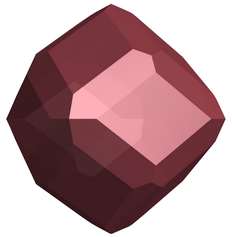
GARNET

Final Presentation

Casper Pranger, Marie Bocher
William Sawyer, Andreas Herten

About GARNET

- 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

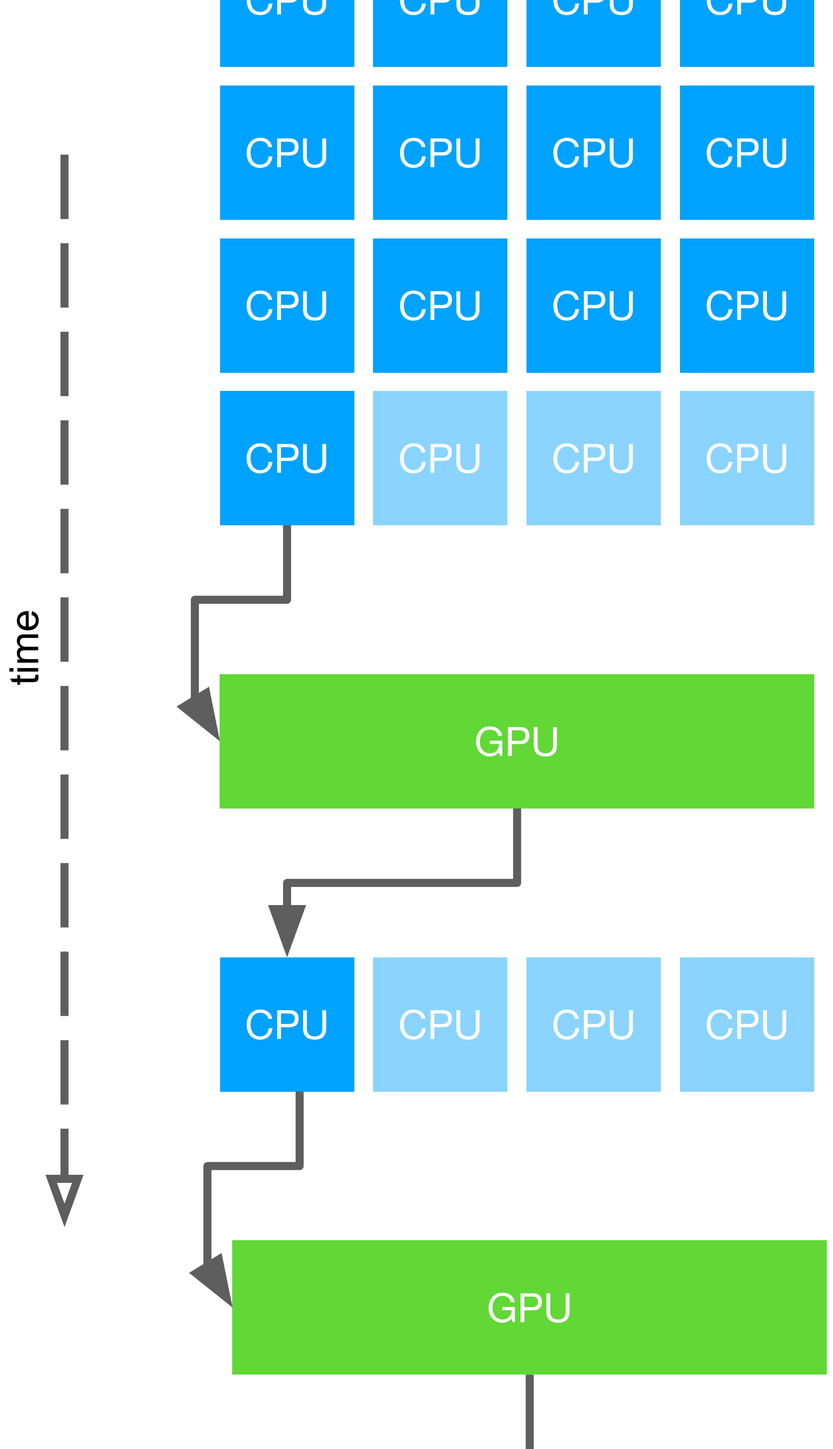


GARNET

Employed Technologies

- **CPU:**
  - C++14 (Variadic templates, lambdas, iterators, …)
  - MPI 3 Shared Memory & P2P
- **GPU:** same as above (also variadic templates!), plus **Thrust**
  - CUDA 9.0
  - GCC 6.2.0 (latest version supported by `nvcc`)

GARNET



- Multiple ranks (with MPI-shared memory) run on CPU data
- Before GPU invocation: MPI Barrier
- Rank 0 launches GPU computation on full memory
- After GPU computation conclusion: MPI Barrier
- Continue run of multiple MPI ranks on CPU data

Source Code Snippet – user interface

```
////////////////
//              //
//   PHYSICS    //
//              //
////////////////

// const double dt,
auto bulk_constitutive =
  [=] __host__ __device__ ( double v_1, double DeltaP )
    { return v_1 - dt * DeltaP / rho;  };

// const double dt,
auto momentum_balance =
  [=] __host__ __device__ ( double P_1, double Deltav )
    { return P_1 - dt * Deltav / beta; };
```

GARNET

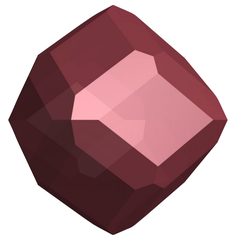
Source Code Snippet – user interface

```
////////////////////////////
//                        //
//   INITIAL CONDITIONS   //
//                        //
////////////////////////////

auto gaussian_pulse =
  [=] __host__ __device__ ( gpu::array<double,N> x )
    { return std::exp(-0.5*(x[0]*x[0]+x[1]*x[1])/(dx*dx)); };

P_1.SetCoordinated( gaussian_pulse );
```

