CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

CUDA: MiniApp

Ben Cumming, CSCS
July 27, 2015

Unit Tests

Unit testing is a **very good** development habit to get into

- write tests for features before implementing them
- tests will pass when feature is finished
- if a feature is broken in the future, the test will catch it immediately
- tests can also be used to catch performance regressions

The CUDA miniapp uses unit tests to validate the blas1 functions in `linalg.cu`

- initially all of the tests fail!
- the tests are run with each `make`

There are some very good open source unit testing frameworks available

- I use Google Test for my own work

Step 1: Host-Device Data Synchronization

The `Field` class implements storage of the 2D fields used in the miniapp

- it is extended for the CUDA version to store a copy of each field in both host and device memory
- the data is pointed to by the `Field::host_pointer_` and `Field::device_pointer_` members

The data stored in the host and device regions can get out of synch

1. implement the member functions that synchronize the host and device data fields in `data.h`
2. ensure that the initial conditions are updated on the device after computation in `main.cu`

One of the unit tests will pass when task 1 is finished.

Step 2: Linear Algebra

Implement all of the BLAS level 1 linear algebra kernels in `linalg.cu`

- look for the `TODO`s
- each kernel is covered by a unit test
- you will have to do some research into the cublas routines

all unit tests will pass when this task is finished

Step 3: Stencils

implement the stencils in `operators.cu`

- look for the `TODO`s

the number of CG iterations and visualization can be used to validate results

- **extra** can you use shared memory for the interior stencil?
- **extra** can you design an implementation that uses shared memory to implement the interior, boundary and corner stencils in one, simplified kernel?

How does time to solution compare with that for the MPI, OpenMP and serial versions?

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Thank you for lasting this long!