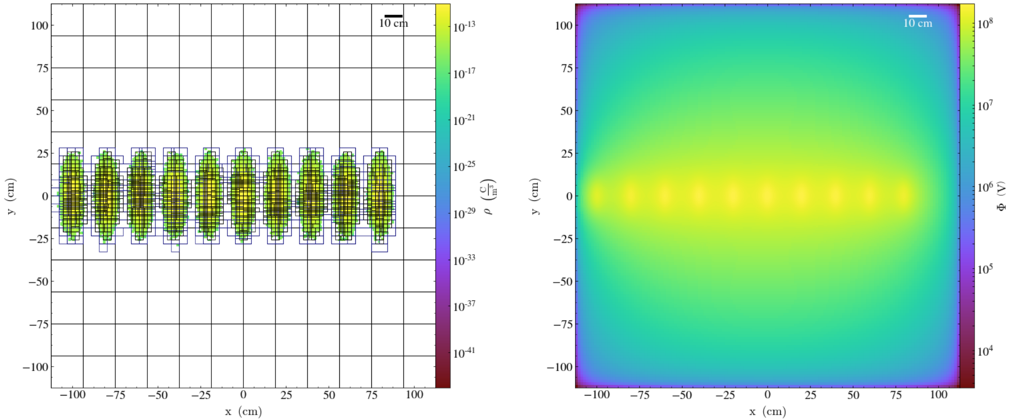
Bit pattern

20	21	22	23	24
15	16	17	18	19
10	11	12	13	14
5	6	7	8	9
0	1	2	3	4

20	21	22	23	24
15	16	17	18	19
10	11	12	13	14
5	6	7	8	9
0	1	2	3	4

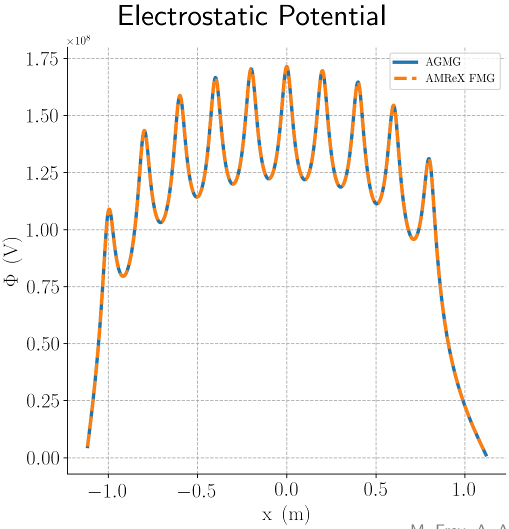
bit pattern	unsigned long	case
0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0	473′536	0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0	14′798	1
0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0	236′768	2
0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0	15′153′152	3
0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0	947′072	4
0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0	29′596	5
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1	7′399	6
0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0	7′576′576	7
1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0	30′306′304	8

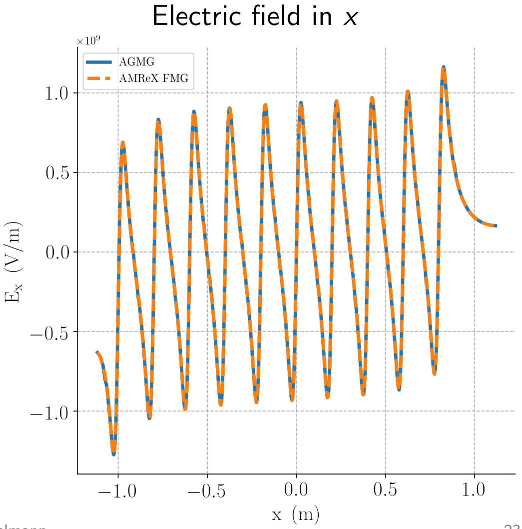
Benchmark - Projection-Plots (axis z)



M. J. Turk, B. D. Smith, J. S. Oishi, S. Skory, S. W. Skillman, T. Abel, M. L. Norman, yt: A Multi-code Analysis Toolkit for Astrophysical Simulation Data, Astrophysical Journal, Supplement 192 (2011) 9.

