Improving IPPL: a performance portable library for grids and particles

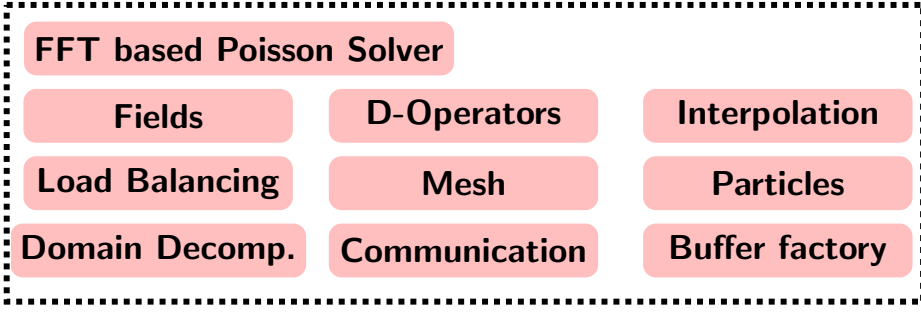
15th JLESC meeting

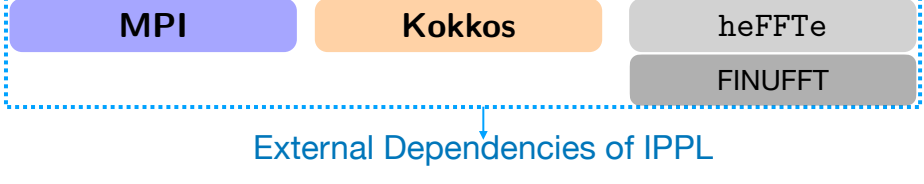
August 22, 2023 | Sriramkrishnan Muralikrishnan | Jülich supercomputing Centre, Germany

Goals of Independent Parallel Particle Layer (IPPL)

- **Open source** modern C++ (requires at least C++ 17) library for **grid and particle-based methods**
- **Performance Portability** across **heterogeneous parallel architectures (different CPUs, GPUs etc.)**
- Development of reusable, cross-domain components to **enable rapid application development**
- Prototyping library for the **development of novel numerical methods targeting exascale architectures**
- **Shorter time** from problem inception **to working parallel simulations**
- Primary application is **particle-in-cell** methods (backend of production particle accelerator code OPAL) but applicable for other use cases too

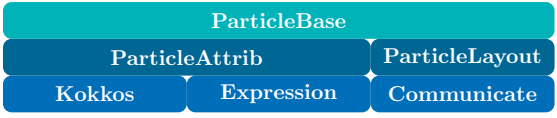
Structure of IPPL





https://gitlab.psi.ch/OPAL/Libraries/ippl

Particles in IPPL



- **Attributes:**
  - Struct of `Kokkos::Views`
  - Expression templates
  - Easily added to application
- **Communication:**
  - Particle layout classes
  - De-/serialize `Kokkos::View<char*>`
  - Pre-allocated buffers

```
using namespace ippl;

template<class PLayout>
struct Bunch
: public ParticleBase<PLayout> {
    Bunch(PLayout& playout)
    : ParticleBase<PLayout>(playout)
    {
        // add application attributes
        this->addAttribute(R);
        this->addAttribute(V);
        this->addAttribute(mass);
        this->addAttribute(charge);
    }

    ~Bunch() { }

    ParticleAttrib<double> mass, charge;

    ParticleAttrib<Vector<double>> R, V;
};

// compiles to single Kokkos kernel
bunch->R = bunch->R + dt * bunch->V;
```

Fields in IPPL

- **Scalar/Vector fields:**
  - `Kokkos::View`s
  - Expression templates
- **Interface to heFFTe**[6]
- **Communication:**
  - Field layout
  - Distribution of local domains globally known
  - De-/serialize `Kokkos::View<char*>`
  - Pre-allocated buffers

```
using namespace ippl;

constexpr unsigned int dim = 3;

int pt = 256;
Index I(pt);
NDIndex<dim> owned(I, I, I);

// specifies SERIAL, PARALLEL dimensions
e_dim_tag decomp[dim] = {PARALLEL,
                         PARALLEL,
                         PARALLEL};

FieldLayout<dim> layout(owned, decomp);

double dx = 1.0 / double(pt);
Vector<double, dim> hx = {dx, dx, dx};
Vector<double, dim> origin = {0, 0, 0};

UniformCartesian<double, dim>
    mesh(owned, hx, origin);

Field<double, dim> field(mesh, layout);
```

