Teammates: Marie Bocher, Casper Pranger

Mentors: Andreas Hirten, William Sawyer

Also involved: Dave May, Alexey Gokhberg, Patrick Sanan

GARNET

General

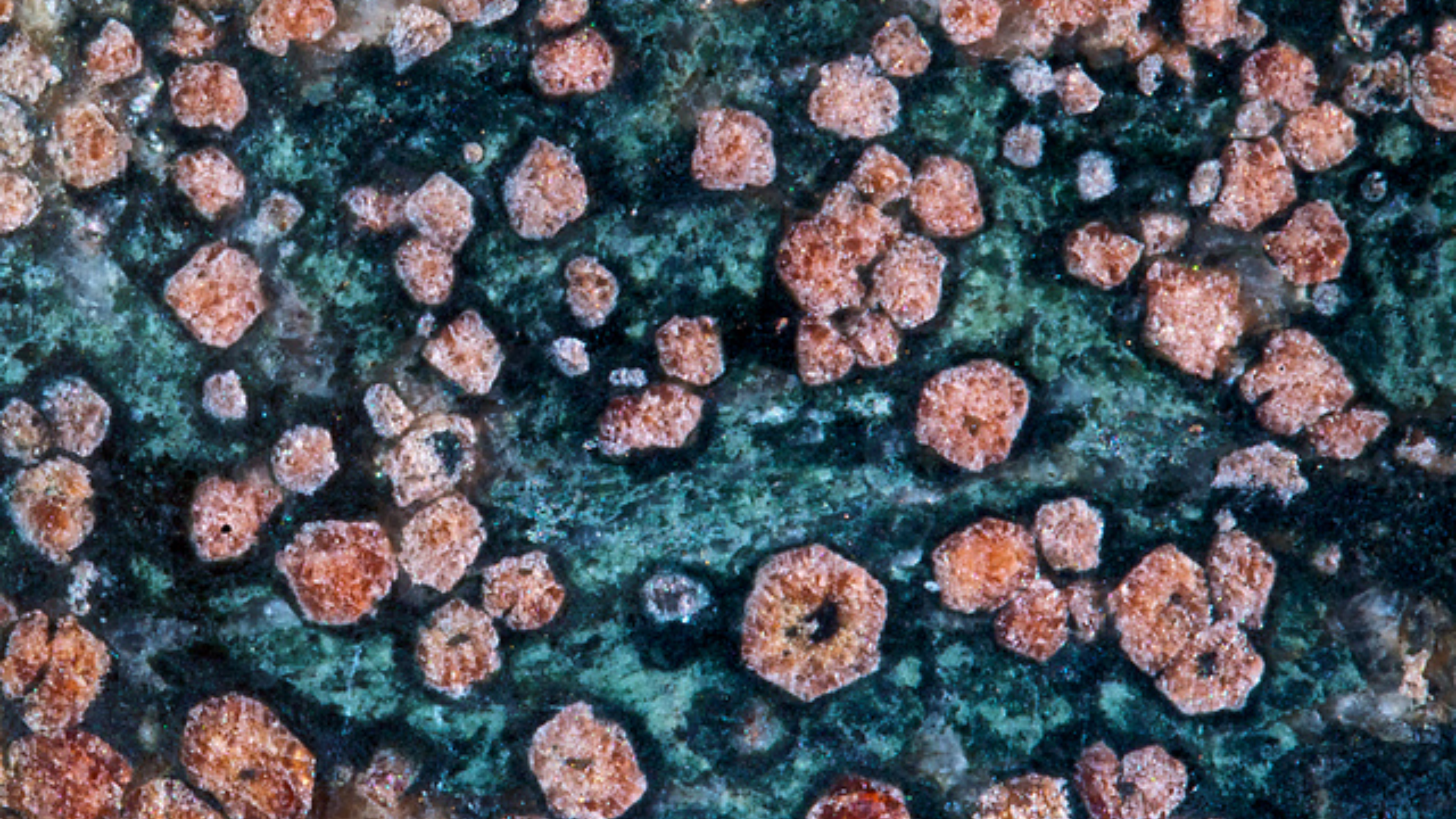
Algorithm

Regular grid
Residual-based
Root-finding

Newton-krylov
Nonlinear
N-dimensional

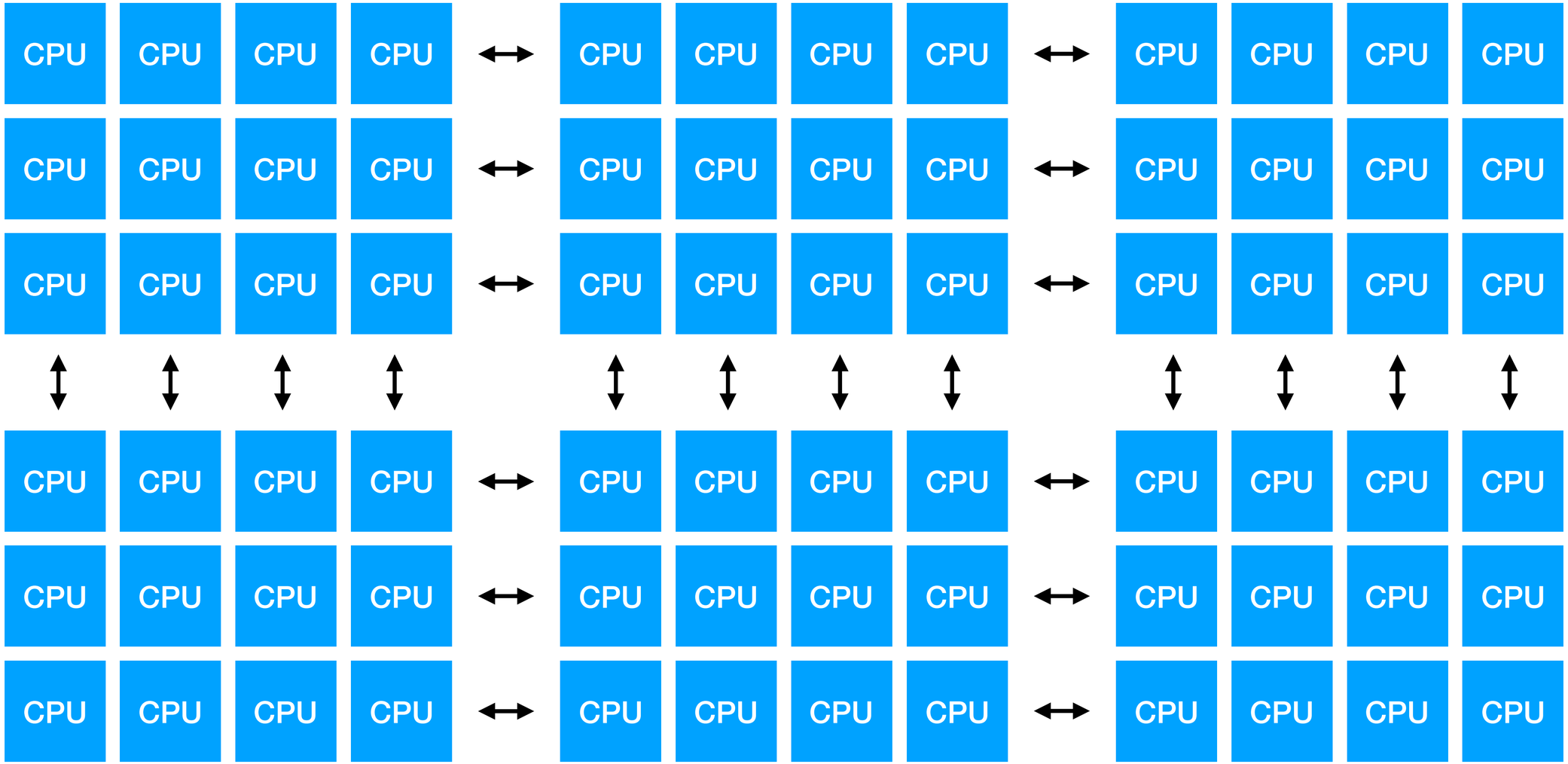
Earth sciences

Time-dependent
Tightly-coupled
Toolbox



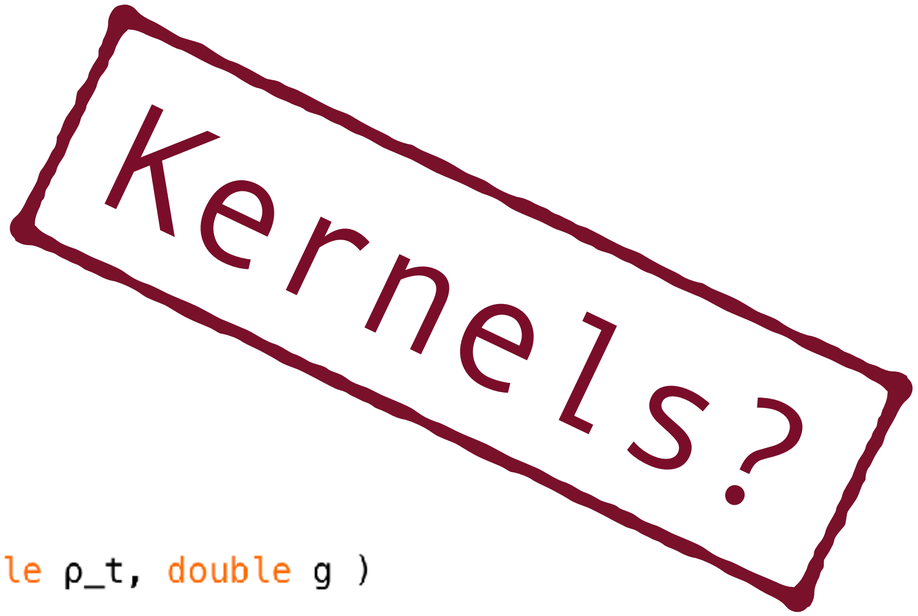
- Is built on top of PETSc, making use of its PC, KSP, and SNES objects.
- Provides automatic space/time discretization with staggered grid FD (1D/2D/3D).
- **Restriction:** (logically) cartesian rectangular.
- **Lightweight:** communication pattern only ever unidirectional 1D. Arithmetic indexing.
- Provides a toolbox of (tensor-valued) fields and differential operators.
- Implements only numerics, leaves physics to the user.
  - **Goal:** enabling fast development and execution of physics and algorithms

- Distributed domain decomposition (MPI P2P)
- Shared domain decomposition (MPI-3 SHMEM)