Benchmark - Lineplots AMReX FMG vs. AGMG





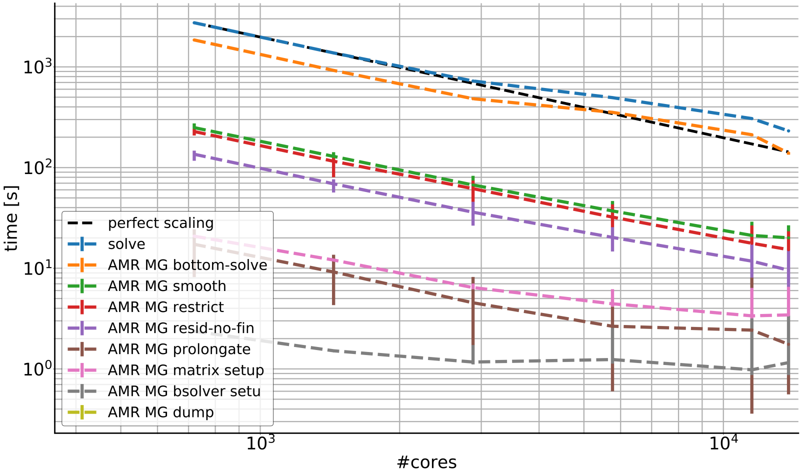
Benchmark

- 10 bunches (each $\approx 3 \cdot 10^8$ particles)
- base grid: 576^3
- max. grid: 24

  ($\rightarrow$ max. 13'824 cores on base level)
- #level of refinement: 2
- 100 solves
  (move each particle randomly within $[-0.001, 0.001]$ after every solve)

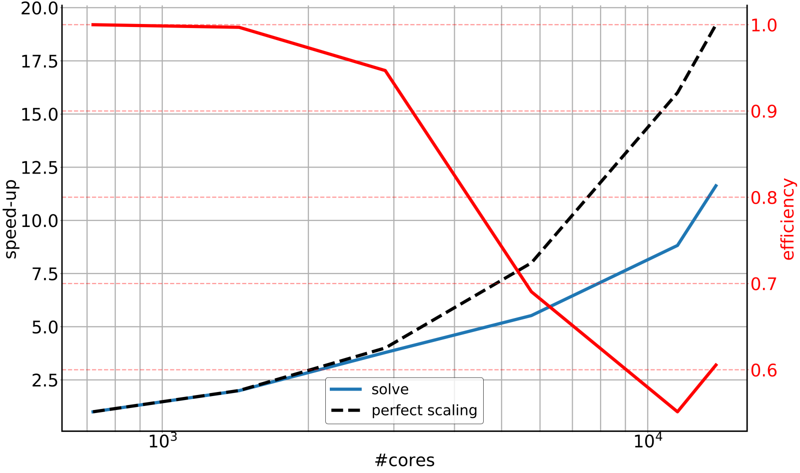
Strong Scaling

720 to 13'824 cores (no hyperthreading) on Piz Daint



Parallel Efficiency

720 to 13'824 cores (no hyperthreading) on Piz Daint



Conclusion

Solver ...

- works and gives **good scalability** (CPUs)
- **runs on GPUs** (EuroHack18: GPU Programming Hackathon)
- is hardware **architecture independent**
- is part of open-source beam dynamics code **OPAL**
- uses **structured aggregation** for bottom level linear system of equations solve (Sandia visit: 15th - 19th Oct. 2018)

Thanks to

A. Almgren (LBNL)

P. Arbenz (ETH)

A. Myers (LBNL)

W. Zhang (LBNL)

D. F. Martin (LBNL)

K. D. Devine (Sandia)

C. Siefert (Sandia)

L. Berger-Vergiat (Sandia)