Nuclear Fuel Performance

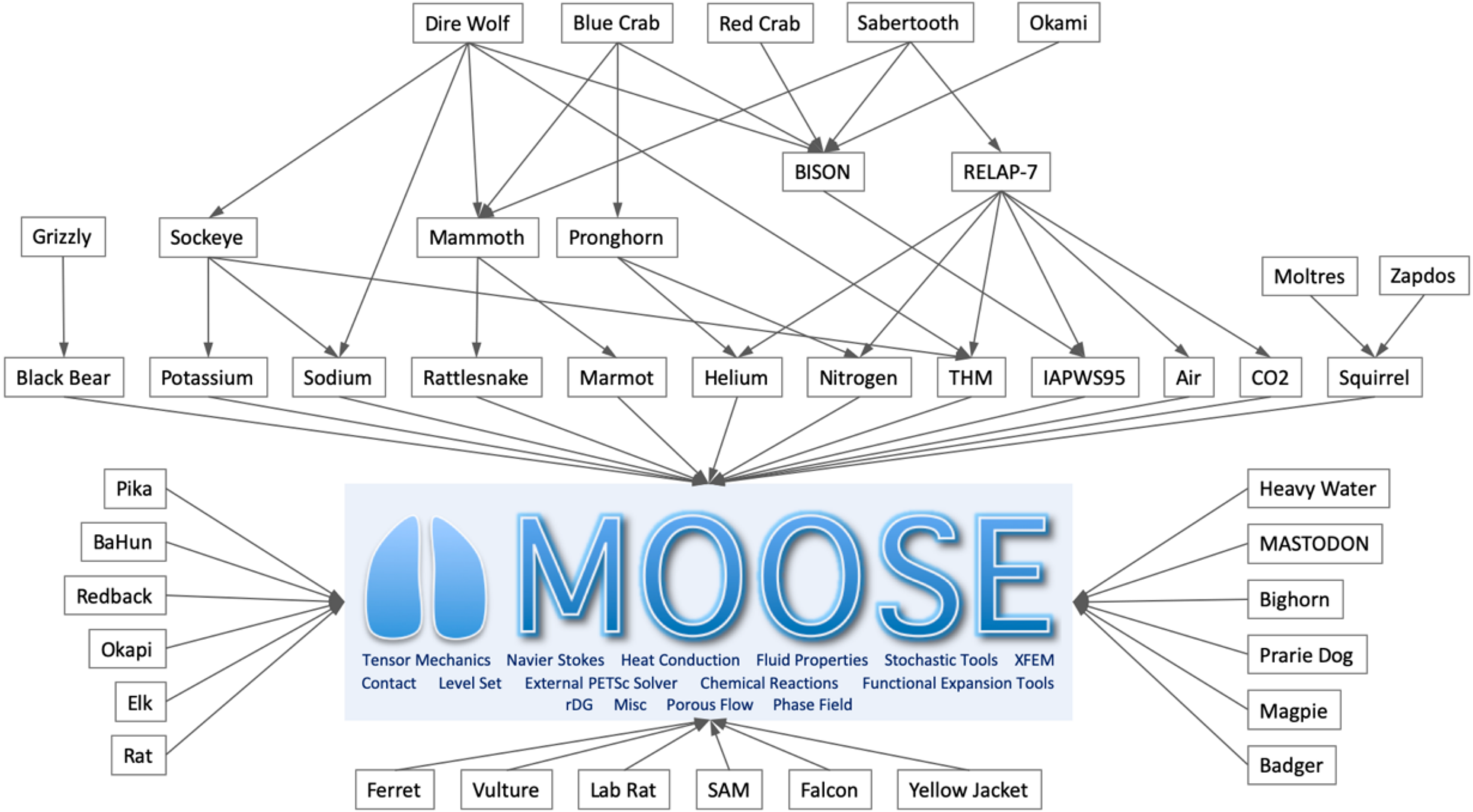
NE-591-010

Spring 2021

MOOSE– Massively Object-Oriented Simulation Environment

the multiphysics framework

https://mooseframework.inl.gov/


Dire Wolf
Blue Crab
Red Crab
Sabertooth
Okami
BISON
RELAP-7
Grizzly
Sockeye
Mammoth
Pronghorn
Moltres
Zapdos
Black Bear
Potassium
Sodium
Rattlesnake
Marmot
Helium
Nitrogen
THM
IAPWS95
Air
CO2
Squirrel
Pika
BaHun
Redback
Okapi
Elk
Rat
MOOSE
Tensor Mechanics
Navier Stokes
Heat Conduction
Fluid Properties
Stochastic Tools
XFEM
Contact
Level Set
External PETSc Solver
Chemical Reactions
Functional Expansion Tools
rDG
Misc
Porous Flow
Phase Field
Heavy Water
MASTODON
Bighorn
Prarie Dog
Magpie
Badger
Ferret
Vulture
Lab Rat
SAM
Falcon
Yellow Jacket