- Is a templated C++14 header-only library.

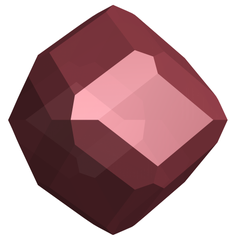

CPU

GPU

```
////////////////////////////////////////
//                                    //
//   POINT-WISE FUNCTION DEFINITIONS  //
//                                    //
////////////////////////////////////////

auto porosity_evolution = [&]( double P_t, double P_f )
    { return ( P_f - P_t ) / η_φ; };

auto total_density = [&]( double& φc )
    { return ρ_s * ( 1 - φc ) + ρ_f * φc; };

auto bulk_constitutive = [&]( double ϵ )
    { return 2 * η_s * ϵ; };

auto bulk_momentum_balance = [&]( double Δτ, double ΔP_t, double ρ_t, double g )
    { return Δτ - ΔP_t + ρ_t * g; };

auto bulk_mass_balance = [&]( double Δv_s, double P_t, double P_f, double φ )
    { return Δv_s + ( P_t - P_f ) / ( 1 - φ ) / η_φ ); };

auto darcy_flux = [&]( double ΔP_f, double φc, double g )
    { return  -k0 / η_f * std::pow(φc/φ0,3) * ( ΔP_f - ρ_f * g ); };

auto fluid_mass_momentum_balance = [&]( double Δq_D, double P_t, double P_f, double φ )
