OAK RIDGE National Laboratory

Accelerating OpenFOAM on Frontier GPUs for Fusion Multiphysics Simulations

Arpan Sircar

R&D Associate Staff

Nuclear Energy and Fuel Cycle Division

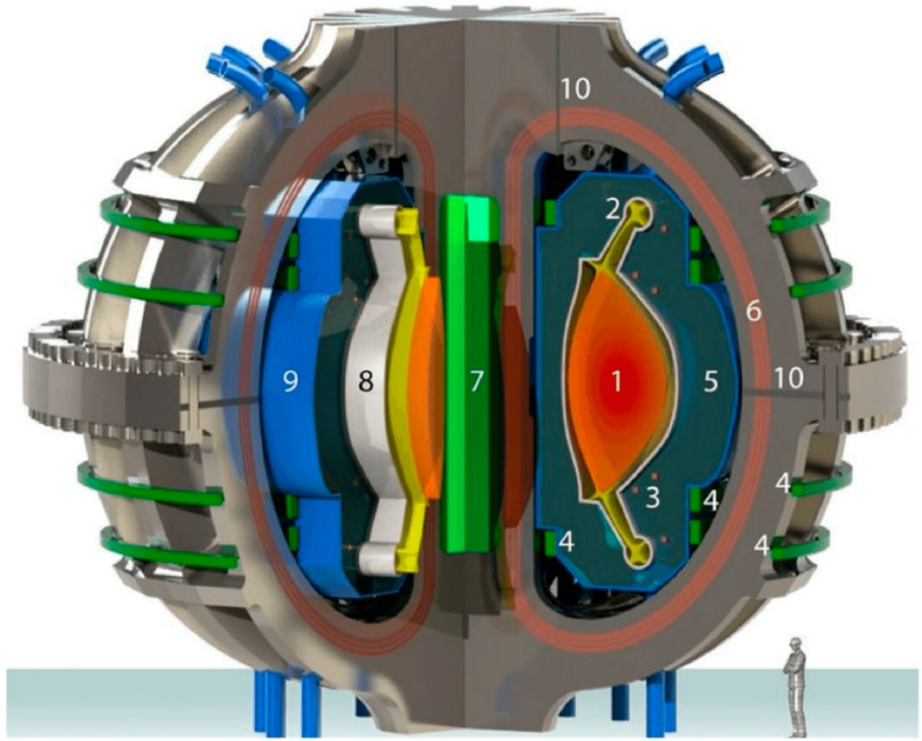
ORNL is managed by UT-Battelle, LLC for the US Department of Energy

September 11, 2024

U.S. DEPARTMENT OF ENERGY

Background and Significance

- Target: integrated simulation of fusion blankets for faster design analyses
- How: accurate high-fidelity computational tools
  - FERMI (ARPAe award for engineering multiphysics)
  - FREDA (SciDAC award to coupled plasma physics and engineering modules)
  - INCITE award for exascale simulations
- This work: focus on acceleration of the fluid simulations

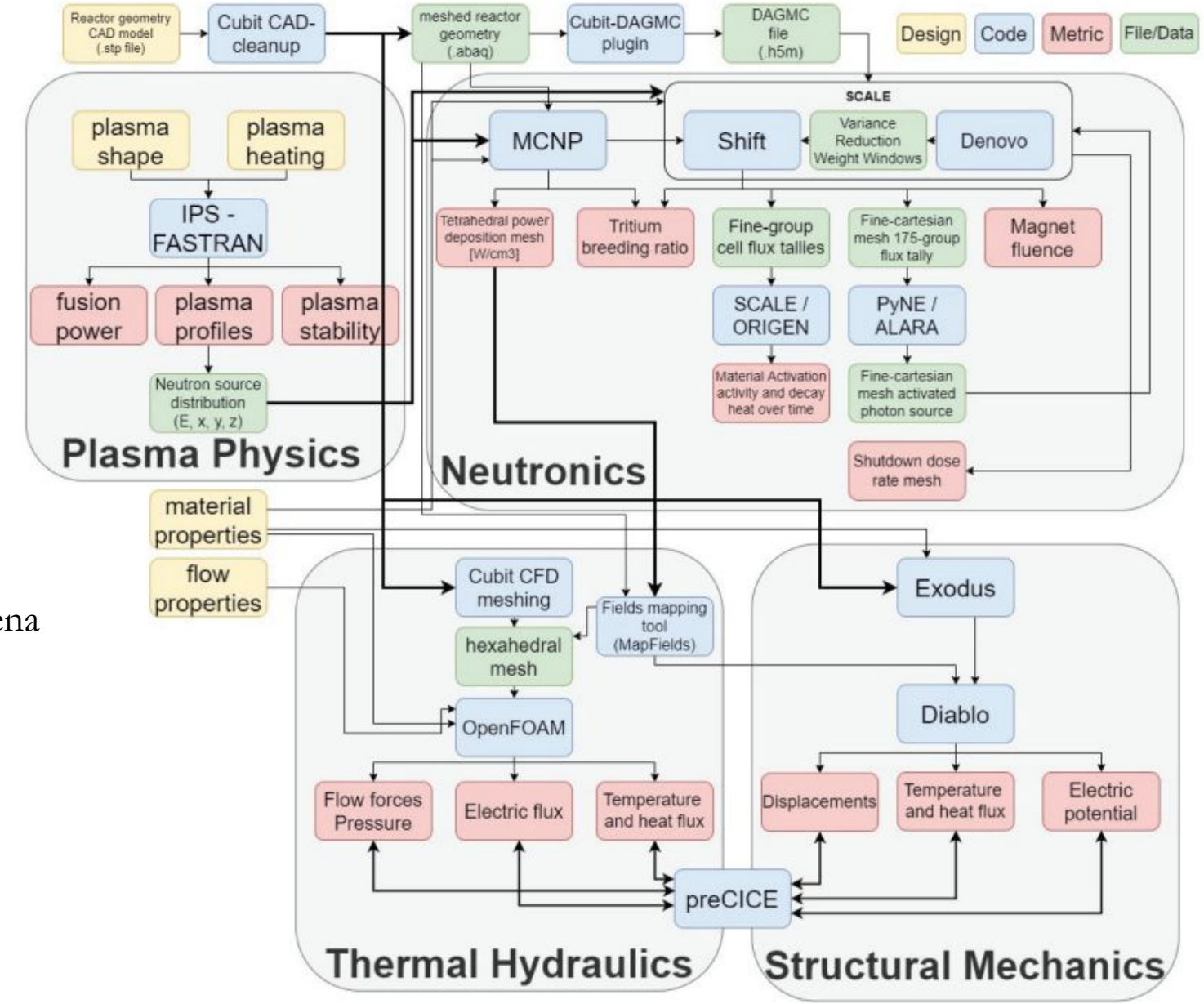


1. plasma
2. newly designed divertor
3. copper trim coils
4. HTS poloidal field coils
5. FLiBe blanket
6. HTS toroidal field coils
7. HTS solonoid
8. vacuum vessel
9. FLiBe tank
10. joints in toroidal field coils

The ARC-class fusion device is a tokamak fusion design from Commonwealth Fusion Systems (CFS)