MOOSE Physics Modules

Chemical Reactions
Contact
External PETSc Solver
Fluid Properties
Function Expansion Tools
Heat Conduction
Level Set
Navier Stokes
Phase Field
Porous Flow
rDG
Stochastic Tools
Tensor (solid) Mechanics
XFEM
Shallow Water (work in progress)
Ray Tracing (work in progress)

Governing Equations

Conservation of Mass:

$$\nabla \cdot \bar{u} = 0 \tag{1}$$

Conservation of Energy:

$$C\left(\frac{\partial T}{\partial t} + \epsilon \bar{u} \cdot \nabla T\right) - \nabla \cdot k \nabla T = 0 \tag{2}$$

Darcy's Law:

$$\bar{u} = -\frac{\mathbf{K}}{\mu}(\nabla p - \rho \bar{g}) \tag{3}$$

where $\bar{u}$ is the fluid velocity, ϵ is porosity, $\mathbf{K}$ is the permeability tensor, μ is fluid viscosity, p is the pressure, ρ is the density, $\bar{g}$ is the gravity vector, and T is the temperature.

Input Files

All capabilities of MOOSE, modules, and your application are compiled into a single executable. An input file is used define which capabilities are used to perform a simulation.

MOOSE uses the "hierarchical input text" (hit) format.

```
[Kernels]
  [diffusion]
    type = ADDiffusion # Laplacian operator using automatic differentiation
    variable = pressure # Operate on the "pressure" variable from above
  []
[]
```

Input File

A basic MOOSE input file requires six parts, each of which will be covered in greater detail later.

- `[Mesh]`: Define the geometry of the domain
- `[Variables]`: Define the unknown(s) of the problem
- `[Kernels]`: Define the equation(s) to solve
- `[BCs]`: Define the boundary condition(s) of the problem
- `[Executioner]`: Define how the problem will solve
- `[Outputs]`: Define how the solution will be written

```
[Mesh]
  type = GeneratedMesh # Can generate simple lines, rectangles and rectangular prisms
  dim = 2              # Dimension of the mesh
  nx = 100             # Number of elements in the x direction
  ny = 10              # Number of elements in the y direction
  xmax = 0.304         # Length of test chamber
  ymax = 0.0257        # Test chamber radius
[]

[Variables]
  [pressure]
    # Adds a Linear Lagrange variable by default
  []
[]

[Kernels]
  [diffusion]
    type = ADDiffusion  # Laplacian operator using automatic differentiation
    variable = pressure # Operate on the "pressure" variable from above
  []
[]

[BCs]
  [inlet]
    type = DirichletBC  # Simple u=value BC
    variable = pressure # Variable to be set
```

Run Example via Command Line

```
cd ~/projects/moose/tutorials/darcy-thermo_mech/step01_diffusion
make -j 12 # use number of processors for you system
cd problems
../darcy_thermo_mech-opt -i step1.i
```

Mesh

`FileMesh` is the default type:

```
[Mesh]
  file = sample.msh
  dim = 3
[]
```

MOOSE supports reading and writing a large number of formats and could be extended to read more.

Built-in mesh generation is implemented for lines, rectangles, and rectangular prisms

```
[Mesh]
  type = GeneratedMesh
  dim = 2
  xmin = -1
  xmax = 1
  ymin = -1
  ymax = 1
  nx = 2
  ny = 2
  elem_type = QUAD9
[]
```

The sides are named in a logical way and are numbered:

- 1D: left = 0, right = 1
- 2D: bottom = 0, right = 1, top = 2, left = 3
- 3D: back = 0, bottom = 1, right = 2, top = 3, left = 4, front = 5

Variables

The `Variables` block within an input file is utilized to define the unknowns within a system of partial differential equations. These unknowns are often referred to as nonlinear variables within documentation. The nonlinear variables defined within this block are used by Kernel objects to define the equations for a simulation.

```
[Variables]
  [convected]
    order = FIRST
    family = LAGRANGE
  []
[]

[Kernels]
  [diff]
    type = Diffusion
    variable = convected
  []

  [conv]
    type = ExampleConvection
    variable = convected
    velocity = '0.0 0.0 1.0'
  []
[]
```

Kernel

A "Kernel" is a piece of physics. It can represent one or more operators or terms in the weak form of a partial differential equation. With all terms on the left-hand-side, their sum is referred to as the "residual". The residual is evaluated at several integration quadrature points over the problem domain. To implement your own physics in MOOSE, you create your own kernel by subclassing the MOOSE `Kernel` class.