[6] Ayala A., Tomov S., Haidar A., Dongarra J. (2020) heFFTe: Highly Efficient FFT for Exascale. ICCS 2020. https://doi.org/10.1007/978-3-030-50371-0_19

Expression templates

- Avoids temporary objects in mathematical expressions
- Reduces number of Kokkos kernels
- Assignment operator evaluates expression
- Available for particles and fields
- Supported:
  - Binary operations: $+, -, *, /$
  - Comparison operations: $<, \leq, >, \geq, ==, != $
  - Bitwise operations
  - Many operations of `cmath` header

```
template<typename T, class... Properties>
template <typename E, size_t N>
ParticleAttrib<T, Properties...>&
ParticleAttrib<T, Properties...>::operator=(
    detail::Expression<E, N> const& expr)
{
    using capture_type =
        detail::CapturedExpression<E, N>;

    capture_type expr_ =
        reinterpret_cast<
            const capture_type&>(expr);

    Kokkos::parallel_for(
        "ParticleAttrib::operator=()",
        dview_m.extent(0),
        KOKKOS_CLASS_LAMBDA(const size_t i) {
            dview_m(i) = expr_(i);
    });

    return *this;
}
```

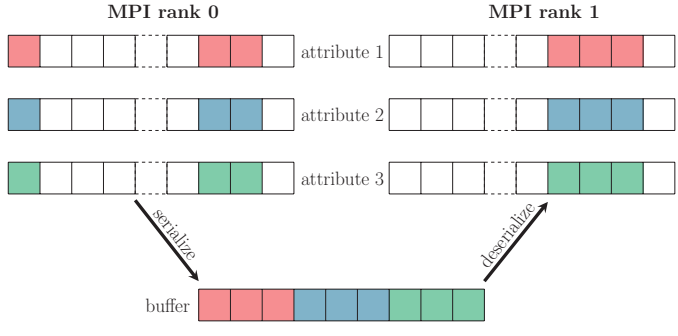
Communication

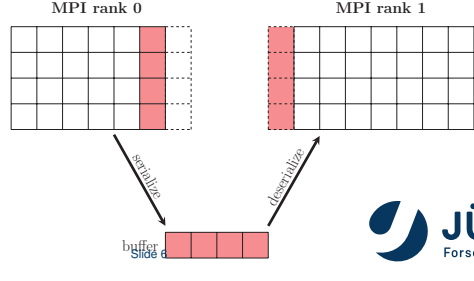
Particles:

- Locate particles to send
- Pack attributes into buffer
- Send buffer to receiver
- Unpack attributes

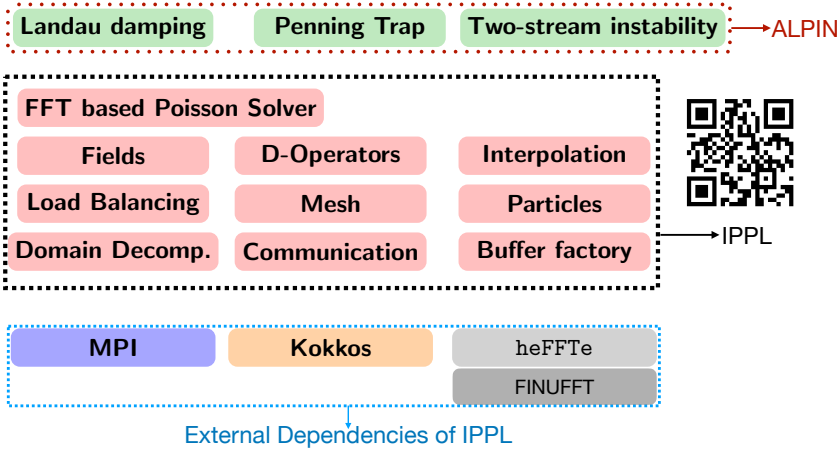
Fields:

- Get grid intersection
- Pack intersection into buffer
- Send buffer to receiver
- Unpack intersection



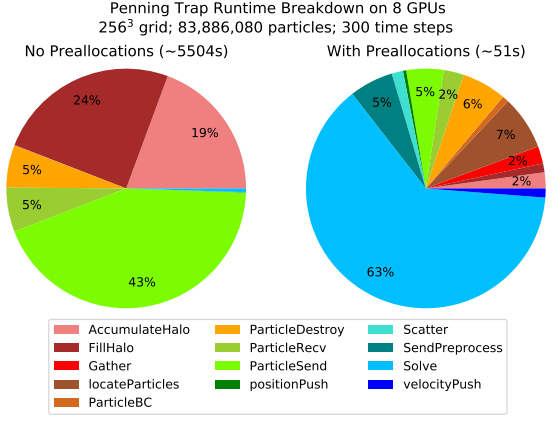


ALPINE (A set of performance portable pLasma physics Particle-in-cell mINi-apps for Exascale computing)



