OPAL: From Today to Tomorrow

Andreas Adelmann for the OPAL developer team

August 28, 2023

Content

- General Introduction
  - OPAL in a nutshell
  - Release roadmap
  - Flavours in OPAL
  - OPAL architecture
  - Future plans
- OPAL-X
  - Why are we interested in exascale

Object Oriented Parallel Particle Library (OPAL)

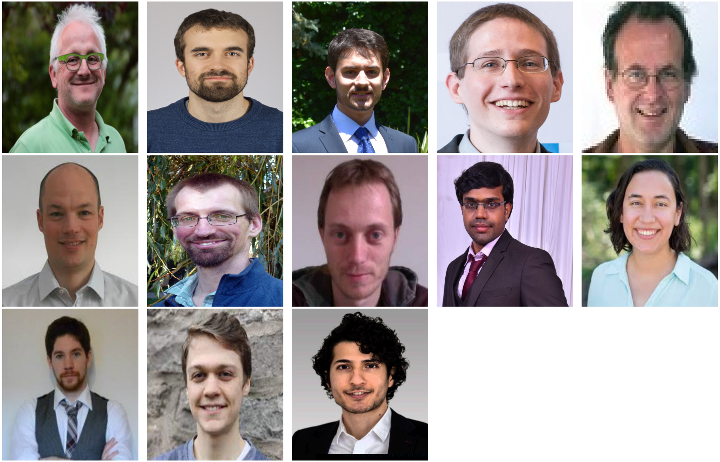
https://gitlab.psi.ch/OPAL/src/wikis/home

OPAL is a versatile open-source tool for charged-particle optics in large accelerator structures and beam lines including 3D EM field calculation, collisions, radiation, particle-matter interaction, and multi-objective optimisation

- OPAL is built from the ground up as an HPC application
- OPAL runs on your laptop as well as on the largest HPC clusters
- OPAL uses the MAD language with extensions
- OPAL is written in C++, uses design patterns, easy to extend
- The OPAL Discussion Forum: https://psilists.ethz.ch/sympa/info/opal
- International team of 13 active developers and a user base of $\mathcal{O}(100)$
- The OPAL **sampler** command can generate labeled data sets using the largest computing resources and allocations available

PAUL SCHERRER INSTITUT PSI

The Active OPAL Developer Team



PAUL SCHERRER INSTITUT PSI

Ciemat Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas

SLAC NATIONAL ACCELERATOR LABORATORY

UKRI Science and Technology Facilities Council

MIT Massachusetts Institute of Technology

JÜLICH

University of St Andrews

NIU Northern Illinois University

PAUL SCHERRER INSTITUT PSI

Two OPAL flavours, OPAL-T & OPAL-CYCL

- Common features
  - 3D space charge: in unbounded, and bounded domains
  - particle Matter Interaction (protons)
  - parallel hdf5 & SDDS output
  - sampler & multi-objective optimisation
  - from e, p to Uranium (q/m is a parameter)
- OPAL-CYCL (+ FFAs + Synchrotrons)
  - neighbouring turns
  - time integration, 4th-order RK, LF, adaptive schemes
  - find matched distributions with linear space charge
  - spiral inflector modelling with space charge
- OPAL-T
  - rf-guns, injectors, beamlines [1]
  - auto-phasing (with veto)
  - full EM in `undulator` element since OPAL 2021.1
  - Particle-Particle-Particle-Mesh solver since OPAL 2022.1

[1] Proton therapy gantries & degrader

Two OPAL flavours, OPAL-T & OPAL-CYCL

- Common features
  - 3D space charge: in unbounded, and bounded domains
  - particle Matter Interaction (protons)
  - parallel hdf5 & SDDS output
  - sampler & multi-objective optimisation
  - from e, p to Uranium (q/m is a parameter)
- OPAL-CYCL (+ FFAs + Synchrotrons)
  - neighbouring turns
  - time integration, 4th-order RK, LF, adaptive schemes
  - find matched distributions with linear space charge
  - spiral inflector modelling with space charge
- OPAL-T
  - rf-guns, injectors, beamlines [1]
  - auto-phasing (with veto)
  - full EM in `undulator` element since OPAL 2021.1
  - Particle-Particle-Particle-Mesh solver since OPAL 2022.1