`ADHeatConduction` is the implementation of the heat diffusion equation in HeatConduction within the framework of automatic differentiation. The `ADHeatConduction` kernel implements the heat equation given by Fourier's Law where The heat flux is given as

$$\mathbf{q} = -k\nabla T,$$

where k denotes the thermal conductivity of the material. k can either be an `ADMaterial` or traditional `Material`.

Materials

The material system operates by creating a producer/consumer relationship among objects

- `Material` objects **produce** properties.
- Other MOOSE objects (including materials) **consume** these properties.

```
[Materials]
  [block_1]
    type = OutputTestMaterial
    block = 1
    output_properties = 'real_property tensor_property'
    outputs = exodus
    variable = u
  []
  [block_2]
    type = OutputTestMaterial
    block = 2
    output_properties = 'vector_property tensor_property'
    outputs = exodus
    variable = u
  []
[]

[Outputs]
  exodus = true
[]
```

GenericConstantMaterials

GenericConstantMaterial is a simple way to define constant material properties.

Two input parameters are provided using "list" syntax common to MOOSE:

```
prop_names  = 'conductivity density'
prop_values = '0.01         200'
```

Boundary Conditions

$$
\begin{aligned}
-\nabla^2 u &= f && \in \Omega \\
u &= g && \in \partial\Omega_D \\
\frac{\partial u}{\partial n} &= h && \in \partial\Omega_N,
\end{aligned}
$$

DirichletBC

Imposes the essential boundary condition $u = g$, where g is a constant, controllable value.

NeumannBC

Imposes the integrated boundary condition $\frac{\partial u}{\partial n} = h$, where h is a constant, controllable value.

MOOSE Solve Types

The solve type is specified in the `[Executioner]` block within the input file:

```
[Executioner]
  solve_type = PJFNK
```

Available options include:

- **PJFNK**: Preconditioned Jacobian Free Newton Krylov (default)
- **JFNK**: Jacobian Free Newton Krylov
- **NEWTON**: Performs solve using exact Jacobian for preconditioning
- **FD**: PETSc computes terms using a finite difference method (debug)

Outputs

The output system is designed to be just like any other system in MOOSE: modular and expandable.

It is possible to create multiple output objects for outputting:

- at specific time or timestep intervals,
- custom subsets of variables, and
- to various file types.

There exists a short-cut syntax for common output types as well as common parameters.

```
[Outputs]
  interval = 10
  exodus = true
  [all]
    type = Exodus
    interval = 1 # overrides interval from top-level
  []
[]
```