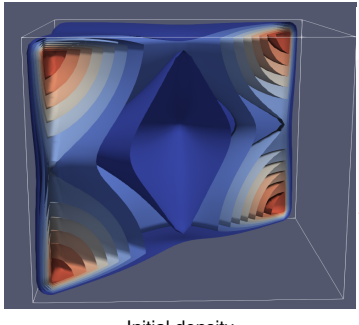
JÜLICH Forschungszentrum

Optimization by Preallocation

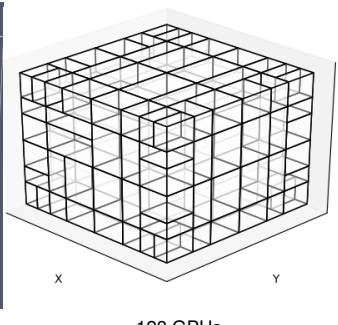


By preallocating the buffers used for field and particle communications and hence avoiding frequent `cudaMalloc` and `cudaFree` calls we are able to speedup the communication times in GPUs by $\mathcal{O}(10^3)$

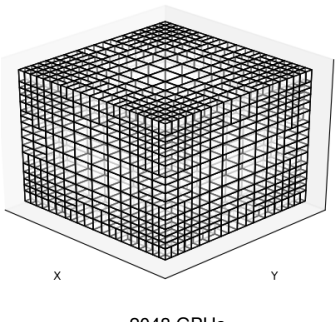
Nonlinear Landau Damping



Initial density



128 GPUs

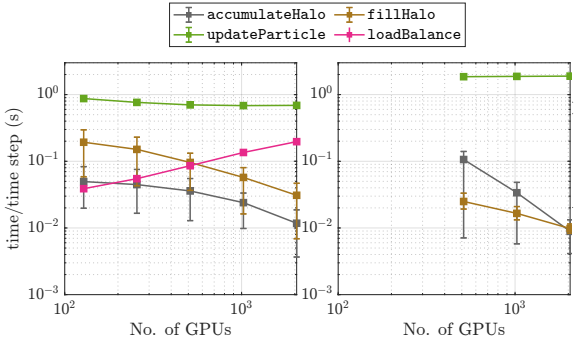


2048 GPUs

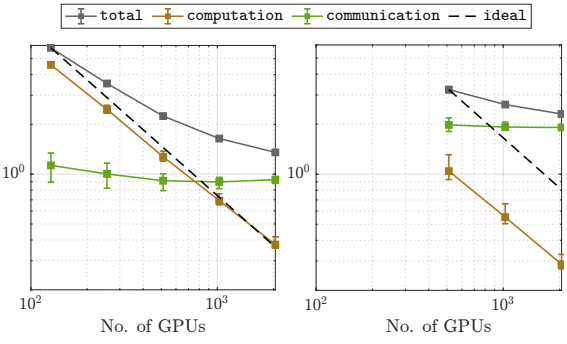
The non-uniform particle density requires particle load balancing to reduce memory requirements per GPU as well as communication costs

Strong scaling GPUs: Nonlinear Landau Damping

1024^3 grid, No. of particles $\approx$ 8 billion



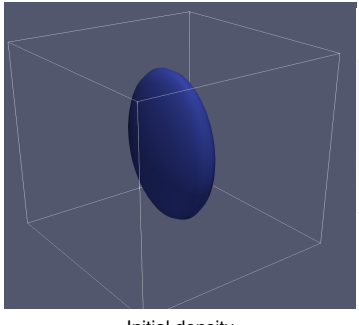
Communication kernels with (left) and without (right) load balancing



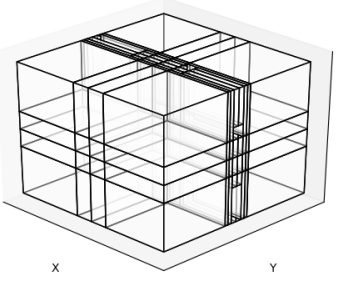
Computation Vs communication with (left) and without (right) load balancing

