WHY I HATE CI

GRTECLYN UPDATE: PORTING THE MATTER CLASSES

Juliana Kwan (DAMTP, Cambridge)

✓ **Update gpu-build.yml with CUDA 12.5**

Browse files

training/202406_grtl_meeting

julianakwan committed 5 days ago Verified

1 parent 82fdf08 commit d744e34

Showing **1 changed file** with **1 addition** and **1 deletion**.

Whitespace | Ignore whitespace | Split | Unified

2 .github/workflows/gpu-build.yml

```
@@ -29,7 +29,7 @@ jobs:
29 29        AMREX_HOME: ${{ github.workspace }}/amrex
30 30        BUILD_ARGS: USE_${{ matrix.gpu-backend }}=TRUE DEBUG=${{ matrix.debug }} USE_MPI=TRUE
31 31        CUDA_MAJOR_VERSION: 12
32    -      CUDA_MINOR_VERSION: 0
   32 +      CUDA_MINOR_VERSION: 5
33 33
34 34      steps:
35 35      - name: Checkout AMReX
```

✗ **Merge branch 'training/202406_grchombo_meeting' of github.com:GRTLCol...**

…laboration/GRTeclyn into training/202406_grchombo_meeting because I modified gpu-build.yml on the website interface

training/202406_grtl_meeting

julianakwan committed 5 days ago

Showing **1 changed file** with **1 addition** and **1 deletion**.

Whitespace | Ignore whitespace | Split | Unified

2 .github/workflows/gpu-build.yml

```
@@ -29,7 +29,7 @@ jobs:
29 29        AMREX_HOME: ${{ github.workspace }}/amrex
30 30        BUILD_ARGS: USE_${{ matrix.gpu-backend }}=TRUE DEBUG=${{ matrix.debug }} USE_MPI=TRUE
31 31        CUDA_MAJOR_VERSION: 12
32    -      CUDA_MINOR_VERSION: 0
   32 +      CUDA_MINOR_VERSION: 5
33 33
34 34      steps:
35 35      - name: Checkout AMReX
```

HOW TO AMREXIFY EXISTING GRCHOMBO CODE

- BoxLoops -> ParallelFor

```
72        // Calculate CCZ4 right hand side
73        if (m_p.max_spatial_derivative_order == 4)
74        {
75            BoxLoops::loop(CCZ4RHS<MovingPunctureGauge, FourthOrderDerivatives>(
76                                m_p.ccz4_params, m_dx, m_p.sigma, m_p.formulation),
77                           a_soln, a_rhs, EXCLUDE_GHOST_CELLS);
78        }
```

GRChombo

- Some functions need to be prefaced by macros so that CUDA/HIP knows they belong to the device or host.

```
106         // Calculate CCZ4 right hand side
107         if (simParams().max_spatial_derivative_order == 4)
108         {
109             CCZ4RHS<MovingPunctureGauge, FourthOrderDerivatives> ccz4rhs(
110                 simParams().ccz4_params, Geom().CellSize(0), simParams().sigma,
111                 simParams().formulation);
112             amrex::ParallelFor(
113                 a_rhs,
114                 [=] AMREX_GPU_DEVICE(int box_no, int i, int j, int k) {
115                     ccz4rhs.compute(i, j, k, rhs_arrs[box_no], soln_c_arrs[box_no]);
116                 });
117         }
```

GRTeclyn

```
#define AMREX_GPU_HOST        __host__
#define AMREX_GPU_DEVICE      __device__
#define AMREX_GPU_GLOBAL      __global__
#define AMREX_GPU_HOST_DEVICE __host__ __device__
```

HOW TO AMREXIFY EXISTING GRCHOMBO CODE

- A lot of compute functions now have the form (int i, int j, int k, const amrex::Array4<data_t> &out, const amrex::Array4<data_t const> &in) instead of passing in a Cell object
  - inner const -> data is read only
  - outer const -> can't reposition pointer
- Member functions are now public so they are discoverable by CUDA (these were protected in GRChombo)

```
template <class matter_t, class gauge_t, class deriv_t>
template <class data_t>
void MatterCCZ4RHS<matter_t, gauge_t, deriv_t>::compute(
    Cell<data_t> current_cell) const
```

GRChombo

```
template <class matter_t, class gauge_t, class deriv_t>
template <class data_t>
AMREX_GPU_DEVICE AMREX_FORCE_INLINE void
MatterCCZ4RHS<matter_t, gauge_t, deriv_t>::compute(
    int i, int j, int k, const amrex::Array4<data_t> &rhs,
    const amrex::Array4<data_t const> &state) const
```

GRTeclyn

PORTING THE MATTER CLASSES

- MatterCCZ4RHS: usual CCZ4RHS but includes two extra variables for the matter field and conjugate momentum
- ScalarField class: matter_t object that is a specific type for many of these other objects
- DefaultPotential class: empty potential V = 0
- Potential: has a mass term also
- MatterWeyl4
- EMTensor
- MatterConstraints: includes Ham and Mom
- ScalarBubble: generates initial condition -> moved to InitialConditions/ScalarField directory
- FixedGridTagging : (incidental) the ScalarField example had fixed grids, so I ported this also

UNIT TESTING

- These classes have been tested under the doctest framework:
  - BSSNMatterTest
    - Run using -dt-tc="Matter BSSN"
    - Tests for:
      - RHS calculation
      - Hamiltonian and the momentum vectors ('Ham', 'Mom1', ...)
  - MatterWeyl4
    - Run using -dt-tc="Matter Weyl4":
    - Tests for:
      - EMTensor ('rho')
      - Weyl4 calculation ('WeylRe' and 'WeylIm")
- Note that I have a slightly different approach to the HDF5 file dump to Miren - all the fields are named in the HDF5 file so you can separate them for debugging using h5dump or h5ls or h5py.