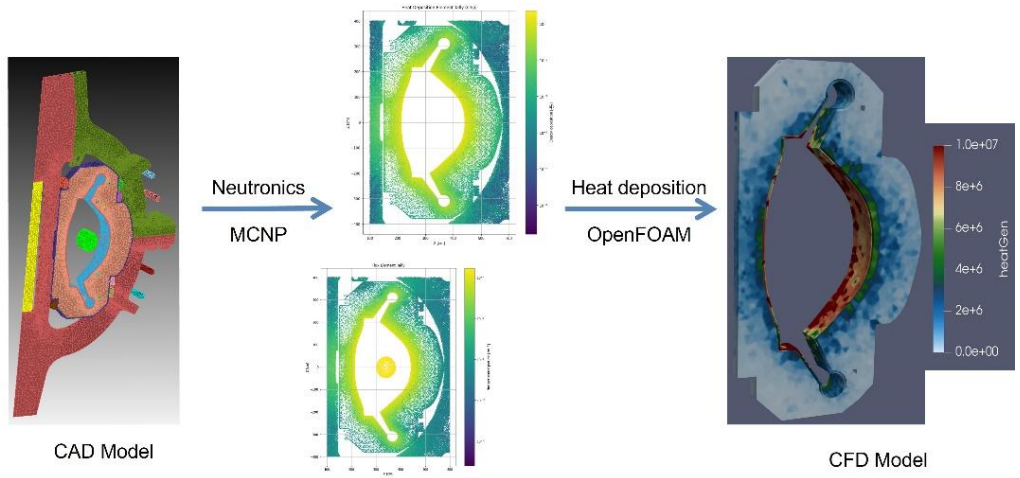
FERMI: Fusion Energy Reactor Models Integrator

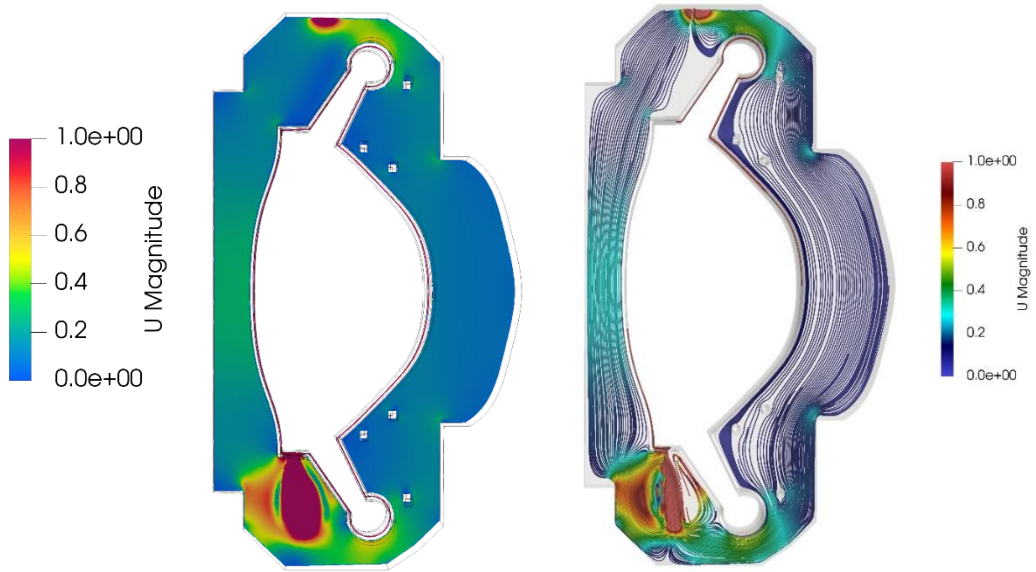
Multiphysics simulations for engineering analysis of fusion blankets



- Coupled simulations for multiphysics phenomena
- Scalable procedures for advanced high performance computing architectures
- Optimization, parameterization, validation & verification, predictions and informed design

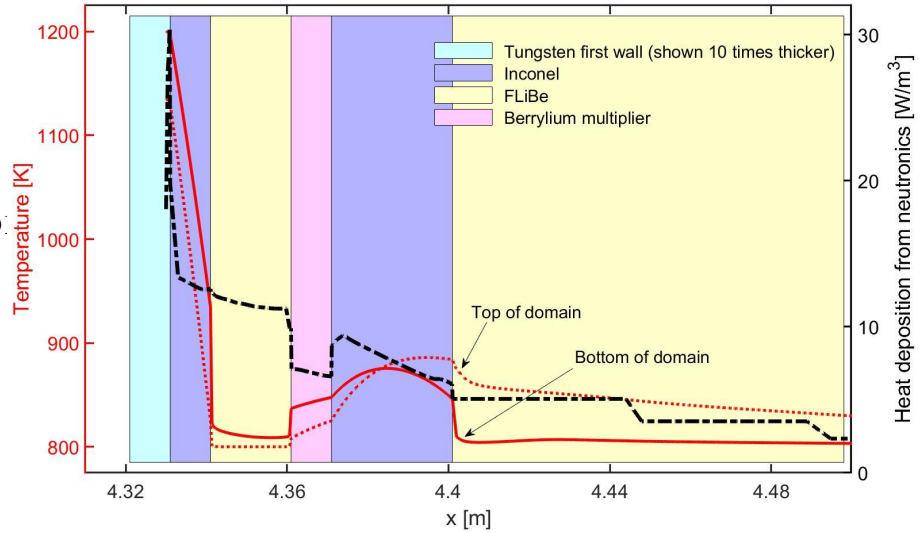
Coupled neutronics and CFD simulations are performed

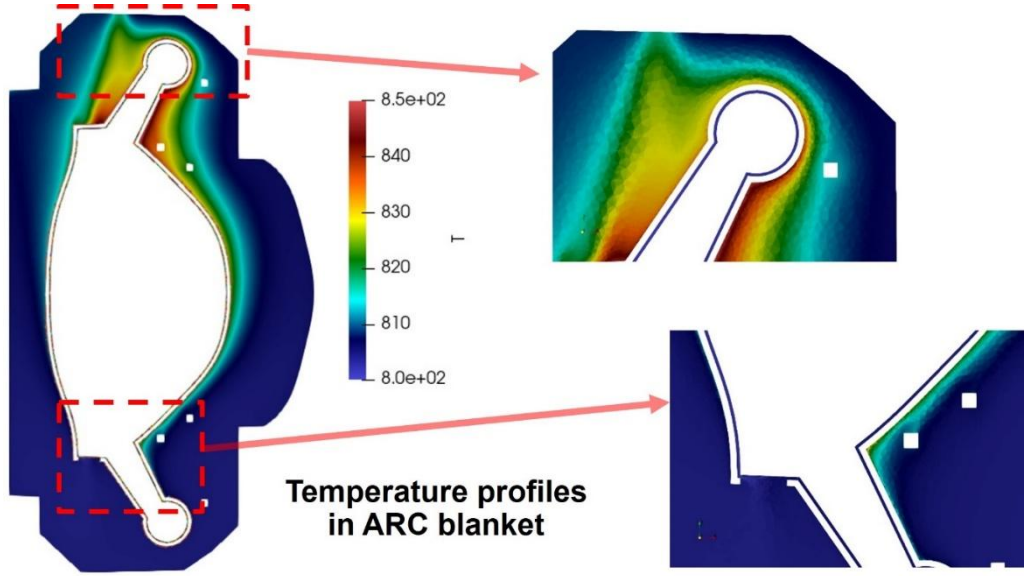




Heat deposition from neutronics is transferred to the CFD domain and mapped onto the CFD grid

