Discussion on doing matrix element calculation on GPU for M3D-C1

The loop layers of M3D-C1

- Loop over all elements
  - Loop over basis function (nu)
    - Calculate the integral of (mu, F[nu]) using quadratures
      - Loop over test function (mu)

Parallization over element loop

- Loop over all elements (MPI and OpenMP)
  - Loop over basis function (nu)
    - Calculate the integral of (mu, F[nu]) using quadratures
      - Loop over test function (mu)

Parallization over element loop

- Loop over all elements (MPI and OpenMP)
  - Loop over basis function (nu)
    - Calculate the integral of (mu, F[nu]) using quadratures
      - Loop over test function (mu)
- Issues with parallization over elements
  - Calculation in on element is heavy
  - Insertion of matrix element requires serial excution (OpenMP critical) and calls to SCOREC library

Parallization over basis and test

- Loop over all elements (MPI)
  - Loop over basis function (nu) (OpenACC)
    - Loop over test function (mu) (OpenACC)
      - Calculate the integral of (mu, F[nu]) using quadratures
- Calculation for one OpenACC kernel is light (memory efficient)
- Jobs are independent, no serial part is needed.

Element paralization using CUDA MPS

- CUDA Multi-Process Server (MPS) is a feature that allows multiple CUDA processes to share a single GPU context. each process receive some subset of the available connections to that GPU.
- MPS allows overlapping of kernel and memcopyoperations from different processes on the GPU to achieve maximum utilization
- Requirement
  - Launch MPS server before the program begins
  - Officially supported by Summit cluster

List of changes required

- Offload global variables required for element calculation
- Offload element quadrature results (*79)
- Reduce dimension of test function variables in element calcualtion
  - Replace intx with int in metricterms.f90
- Make all the temporary variables private for OpenACC kernels

Test results

- We test the code on traverse cluster
- 4 planes with 6770 elements on each
- numvar=3 isplitstep=1
- 4 nodes 64 cores per node vs. 4 GPUs per node
- 4*32 MPI processes
- Time taken for velocity matrix terms calculation
  - CPU 40s / 34s (Open MP 4 threads)
  - GPU 6.7s
- OpenACC kernel takes 1541MB for each process