GARNET

Source Code Snippet – backend

```
template< size_t N >
template< class F, class T, size_t ... I >
void Node<N>::set_impl( F&& f, T args, std::index_sequence<I...> )
{

  ...

  auto assignment_f = [=] __host__ __device__ ( arg_tuple_t args ) -> bool
    { thrust::get<0>(args) = f( thrust::get<I+1>(args)... ); return true; };

  grid->amorphous_loop( domain, assignment_f, *this,
std::forward<decltype(std::get<I>(args))>(std::get<I>(args))... );

  ...

}
```



GARNET

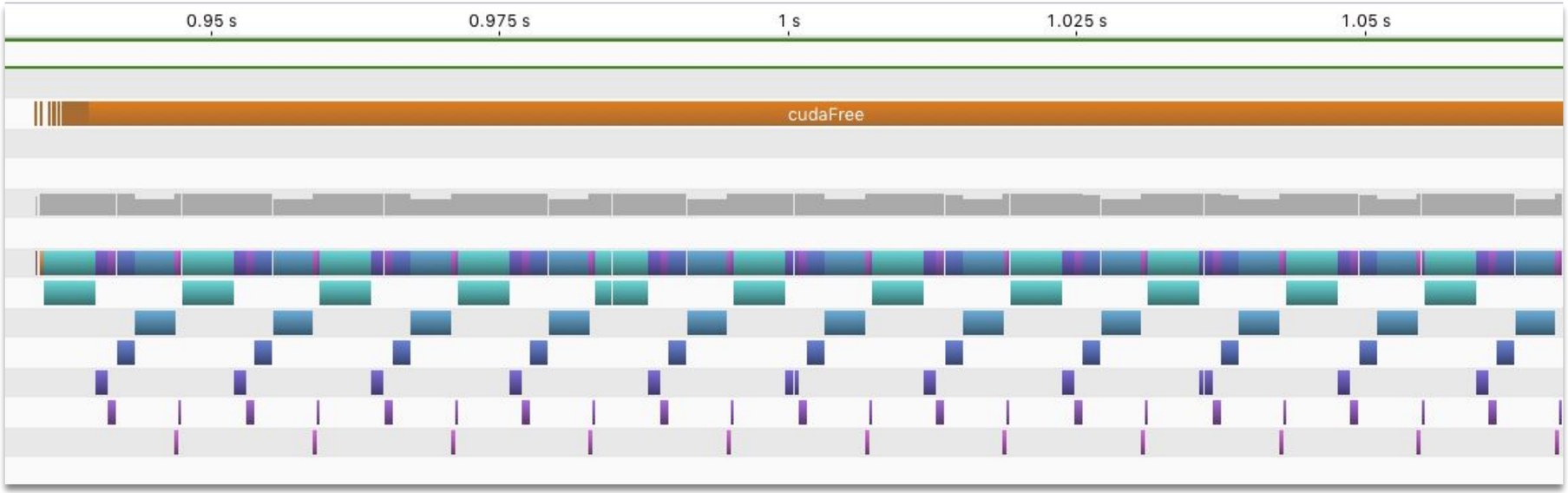
Source Code Snippet – backend

```
template< size_t _N >
template< class F, class ... A >
void Grid<_N>::amorphous_loop( Domain<_N>* domain, F&& f, A&&... args ) const
{
  auto arg_begin_iterator = thrust::make_zip_iterator(
    thrust::make_tuple(
      thrust_iteratoror<std::decay_t<A>,N>(args).d_begin()...
    )
  );

  auto arg_end_iterator   = thrust::make_zip_iterator(
    thrust::make_tuple(
      thrust_iteratoror<std::decay_t<A>,N>(args).d_end()...
    )
  );

  auto discard = thrust::make_discard_iterator();

  MPI_Barrier( domain->get_shared_comm() );
  if( mpi::get_rank( domain->get_shared_comm() ) == 0 )
    thrust::transform( arg_begin_iterator, arg_end_iterator, discard, f );
  MPI_Barrier( domain->get_shared_comm() );
}
```

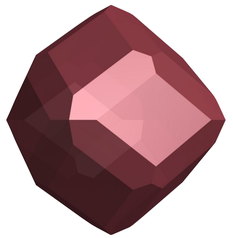
GARNET

Speedup

- Currently: 40×
  - (1 GPU) vs. (CPU with 12 MPI ranks (1 node))
  - Simple example which exclusively runs on GPU
  - *But something strange is going on in CPU code* 



GARNET



Future Work

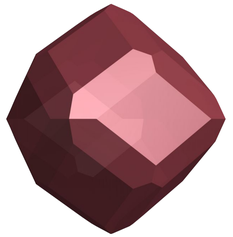
- GPU-capable boundary conditions, MPI P2P halo exchange
- Compose stencil iterators with point-wise iterators at compile-time
- Move back up to real applications
    - complicated physics => more computational intensity on GPU
- Merge CPU- and GPU-capable versions into one code -- How?

GARNET

Comments / Feedback

- Many issues of C++14 compatibility with compilers, debuggers, mentors, libraries…

  Will improve with time? But sooner rather than later ;-)
- Thrust on Github is not the latest version of Thrust
  - Make Thrust variadic already!
- Error messages are cryptic!
  - Let Thrust produce better error message (is that even possible?)
  - Offer CUDA-compatible Clang on Piz Daint
- Hackathon could be maybe two weeks long :-)
- We'll be back! -- would be far more productive with gpu working from the outset



GARNET