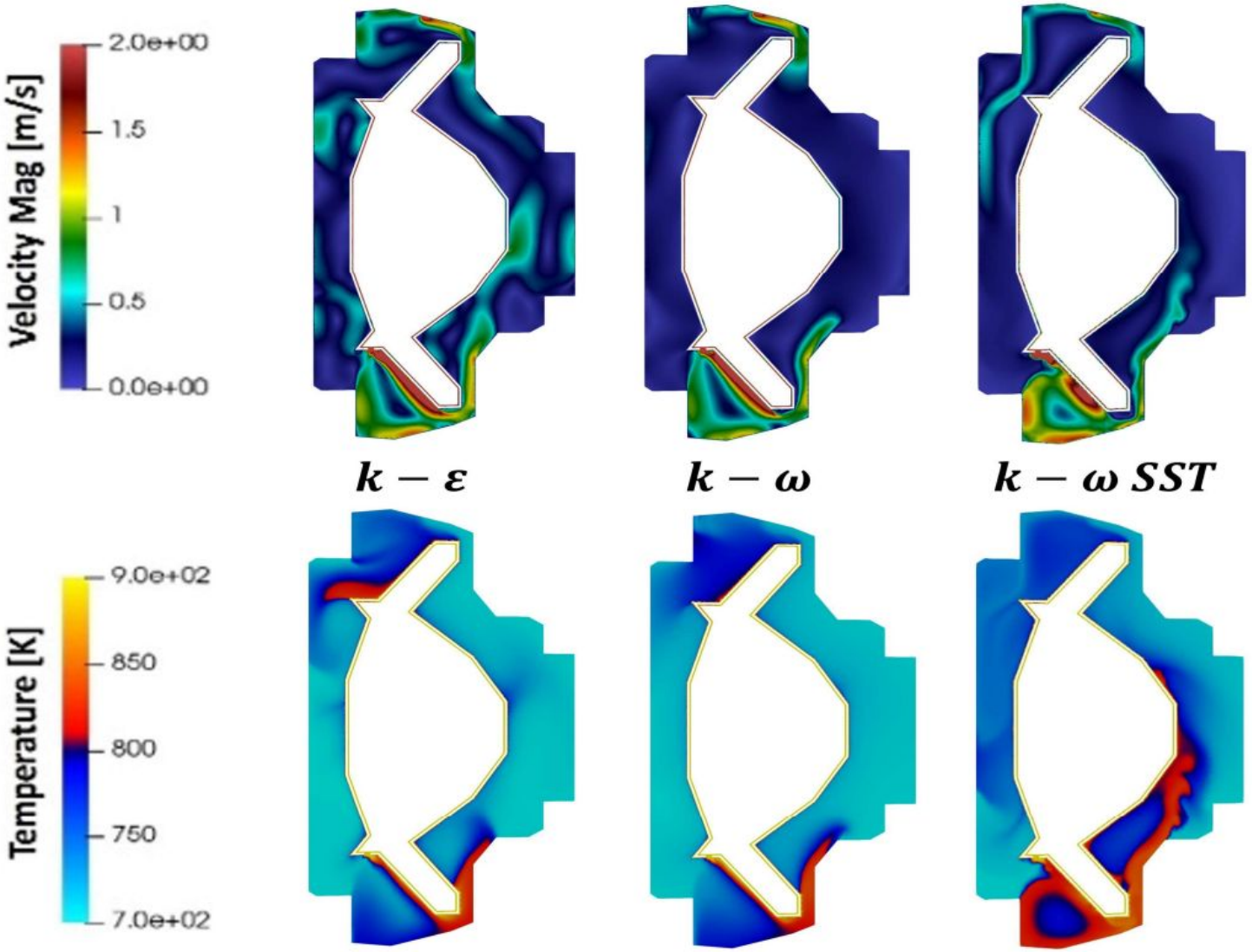
Reduced order CHT simulations o full vacuum vessel



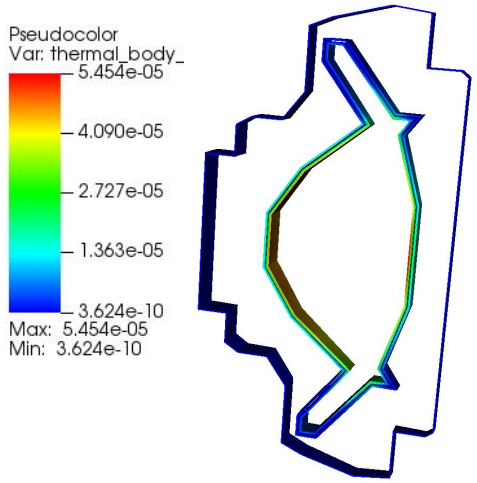


OAK RIDGE National Laboratory

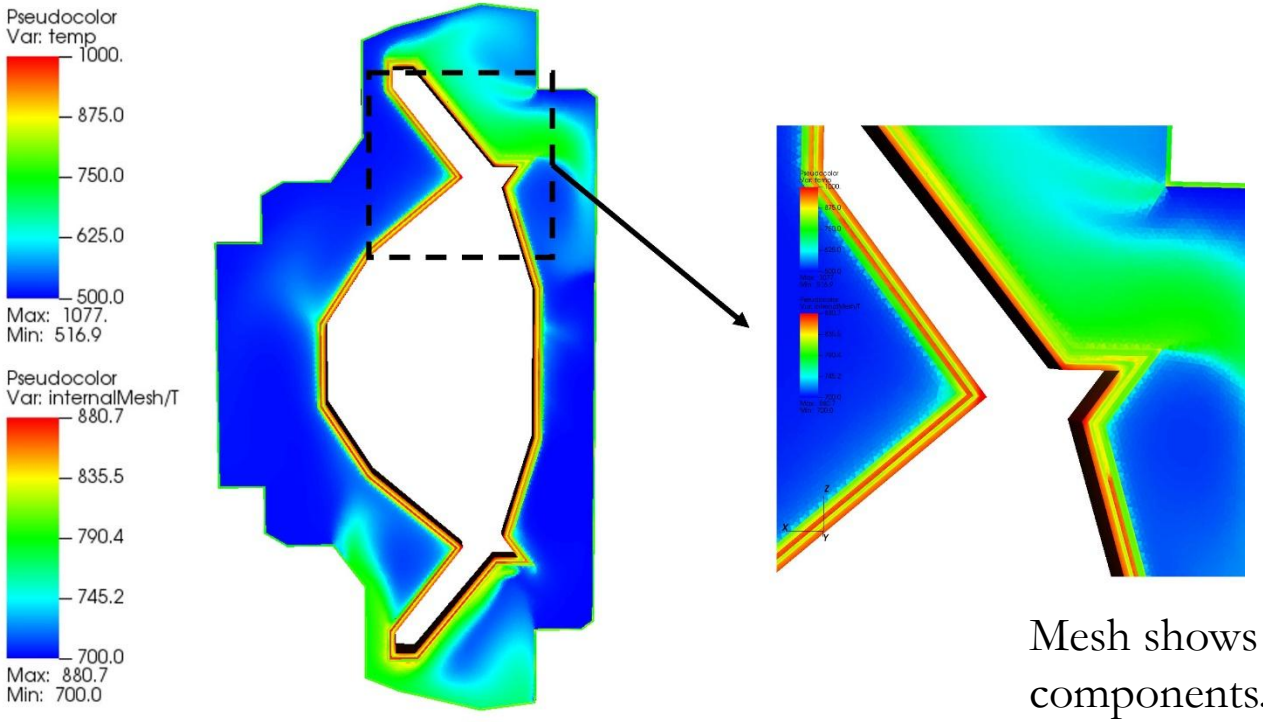
Sensitivity of simulations to turbulence models