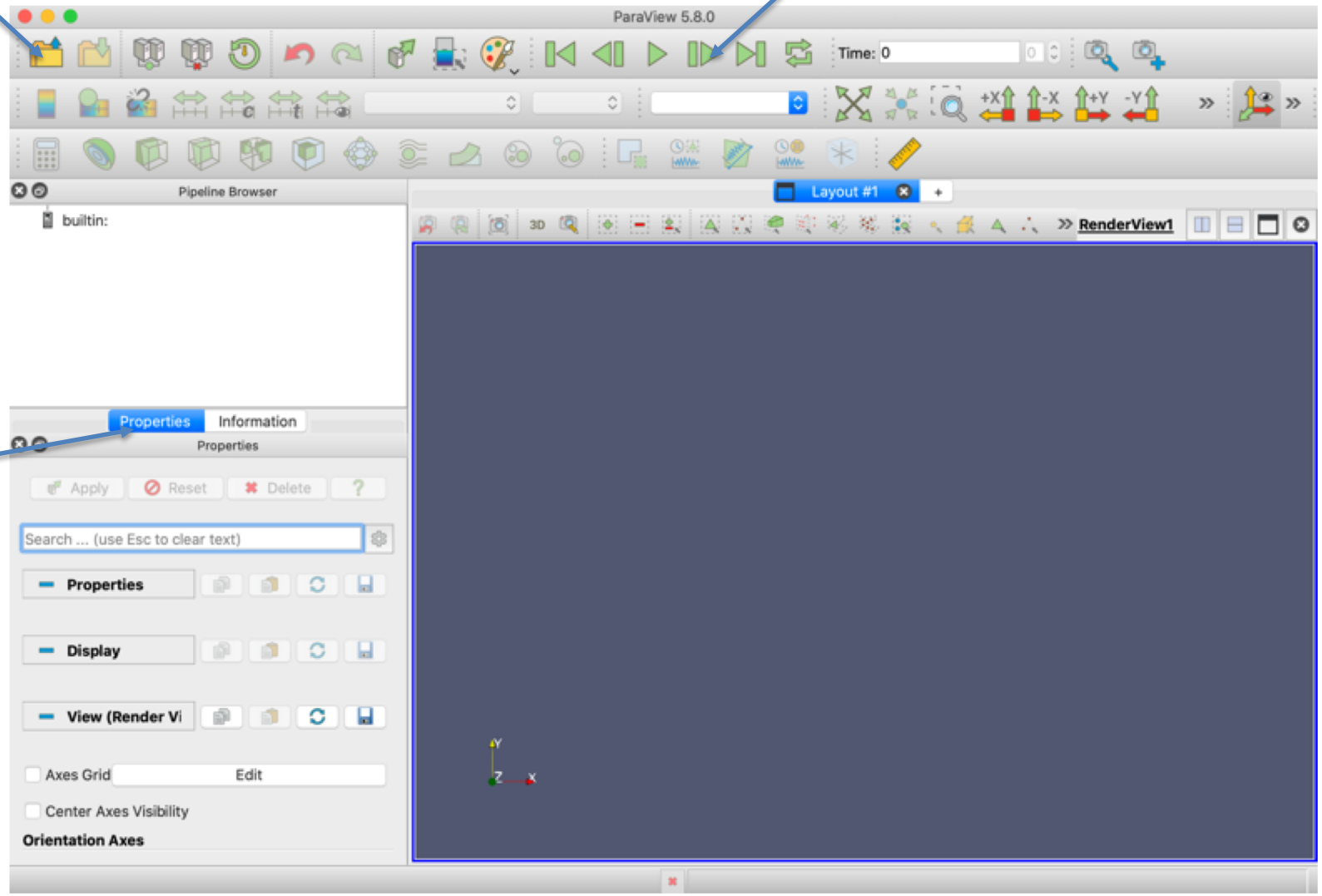
Visualization

- Two ways to visualize
  - Peacock: MOOSE GUI
  - Paraview: External software program that can read exodus files
- I recommend downloading paraview
  - It is free and open source
  - Very flexible
  - Less buggy than peacock
- But, if you try peacock, and it works for you, great!

Paraview

Open new

Move forward in time



Select Properties

MOOSE C++ Fundamentals

- Will not cover here, if you feel like you need some background, go to below site, where they have some basics

- https://mooseframework.org/workshop/#/10

Tutorials/Examples

- A stepped example/tutorial lives in:
  - moose/tutorials/darcy_thermo_mech
- This tutorial has 11 steps where they add complexity to a moose input file/physics problem
- This is a VERY valuable exercise to work through
- Tip : ./mooseapp-opt --dump > syntax.out creates a file “syntax.out” with all of the moose syntax and usage
- Better tip: mooseframework.org is a great resource with everything documented

Access to MOOSE

- Everyone has access to the RDFMG cluster
- Can utilize moose on your local machine, or on cluster
- If off campus, need to get the NCSU VPN
  - https://oit.ncsu.edu/campus-it/campus-data-network/vpn/
- Need some kind of terminal
  - linux/mac is native, for windows need something like PuTTY, Cygwin, MobaXterm, etc.
- ssh uname@rdfmg.ne.ncsu.edu

Building MOOSE on RDFMG cluster

- mkdir projects
- cd projects
-
- git clone https://github.com/idaholab/moose.git
-
- module load GCC/8.3.0-2.32
- module load OpenMPI/4.0.1-GCC-8.3.0-2.32
- module load cmake/3.6.3
- module load PETSc/3.9.1-foss-2018a-downloaded-deps
- module load Python/3.7.4-GCCcore-8.3.0
-
- cd moose
- ./scripts/update_and_rebuild_libmesh.sh ## Note: this step takes a while
-
- cd test
- make -j 4
- ./run_tests -j 4

Making your own App and compiling

```
cd ~/projects
./moose/scripts/stork.sh YourAppName
```