[1] Proton therapy gantries & degrader

PAUL SCHERRER INSTITUT

Two OPAL flavours, OPAL-T & OPAL-CYCL

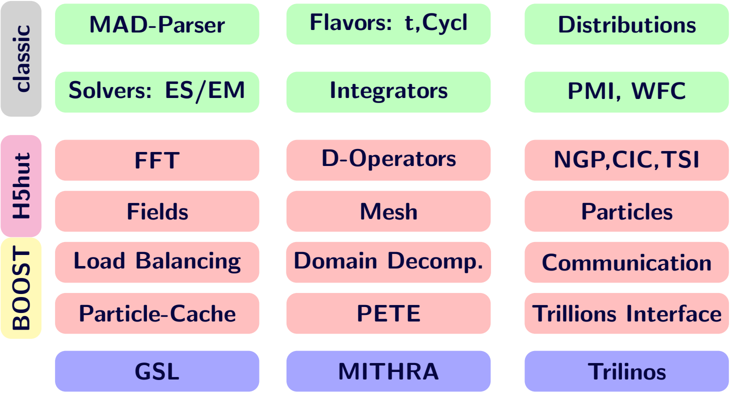
- Common features
  - 3D space charge: in unbounded, and bounded domains
  - particle Matter Interaction (protons)
  - parallel hdf5 & SDDS output
  - sampler & multi-objective optimisation
  - from e, p to Uranium (q/m is a parameter)
- OPAL-CYCL (+ FFAs + Synchrotrons)
  - neighbouring turns
  - time integration, 4th-order RK, LF, adaptive schemes
  - find matched distributions with linear space charge
  - spiral inflector modelling with space charge
- OPAL-T
  - rf-guns, injectors, beamlines [1]
  - auto-phasing (with veto)
  - full EM in `undulator` element since OPAL 2021.1
  - Particle-Particle-Particle-Mesh solver since OPAL 2022.1

[1] Proton therapy gantries & degrader

Software Architecture

MPI based

OPAL



Collision-less (non relativistic) Vlasov-Maxwell equation

$f_s \subset (\mathbb{R}^3 \times \mathbb{R}^3), \mathbb{R} :\rightarrow \mathbb{R}$ and s are the species.

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \nabla_x f_s + \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_v f_s = 0,$$

$$\left.\begin{array}{ll} \partial_t \mathbf{E} - c^2 \mathbf{curl}\,\mathbf{B} = \dfrac{\mathbf{J}}{\varepsilon_0}, & \nabla \cdot \mathbf{E} = \dfrac{\rho}{\varepsilon_0}, \\ \partial_t \mathbf{B} + \mathbf{curl}\,\mathbf{E} = 0, & \nabla \cdot \mathbf{B} = 0, \end{array}\right\} \quad \text{Maxwell's equations}$$

where the source terms are computed by

$$\rho = \sum_s q_s \int f_s \, d\mathbf{v}, \qquad \mathbf{J} = \sum_s q_s \int f_s \mathbf{v} \, d\mathbf{v}.$$

In some cases Maxwell's equations can be replaced by a reduced model like Poisson's equation. The electric and magnetic fields $\mathbf{E}$ and $\mathbf{B}$ are superpositions of external fields and self-fields (space charge),

$$\mathbf{E} = \mathbf{E}_{\text{ext}} + \mathbf{E}_{\text{sc}}, \quad \mathbf{B} = \mathbf{B}_{\text{ext}} + \mathbf{B}_{\text{sc}}.$$

A Direct Fast FFT-Based Poisson Solver

Assume you know G the Green's function

The solution of the Poisson's equation

$$\nabla^2 \phi = -\rho/\varepsilon_0,$$

for the scalar potential, ϕ can be expressed as:

$$\phi(x,y,z) = \int\int\int dx'dy'dz'\rho(x',y',z')G(x-x',y-y',z-z'), \quad (1)$$

where G is the Green function and ρ is the charge density.
Discretisation of Eq. (1) on a grid with cell sizes h_x, h_y and h_z leads to:

$$\phi_{i,j,k} = h_x h_y h_z \sum_{i'=1}^{M_x} \sum_{j'=1}^{M_y} \sum_{k'=1}^{M_z} \rho_{i',j',k'} G_{i-i',j-j',k-k'}, \quad (2)$$

The solution of Eq. (2) can be obtained using FFT based convolution:

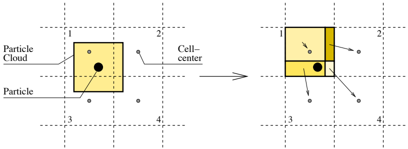
$$\phi_{i,j,k} = h_x h_y h_z \; \mathsf{FFT}^{-1}\{(\mathsf{FFT}\{\rho_{i,j,k}\}) \otimes (\mathsf{FFT}\{G_{i,j,k}\})\}$$

A fast Direct FFT-Based PIC Poisson Solver

Assume you know G the Green's function

Solving for ϕ using $\phi(\mathbf{x}) = \dfrac{1}{4\pi\varepsilon_0}\displaystyle\int G(\mathbf{x},\mathbf{x}')\rho(\mathbf{x})d\mathbf{x}'$ is expensive $\mathcal{O}(N^2)$ with N number of particles/grid-points.