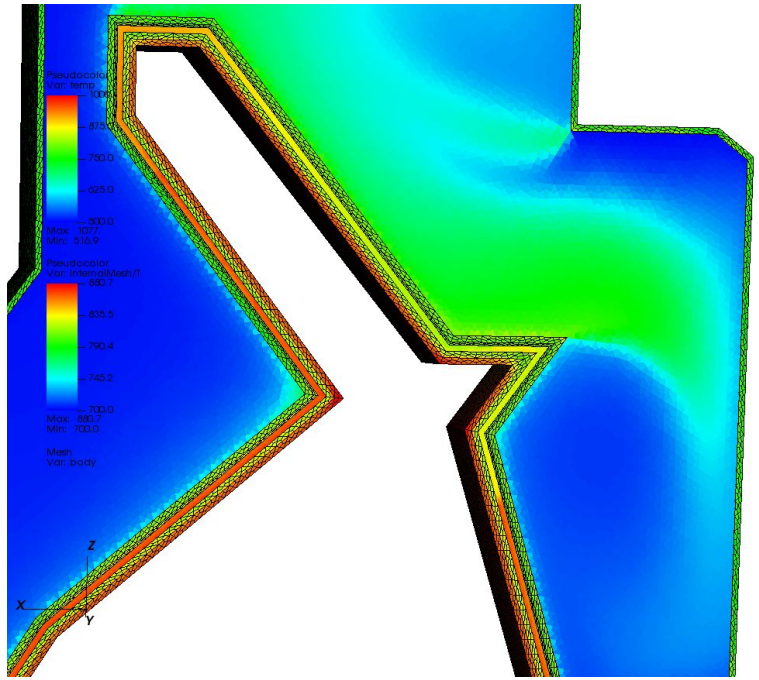
Conjugate heat transfer analysis of 3D ARC vacuum vessel and blanket



Heat deposition from neutronics on the solid components



Mesh shows solid components. Contours show continuity of temperature through fluid-solid interfaces



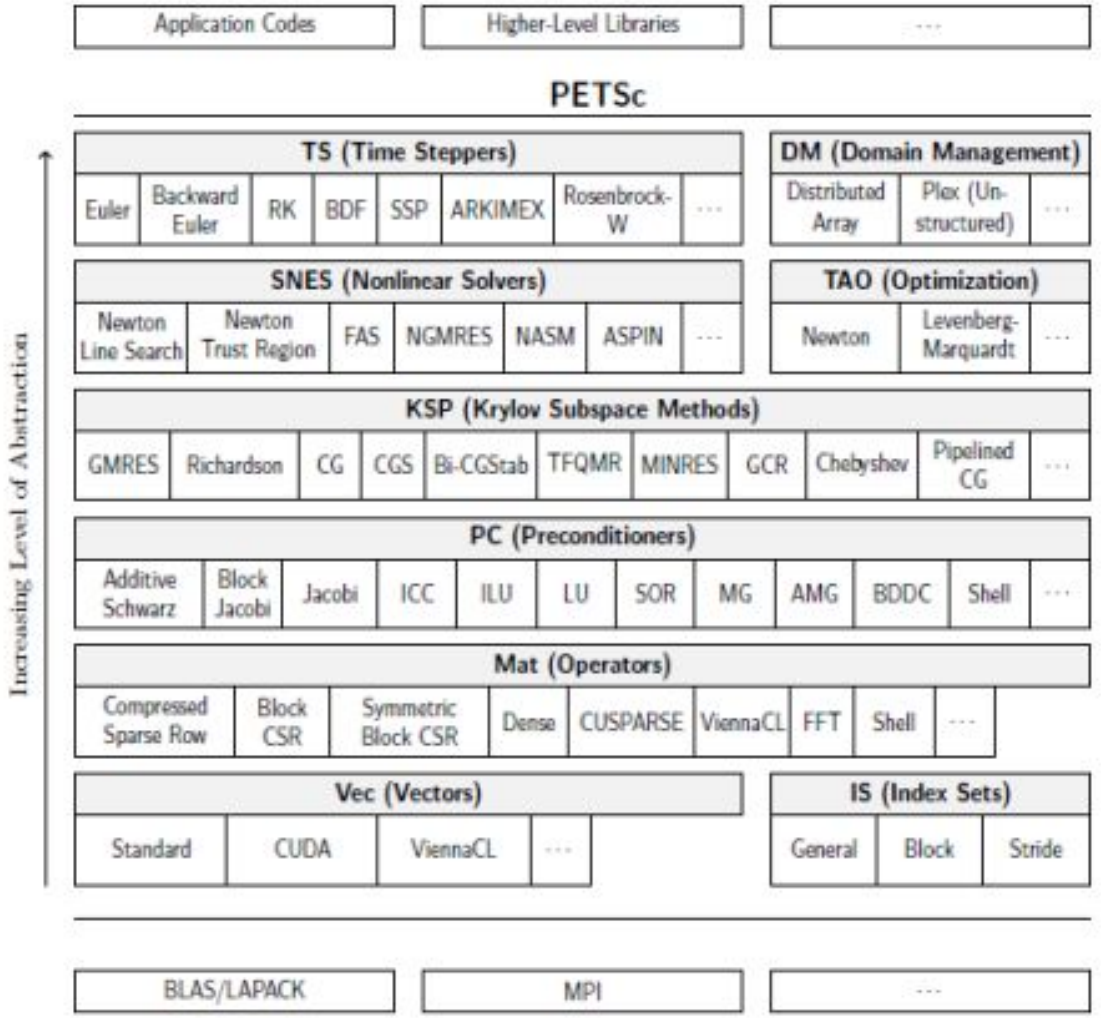
OAK RIDGE National Laboratory

Need for exascale simulations

- As we add more physics, computational time increases.
- Exploration of design space, parametric studies, optimization of design require ensemble simulations.
- OpenFOAM simulations are time consuming – each simulation can take days (up to a week).
- Fusion blankets require more physics such as MHD which is poorly understood.
- For quick time to solution, GPU acceleration is investigated – however, OpenFOAM solvers are native to CPUs. One approach is to use other linear solvers such as PETSc and leverage architecture agnostic models such as Kokkos.