SCALAR FIELD EXAMPLE

- In GRChombo, this was a scalar bubble + Kerr BH -> only the scalar bubble remains
- Same functions you know and love:
  - specificAdvance -> same
  - initialData -> initData
  - specificEvalRHS -> same
  - specificUpdateODE -> same
  - computeTaggingCriterion -> errorEst
  - specificPostTimeStep ->
  - *new* derive
- To see member functions/inheritance diagrams, refer to the AMReX doxygen pages

THE GITHUB REPOSITORY

- GitHub.com/GRTLCollaboration/GRTeclyn
- 3 branches:
  - enhancement/matter_class:
    - updated matter classes, unit tests
  - enhancement/scalar_fields:
    - updated matter classes, no unit tests, and the scalar field example in the Examples/ directory
  - training/202406_grtl_meeting: hands-on exercise for this afternoon

HOW TO RUN

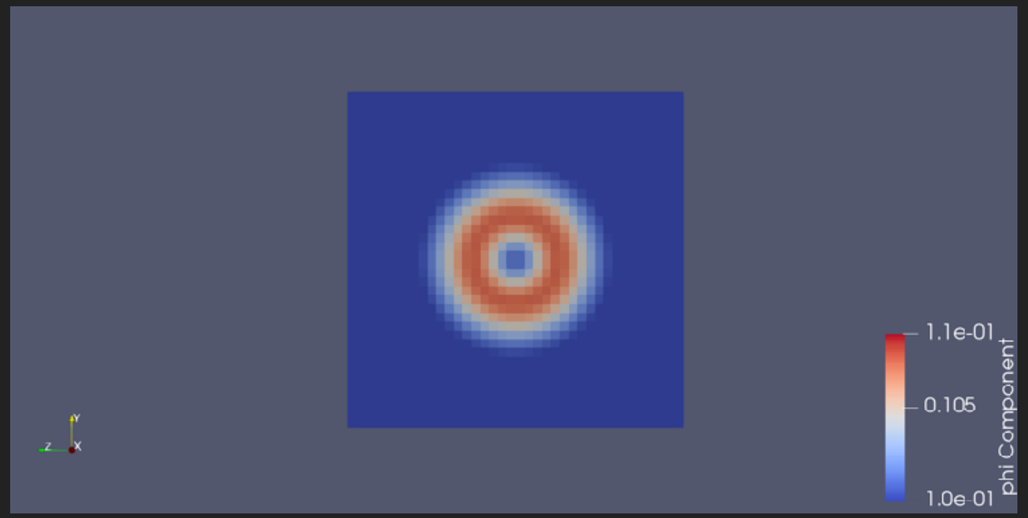
- The readme on training/202406_grtl_meeting contains a comprehensive guide to running the scalar field example on Dawn.
- For other systems, I have found that the old GRChombo module files work well
  - e.g. on Icelake nodes, source ~/GRChombo/InstallNotes/MakeDefsLocalExamples/CSD3-Icelake-Intel-modules.sh, then set USE_COMP=intel-llvm
- Example/ScalarFields includes a sample params.txt file. But you must change the output directory! Also notice that some variables have been commented out because they use the default value.

REGRESSION TESTING

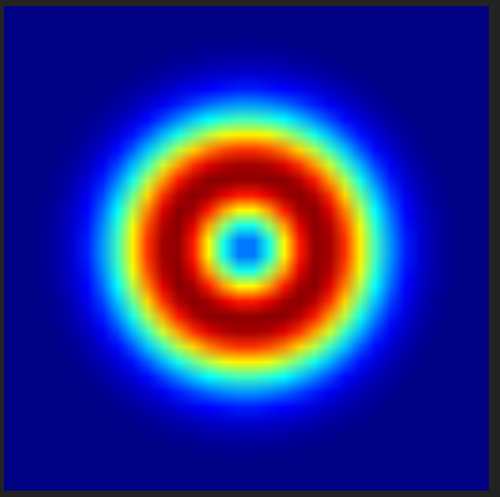
- How do you know if you have the right output?
- There is a params_test.txt in the directory and I have also uploaded a small output file to .github/workflows/data
- AMReX provides a tool called fcompare:
  - fcompare.<comp>.ex plt_00003 .github/workflows/data/plt_scalar_field_compare_00003
- For more details see: https://amrex-codes.github.io/amrex/docs_html/Post_Processing.html#fcompare

VIEWING THE OUTPUTS

- ParaView
  - plt* files; select AMReX/BoxLib grid reader
- yt
  - import yt; my_data = yt.load("plt…")
- The snapshot tool included with AMReX:
  - Go to ~/amrex/Tools/Plotfile; build with make
  - Run with ./fsnapshot.<comp>.ex -v phi -p Palette plt_00003
  - The output is in the parent directory of plt…
  - https://amrex-codes.github.io/amrex/docs_html/Post_Processing.html#fsnapshot



ParaView



fsnapshot

FUTURE WORK

- CI fixes: the gpu-build workflow fails (although it will compile with CUDA and SYCL on CSD3)
- Parameter file could be cleaned up e.g. params.txt has some unused options, plot_vars is currently ignored
- For historical reasons, some files are called NewMatterConstraints etc. and these need to be cleaned up.
- Support for multiple scalar fields