    { return Δq_D - ( P_t - P_f ) / ( 1 - φ ) / η_φ ); };
```

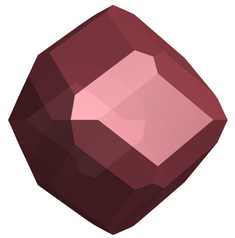


GARNET



```
////////////////////////////////////////
//                                    //
//   OBJECTIVE FUNCTION DEFINITION    //
//                                    //
////////////////////////////////////////

auto residual_evaluation = [&]( auto& Rv_s, auto& RP_t, auto& RP_f )
{
    // Porosity evolution
    φ.SetAlpha( 0. );
    φ.SetBeta ( porosity_evolution, P_t[0], P_f[0] );
    φ.TrivialSolve<BDF<1>>();

    φc.IsInterpolationOf( φ[0] );

    // Bulk momentum balance
    τ   .Set( bulk_constitutive, ϵ().RemoveTrace() );
    ρ_t .Set( total_density, φc );
    Rv_s.Set( bulk_momentum_balance, Δτ(), ΔP_t(), ρ_t, g );

    // Bulk mass balance
    RP_t.Set( bulk_mass_balance, Δv_s(), P_t[0], P_f[0], φ[0] );

    // Fluid mass and momentum balance
    q_D .Set( darcy_flux, ΔP_f(), φc, g );
    RP_f.Set( fluid_mass_momentum_balance, Δq_D(), P_t[0], P_f[0], φ[0] );
};
```