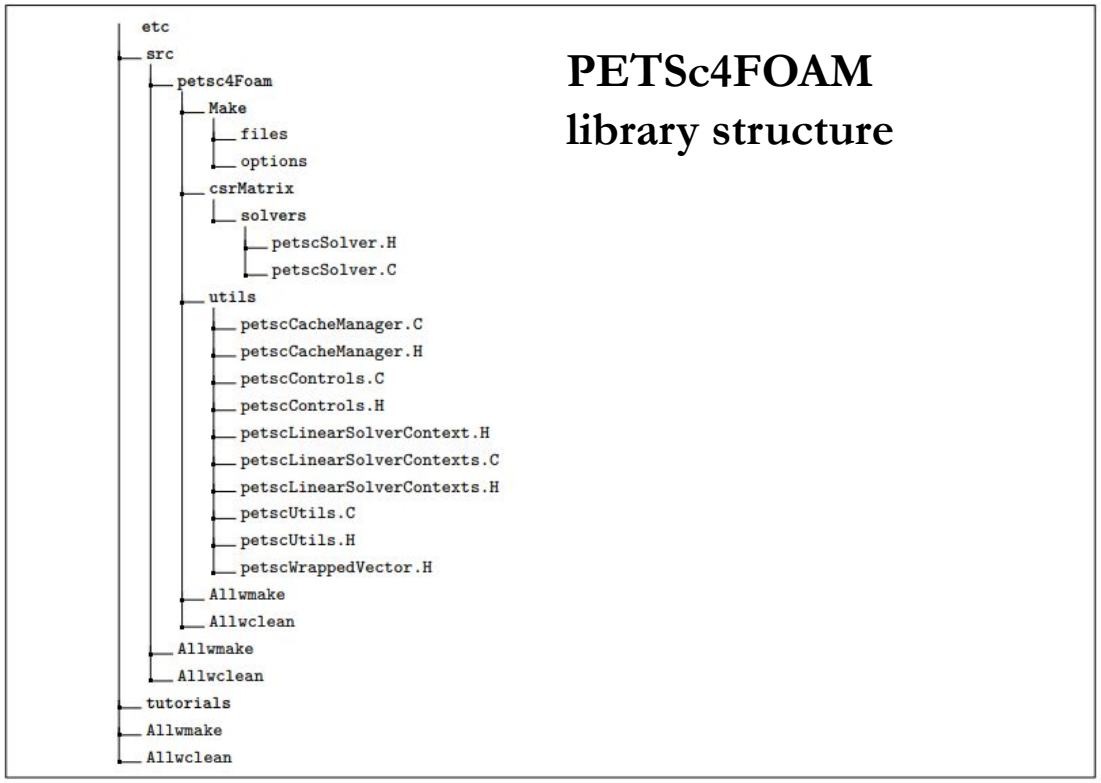
Quick introduction to PETSc

- Exascale HPC is moving towards millions of cores with high-density cores
- PETSc includes a large suite of parallel linear, nonlinear equation solvers and ODE integrators that are easily used in application codes written in C, C++, Fortran and Python
- Offers simple parallel matrix and vector assembly routines that allow the overlap of communication and computation with support for MPI, GPU (through CUDA or OpenCL) and hybrid MPI-GPU
- PETSc modules includes – index sets, vectors, matrices, Krylov subspace methods, preconditioners, multigrid solvers, block solvers and sparse direct solvers



petsc4FOAM framework

- Execution flow of OpenFOAM is unchanged except for the solving part – the final lduMatrix in OpenFOAM (obtained from discretization) is converted to a PETSc csrMatrix.
  Matrix conversion introduces about 2% overhead but allows no intrusion of the OpenFOAM source code
- The csrMatrix is operated on by PETSc Krylov Subspace solvers and a preconditioner object
- The PETSc4FOAM library is decoupled from the OpenFOAM build



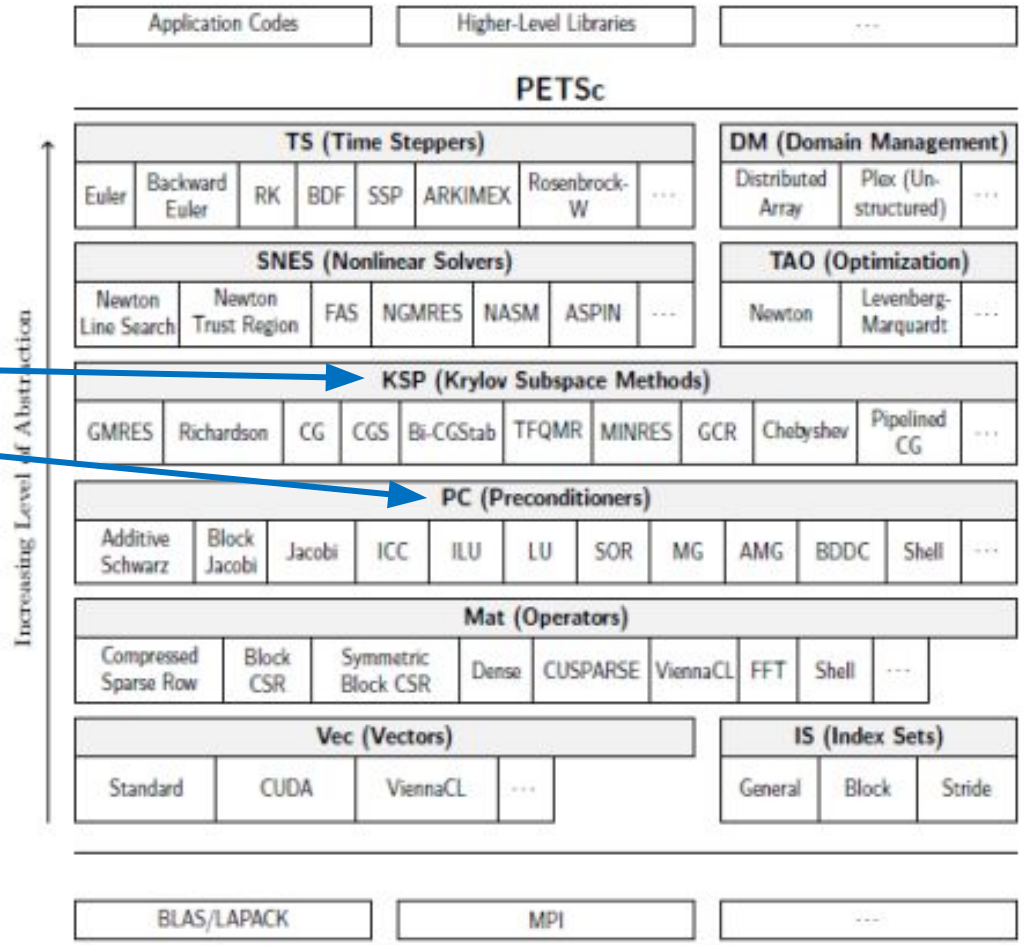
PETSc4FOAM library structure

Changes in OpenFOAM input file *fvSolution*

```
solvers
{
    p
    {
        solver          petsc;
        preconditioner  petsc;

        petsc
        {
            options
            {
                ksp_type cg;
                pc_type bjacobi;
                sub_pc_type icc;
                ksp_cg_single_reduction true;
                mat_type mpiaijmkl;
            }

            caching
            {
                matrixCaching
                {
                    update always;
                }

                preconditionerCaching
                {
                    update always;
                }
            }

        tolerance       1.e-04;
        relTol          0;
        maxIter         3000;
    }

    pFinal
    {
        $p;
        relTol          0;
    }

    U
    {
        solver          PBiCGStab;
        preconditioner  DILU;
        tolerance       0;
        relTol          0;
        maxIter         5;
    }
}
```