1. Let Ω be spanned by a Cartesian structured mesh of $l \times n \times m$ with $l = 1 \dots M_x$, $n = 1 \dots M_y$ and $m = 1 \dots M_z$. The mesh size is a function of time: $h_x(t), h_y(t)$ and $h_z(t)$
2. Discretize $\rho \to \rho_h$ and $G \to G_h$ on a regular grid (PIC)



3. Go to Fourier space $\rho_h \to \widehat{\rho}_h$, $G_h \to \widehat{G}_h$ and convert the convolution into a multiplication $\widehat{\phi}_h = \widehat{\rho}_h * \widehat{G}_h$ in $\mathcal{O}(N \log N)$
4. Use a parallel FFT, particle and field load balancing

OPAL Releases I

- Consolidation has begun with V 2.0
  - ✓ major rewrite of OPAL-T, Distribution class
  - major rewrite of OPAL-CYCL, (ongoing)
  - ✓ strong typing, versioning of input files
  - ✓ no multipacting
  - ✓ no envelope model
  - ✓ Manual is converted to Wiki
  - ✓ `gitlab` and issue tracker
- new features (V. 2.1)
  - ✓ **Sampler - create random samples easily**
  - ✓ GA based MOOP fully integrated
- new features (V. 2.2)
  - ✓ curved multipoles (FFAG and Proton Therapy Modelling)
  - map tracking (experimental)

OPAL Releases II

- ✓ $\mathcal{M}_{sc}$ based on moments of the distribution
  - AMR-Fieldsolver
- new features (V. 2.4)
  - MultiGauss distribution for microbunched beams
  - `SOURCE` element can be made `TRANPARENT` for backtracking particles
  - many more plus all bug fixes can be found here

New Release Numbering since 2021

- OPAL 2021.1
  - New Undulator element with its own FDTD electromagnetic solver
  - Energy loss calculation and beam scattering for all light ions
  - ALPHA particles are supported in BEAM command
  - Gas stripping for DEUTERON beams and H2P beams in AIR
- OPAL 2022.1
  - Python interface for OPAL

OPAL Releases III

- P3M solver (Particle-Particle-Particle-Mesh)

Papers and Presentations: https://gitlab.psi.ch/OPAL/src/-/wikis/OPALPresentations

Future Plans for OPAL

$\Longrightarrow$ we will release one more “old“ OPAL i.e. OPAL 2023.1
$\Longrightarrow$ after that only bugfixes

Expecting for OPAL 2023.1:

- several documented issues are fixed
- pyOPAL is ready for FFA modelling
- more on FFA modelling

OPAL-X

Massive Parallelism, Performance, and Portability

- New hardware capabilities: **Exascale** (10^{18} FLOPS)

  $\Longrightarrow$ **Massive parallelism.**

- Must **efficiently** use these high operations per second

  $\Longrightarrow$ **Performance.**

- Architectures are **heterogeneous** i.e. CPUs & GPUs

  $\Longrightarrow$ **Portability.**

- Algorithm considered: **Particle-In-Cell** codes.

PAUL SCHERRER INSTITUT

PSI

OPAL-X

Massive Parallelism, Performance, and Portability

- New hardware capabilities: **Exascale** (10^{18} FLOPS)

  $\Longrightarrow$ **Massive parallelism**.

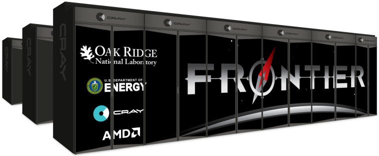
- Must **efficiently** use these high operations per second

  $\Longrightarrow$ **Performance**.

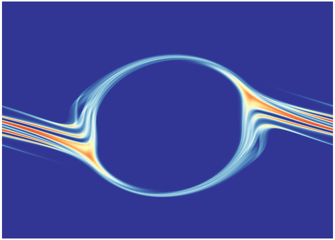
- Architectures are **heterogeneous** i.e. CPUs & GPUs

  $\Longrightarrow$ **Portability**.

- Algorithm considered: **Particle-In-Cell** codes.



Why Exascale Particle-in-Cell?



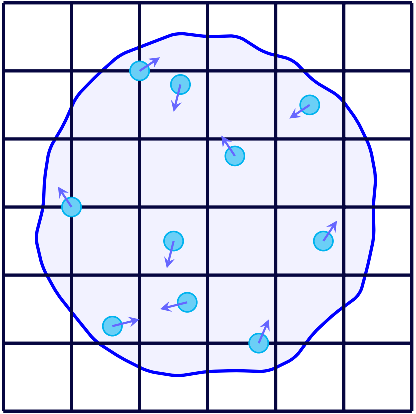
Two-stream instability.