Edit makefile to match right->
vi Makefile

```
cd ~/projects/YourAppName
make -j4
./run_tests -j4
```

```
################################## MODULES ####################################
# To use certain physics included with MOOSE, set variables below to
# yes as needed.  Or set ALL_MODULES to yes to turn on everything (overrides
# other set variables).

ALL_MODULES                    := no

CHEMICAL_REACTIONS             := no
CONTACT                        := no
EXTERNAL_PETSC_SOLVER          := no
FLUID_PROPERTIES               := no
FUNCTIONAL_EXPANSION_TOOLS     := no
HEAT_CONDUCTION                := yes
LEVEL_SET                      := no
MISC                           := no
NAVIER_STOKES                  := no
PHASE_FIELD                    := yes
POROUS_FLOW                    := no
RDG                            := no
RICHARDS                       := no
SOLID_MECHANICS                := yes
STOCHASTIC_TOOLS               := no
TENSOR_MECHANICS               := yes
XFEM                           := no
```

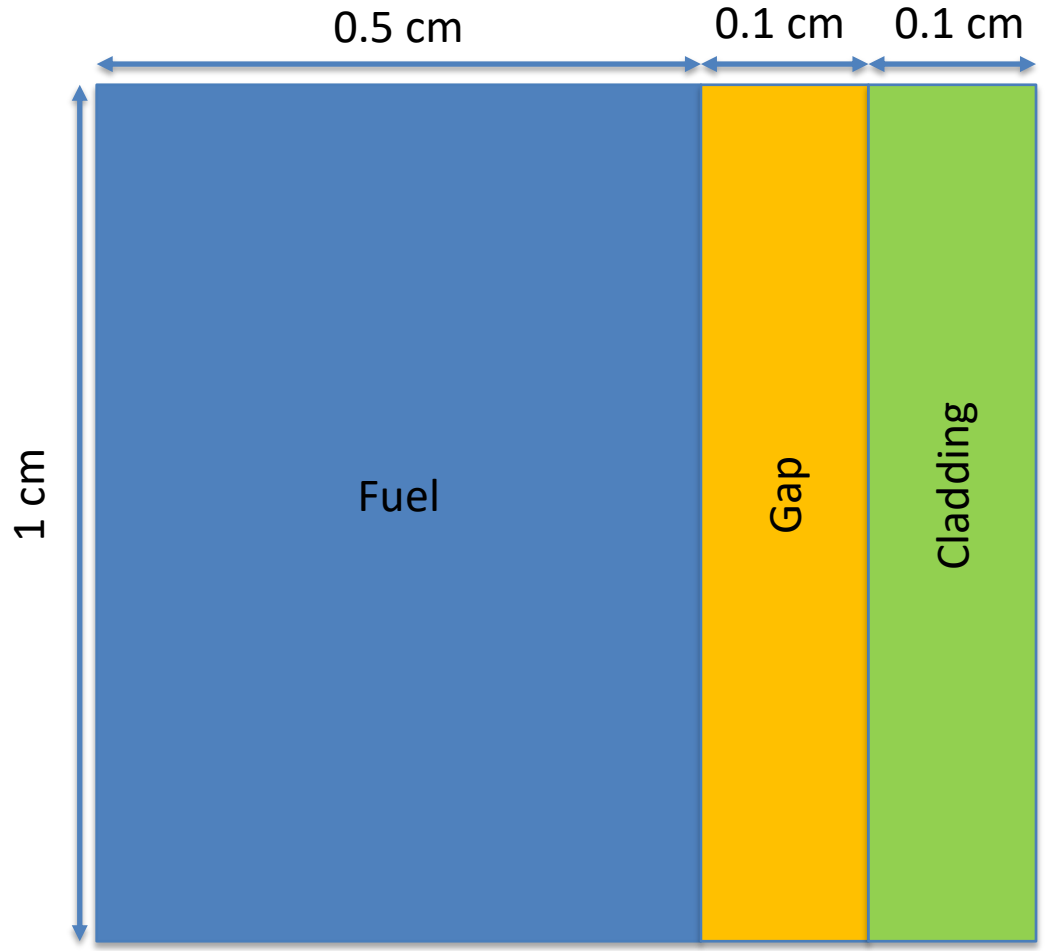
Handy bash commands

- ssh username@rdfmg.ne.ncsu.edu : allows you to get on the cluster
- sftp username@rdfmg.ne.ncsu.edu : allows you to transfer files to/from cluster
- ls : list files in a directory; pwd: print working directory; cd : change directory; cp : copy a file
- file editors: vi, vim, emacs, nano, atom
- slurm is the submission manager
  - sbatch submit.sh: submit job with submit script
  - squeue : check status of jobs
- I will provide you will a submit script that you will need to lightly edit