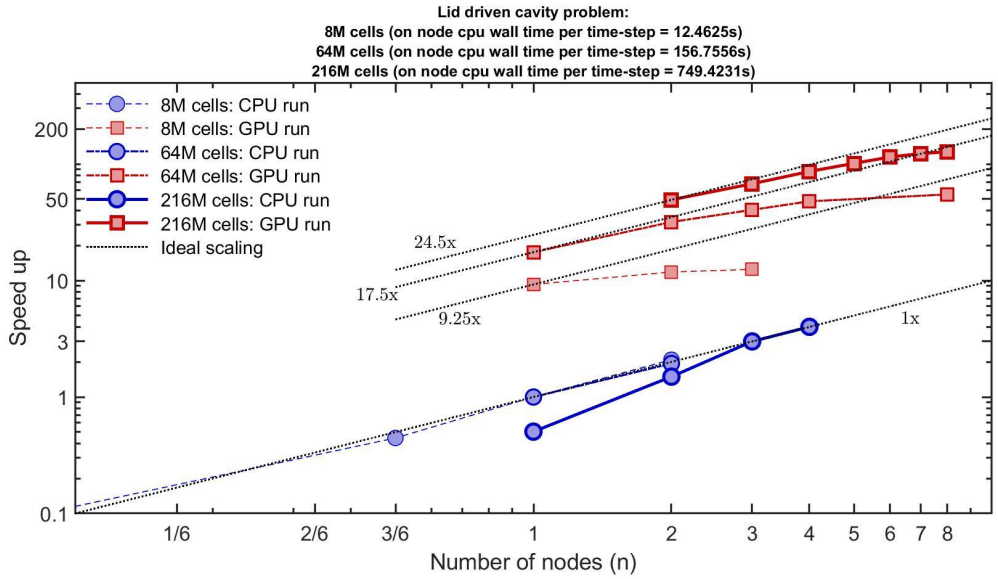
PETSc options in fvSolution

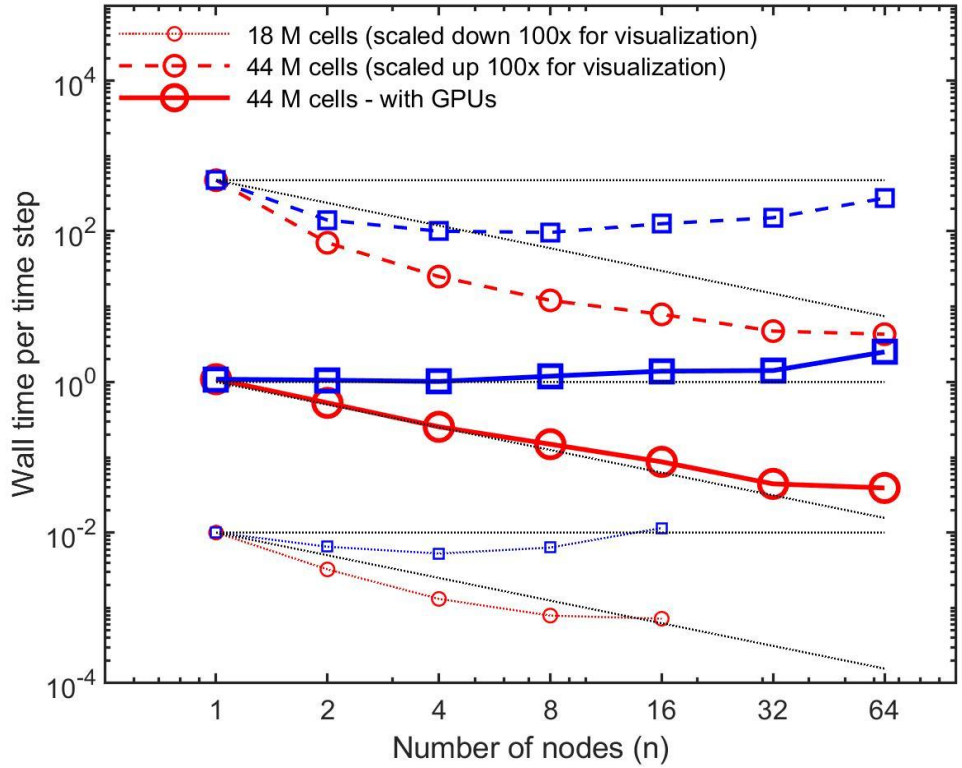
```
petsc
{
    options
    {
        ksp_type cg;
        pc_type bjacobi;
        sub_pc_type icc;
        ksp_cg_single_reduction true;
        mat_type mpiaijmkl;
    }

    caching
    {
        matrixCaching
        {
            update always;
        }

        preconditionerCaching
        {
            update always;
        }
    }
```



MatTypes	Description
MATAIJ	"*aij*" type for sparse matrices
MATSBAIJ	"*sbaij*" type for symmetric block sparse matrices
MATMPIAIJ	"*mpiaij*" type for parallel sparse matrices
MATMPIAIJMKL	type for parallel sparse matrices with Intel MKL support
MATMPISBIAJ	"*mpisbaij*" type for distributed symmetric sparse block matrices
MATHYPRE	"*hypre*" type sequential and parallel sparse matrices based on the hypre *ij* interface
MATSELL	"*sell*" type for sparse parallel matrix in SELL format

AMGx solvers can be used for Nvidia GPUs





Scaling of OpenFOAM for full 3D ARC simulations using CFD+MHD on Summit.
Wall clock time per time step on 1 node with 18M cells = 3.03 s
Wall clock time per time step on 1 node with 44M cells: CPUs only = 14.48s & CPUs+GPUs = 3.3s

petscKokkos4Foam – developed in the INCITE project
(https://code.ornl.gov/2as/petsckokkos4foam)

Step 1: Build OpenFOAM from source using Frontier's MPICH libraries (note: you will need this to run OpenFOAM on Frontier CPUs as well).

Step 2: Install PETSc with Kokkos and HIP – PETSc needs to be configured with the appropriate gfx90a accelerators of Frontier

Step 3: Install petsc4Foam

Step 4: Build the petsc-kokkos interface to OpenFOAM

Step 5: Run simulations by calling the mataijkokkos libraries from PETSc and use Hypre preconditioning

Changes in OpenFOAM configuration

- Building the different libraries does not interfere with the OpenFOAM installation, in fact, its modular nature allows easy update when new versions are released
- Running OpenFOAM requires no changes in the pre or post processing or even the case setup besides for changes in the solver selection files listed below
- *system/fvSolution* will need to be changed to enable the AmgX solver path
  - Add in the options, caching selections, tolerances and maxIterations for AmgX
- For each linear solver that is being accelerated with AmgX a corresponding configuration file is necessary as *system/amgx<system>Options*
  - *<system>* takes on the linear system being solved, for e.g., *p* for pressure, *Ux* for first velocity component, *Uy* for second velocity component and so on
  - Solver details, preconditioner details, convergence and iterations criterion

Debugging tools

Particularly useful when setting up existing code for GPU acceleration to identify where there might be issues.

Valgrind – Tool for debugging memory leaks

- Checks for memory leaks
- Checks for invalid memory access – In our case, was particularly useful to identify CPU-GPU communications.

Rocgdb – AMD source level debugger for linux which enables debugging on ROCm

- Identifies architecture dependent issues – Informed us of need to explicitly pass the GTL to use at run-time.
- Mismatch in libraries used by different codes can be tracked – Guided choice of some library versions (amd-mixed, cpe) we needed.

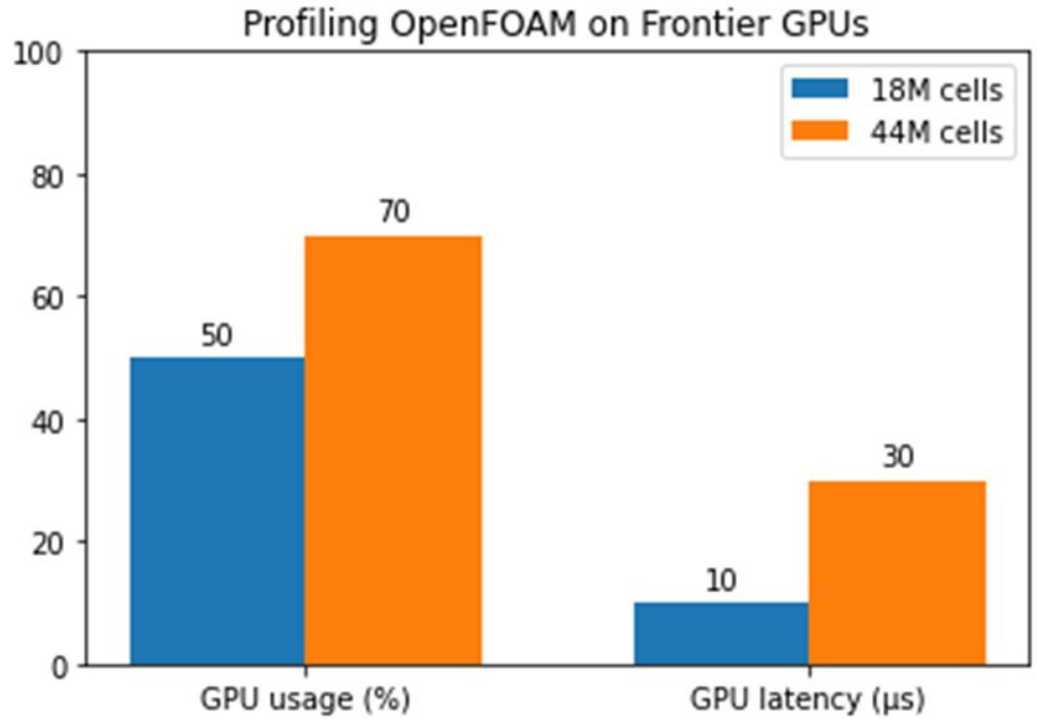
Profiling tools

rocProf – https://github.com/ROCm/rocprofiler

- Allows timing different function calls in the code
- Can be used to estimate the time spent by CPU and GPU in computation and memory transfer
- Useful to assess latencies