JÜLICH Forschungszentrum

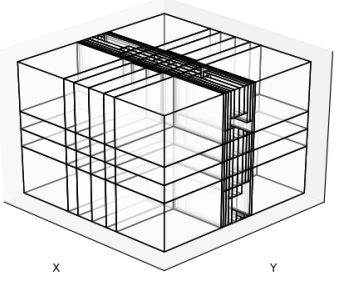
Penning Trap



Initial density



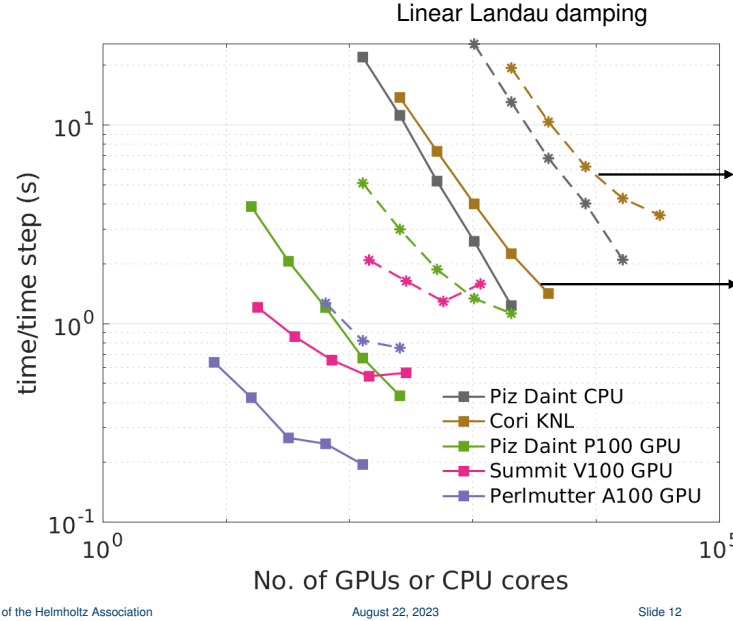
256 GPUs



2048 GPUs

In this case simulations fail without particle load balancing due to highly clustered particle bunch. But the particle load balancing leads to large imbalance in the fields and hence impact Poisson solve

Scaling across different architectures



Dashed lines: 1024^3 grid, 8 billion particles

Solid lines: 512^3 grid, 1 billion particles

Max. problem so far
34 billion grid points,
275 billion particles
on 4608 V100 GPUs in Summit at a rate of
13 billion particles/s

JÜLICH Forschungszentrum

Projects in IPPL

Past completed student projects:

- Performance portable Conjugate Gradient solver (A.Vinciguerra ETHZ Bachelor's thesis 2021)
- Domain decomposition and load balancing in IPPL (M. Ligotino ETHZ semester project 2021)
- Free space Poisson Solvers (S. Mayani EPFL Master's thesis 2021)
- Improving Particle Communication in IPPL (V. Montanaro ETHZ semester project 2022)
- Langevin solver (S. Klapproth ETHZ Bachelor's thesis 2023)

Ongoing projects:

- Extreme scale Electromagnetic PIC (S. Mayani ETHZ PhD thesis 2022-2026)
- Dimension Independence (A. Vinciguerra ETHZ Master's thesis 2023)
- Mixed precision (V. Montanaro ETHZ Master's thesis 2023)
- Collisions in PIC (T. Claglüna ETHZ Master's thesis 2023)
- Particle-in-Fourier and Parallel-in-Time Integration (S. Muralikrishnan and R. Speck 2023)

Things needed for IPPL and possible future projects

- Performant I/O with the capability to write directly the data in (multiple) GPUs to avoid expensive device to host transfers
- In-situ visualization with the data from the device
- Profiling tools (At the moment Kokkos tools can be used at the node level and vendor specific things for multiple nodes e.g. NVProf)
- Currently only finite difference and spectral methods are implemented. Interface to a performance portable FEM library
- Only explicit time integration methods are available at the moment. Interface to a performance portable linear solvers/preconditioners library for implicit/semi-implicit methods

IPPL team

- Sriramkrishnan Muralikrishnan (Jülich Supercomputing Centre, Germany)
- Matthias Frey (University of St Andrews, Scotland)
- Andreas Adelmann (Paul Scherrer Institute, Switzerland)

Students (all from ETH Zürich) who have contributed/contributing through their projects

- Tobiac Claglüna
- Severin Klapproth
- Michael Ligotino
- Sonali Mayani
- Veronica Montanaro
- Alessandro Vinciguerra

We are very interested in collaborating with others and please contact me s.muralikrishnan@fz-juelich.de if you are interested