"1-D" MOOSE Project

- Fuel pin dimensions listed
- Assume reasonable values for thermal conductivities
  - Can assume constant k
- Outer cladding: 500 K
- Mesh: 100x100

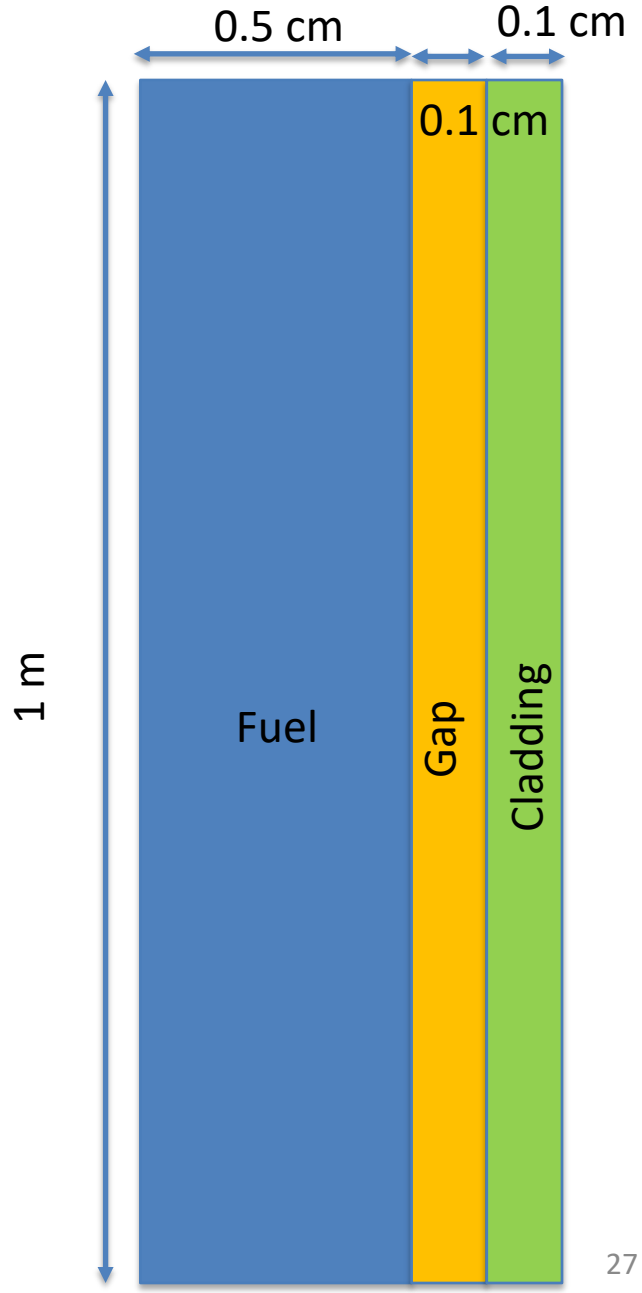
- Solve temperature profile for:
  - Steady-state: Volumetric/Areal heating rate:
    Q = 250 W/cm^2
- Solve for centerline temperature vs time
  - Transient: Volumetric/Areal heating rate
    Q = 150*(1-EXP(-0.01*time))+250 for up to t=200



2-D MOOSE Project

- Fuel pin dimensions listed
- Assume reasonable values for thermal conductivities
  - Can assume constant k
- Utilize azial T_{CO}, with reasonable flow rate, heat capacity, etc.

- Solve temperature profile for:
  - Steady-state: Volumetric/Areal heating rate: Q = 250 W/cm^2
  - @ z=0.25, z=0.5, z=1
- Solve for centerline temperature vs time
  - Transient: Volumetric/Areal heating rate
    Q = 150*(1-EXP(-0.01*time))+250 for up to t=200
  - @ z=0.25, z=0.5, z=1
- Find location of peak centerline temperature at SS, and at t=200 in transient



Submission

- Will upload four input and four output files
  - "1D" SS, "1-D" transient, 2-D SS, 2-D transient
- Will upload a written report, 5-10 pages (including figures), times new roman, 12pt, 1.5 space
- Due April 30 (last day of classes)
- This is an individual project, but some collaboration is encouraged