$\implies$ Compute-intensive

$\implies$ High resolution

- Method used in **plasma physics**, **astrophysics**, and **accelerator physics**.
- Lends itself well to parallelization.
- Massive parallelism $\implies$ higher resolution simulations.

The Particle-in-Cell (PIC) method



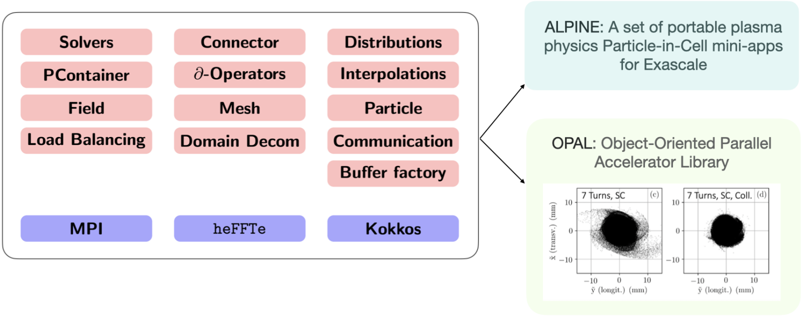
- macro ... real particles.
- Track evolution in phase space, but compute fields on the grid.
- Avoid pairwise computation $\mathcal{O}(N^2)$, $N =$ no. particles.
- Equations of motion:

$$\frac{d\boldsymbol{r}}{dt} = \frac{\boldsymbol{p}}{m\gamma}$$

$$\frac{d\boldsymbol{p}}{dt} = q\left(\boldsymbol{E} + \frac{\boldsymbol{p}}{m\gamma} \times \boldsymbol{B}\right)$$

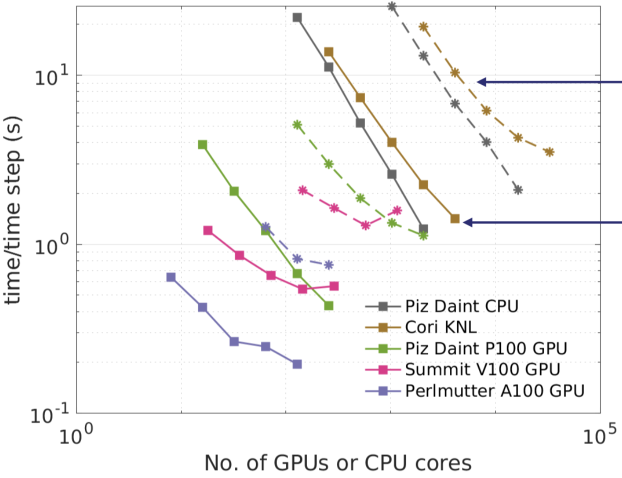
IPPL the base of OPAL

IPPL V2.0 Open source C++ library for Particle-in-Cell



For ALPINE see preprint

IPPL: Scaling across architectures



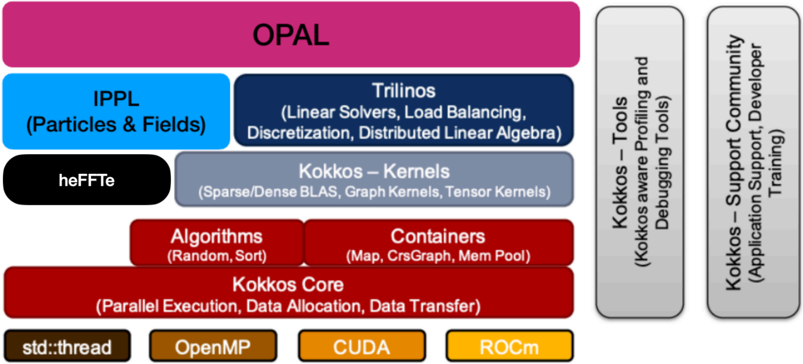
Dashed: 1024^3 grid, 8B particles

Solid: 512^3 grid, 1B particles

PAUL SCHERRER INSTITUT

OPAL-X I

getting getting ready for Exascale



Biggest changes are:

- C++20 (massive reduction of lines of code)

OPAL-X II

getting getting ready for Exascale

- no more OPAL flavours
- full FEM electromagnetic solver (PhD project)
- new $2\frac{1}{2}$ dimensional solver for long bunches
- add collisions beyond P3M
- OPAL can be controlled from Python (pyOPAL-X)