Simple kernel timer (Kokkos) – https://github.com/kokkos/kokkos-tools/wiki/SimpleKernelTimer

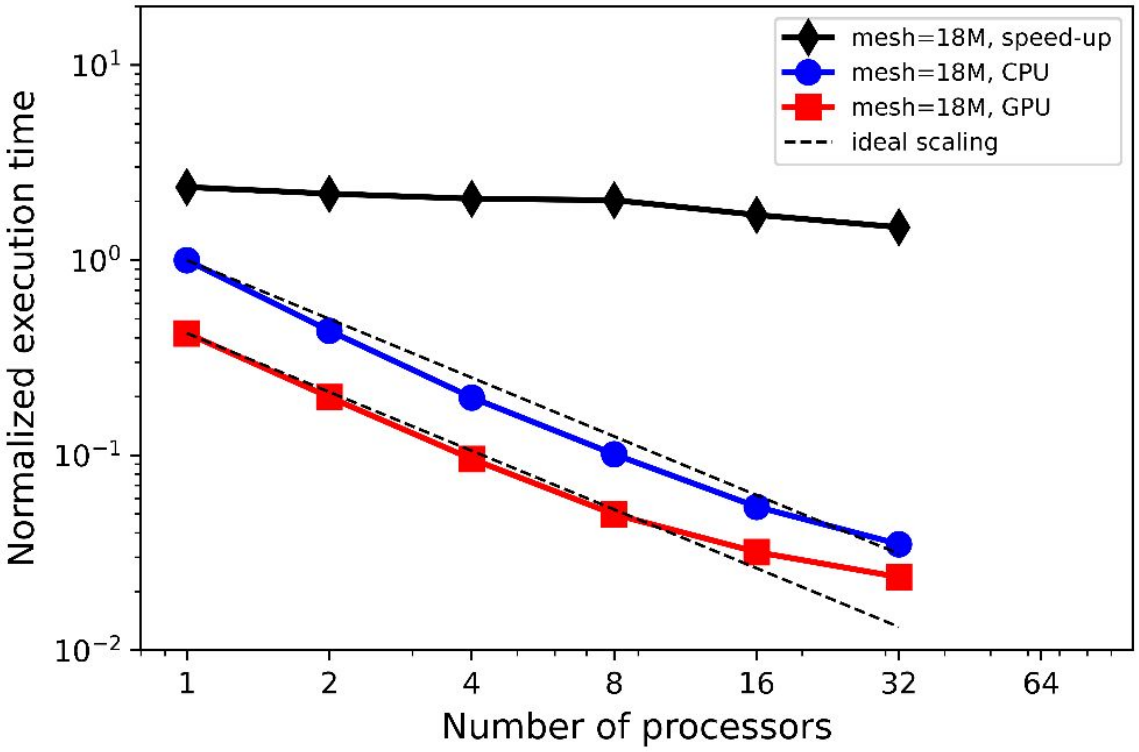
- Gives details of kernel calls by Kokkos
- Useful for identifying time taken by GPU kernel calls

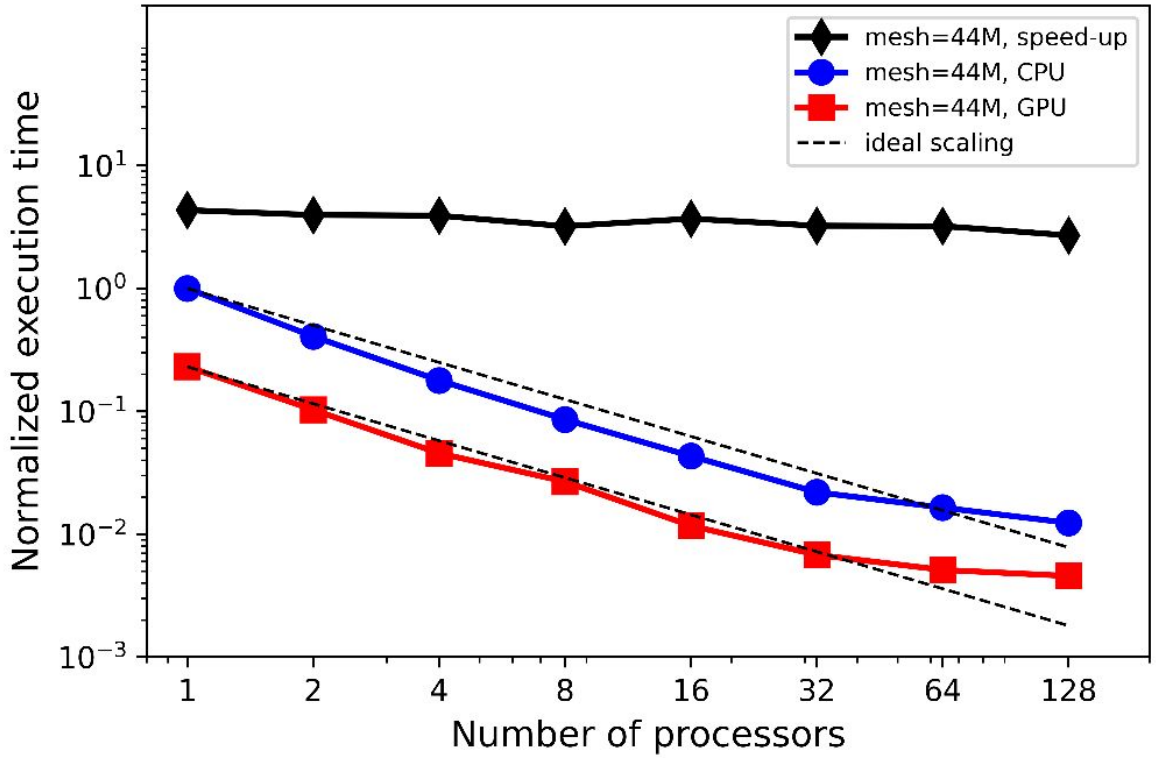


Goals of profiling: -

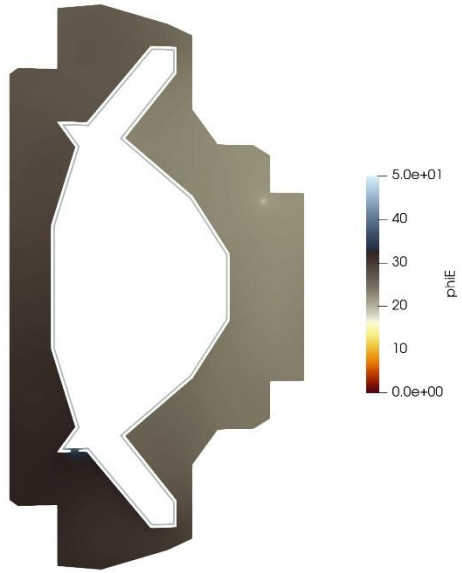
- Increase compute time spent on GPUs
- Ensure proper domain decomposition to leverage high GPU throughput