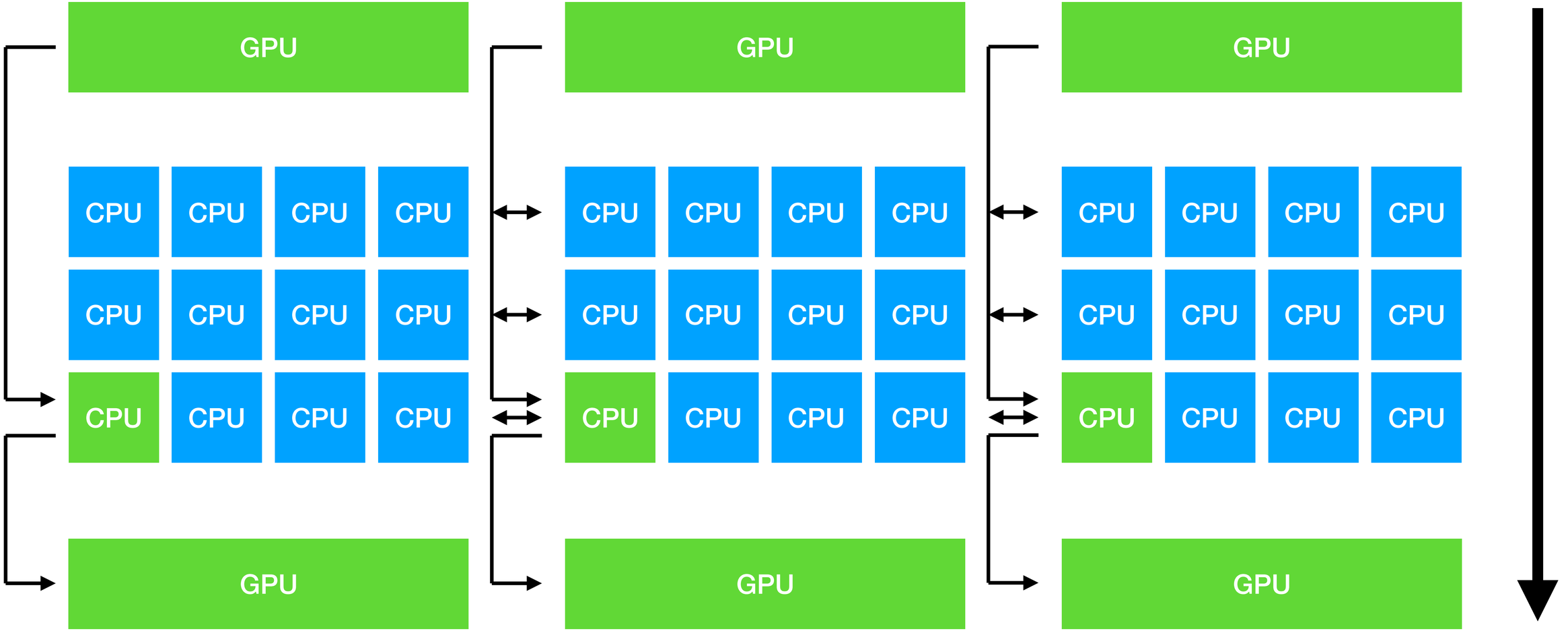
GARNET



```
////////////////////////////////////////
//                                    //
//   OBJECTIVE FUNCTION DEFINITION    //
//                                    //
////////////////////////////////////////

auto residual_evaluation = [&]( auto& Rv_s, auto& RP_t, auto& RP_f )
{
    // Porosity evolution
    φ.SetAlpha( 0. );
    φ.SetBeta ( porosity_evolution, P_t[0], P_f[0] );
    φ.TrivialSolve<BDF<1>>();

    φc.IsInterpolationOf( φ[0] );

    // Bulk momentum balance
    τ   .Set( bulk_constitutive, ϵ().RemoveTrace() );
    ρ_t .Set( total_density, φc );
    Rv_s.Set( bulk_momentum_balance, Δτ(), ΔP_t(), ρ_t, g );

    // Bulk mass balance
    RP_t.Set( bulk_mass_balance, Δv_s(), P_t[0], P_f[0], φ[0] );

    // Fluid mass and momentum balance
    q_D .Set( darcy_flux, ΔP_f(), φc, g );
    RP_f.Set( fluid_mass_momentum_balance, Δq_D(), P_t[0], P_f[0], φ[0] );
};
```

thrust::zip_iterators?

GARNET


GPU
GPU
GPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU
GPU
GPU
GPU

GARNET

- MPI3 shared memory (MPI_Win_alloc_shared) not compatible with CUDA managed memory :-(
- We are using thrust device_vectors and host_vectors and copy data explicitly
- So far, we have spent time fighting compilation issues (C++14)
- We are creating a class that manages both MPI3 shared memory and device memory.