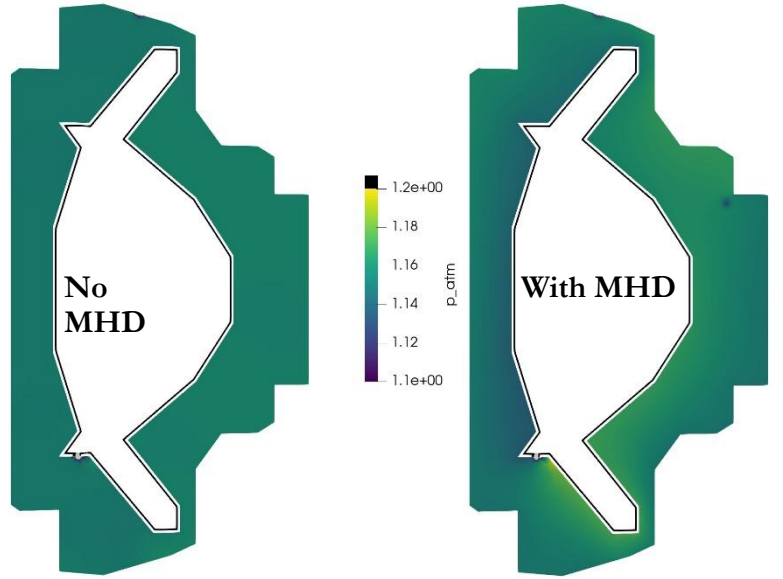
Scaling of OpenFOAM on Frontier for fusion blanket simulations





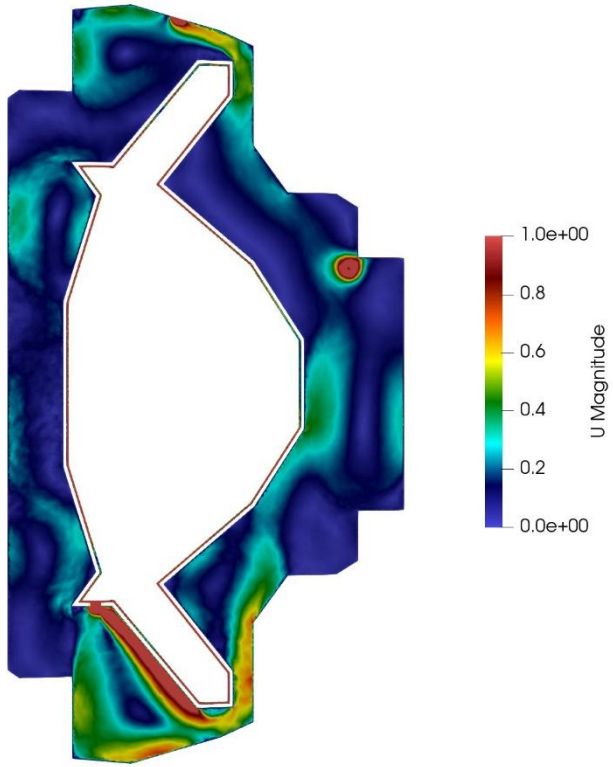
MHD effects can be investigated in reasonable time frames

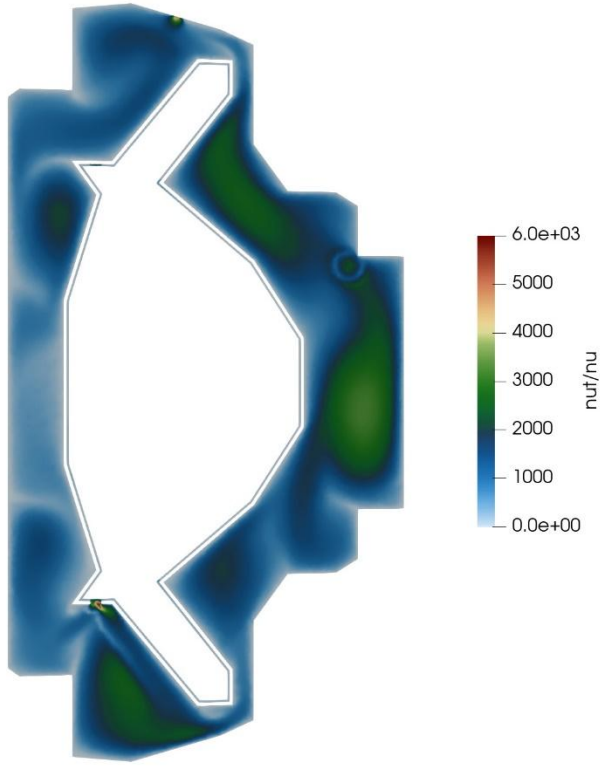


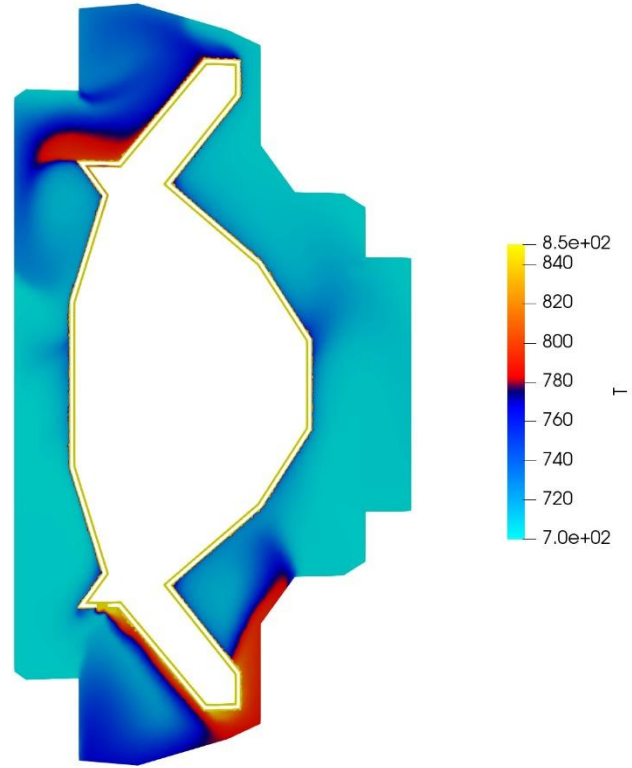


Approximations:-

- Uniform magnetic field of 5 T
- Toroidal direction (into plane of paper)
- Constant FLiBe electrical conductivity of 400 S/m







Next steps

Improving performance on GPUs

- Hypre preconditioners can not use GPU aware MPI for multi-node simulations yet.
- Further profiling analyses to identify bottlenecks of simulation performance.

Enhancing physical understanding of MHD effects

- Ensemble simulations to test sensitivity of flow parameters (Re, Ha, Pr, Gr) on MHD.
- Generate data sets to train AI/ML models for turbulent MHD simulations.

Inclusion of additional physical effects

- Work underway to couple with neutronics (Shift) on GPUs
- Plans to also include structural mechanics to enable full FERMI suite running on Frontier

Explore large-scale data handling

- Investigate parallel meshing and post-processing
- Interface with the IRI program

THANK YOU