

# Application Code Design Constraints

- ❑ Several Axes of Complexity
  - Architecture, infrastructure, data layout, interoperability
    - Many moving parts
- ❑ Requirements
  - Maintainable code with reliable results
  - Retain code portability and performance
  - Measurable and predictable performance
- ❑ The challenges in meeting the requirements; tension between
  - Modularity and performance
  - Readable/maintainable code and portability
  - Easy adaptability to new and heterogeneous architectures and complex multi-physics capabilities - **Extensibility**

# Why Invest in Code Architecture

- ❑ Done right code architecture allows extensibility, capability addition and branching into new regimes and domains
  - Reduces barrier to entry into HPC
  - Enables multidisciplinary development
- ❑ Current Designs good for fat homogeneous nodes
  - Modular structure : APIs that allow for some plug-and-play
  - SPMD model - parallelism mostly confined to infrastructure
  - One generalized data layout, solvers conform to it as far as possible
    - Performance hit when they don't
- ❑ Why Change
  - Loop level OpenMP already breaks the separation between parallel complexity and the numerics
  - The data-structures rely on being coarse-grained, not suitable for fine-grain parallelism
  - There is the cost of bulk synchronous parallelism
    - Hides potential for parallelism
  - Auto-tuning is harder to do

# Why Invest in Mature Codes Now

- ❑ Writing from scratch is unlikely to produce reliable multi-physics codes soon
    - Years of production use eliminates both modeling and implementation bugs
    - Lessons about interoperability internalized by the developers
  - Parallel investment in new codes useful, but for the longer term
  - ❑ Consensus is emerging about techniques that will help cope with heterogeneity
    - Some form of tiling, eDSLs, runtime task scheduling
    - Code architects have to consider infrastructure design choices that will provide plugs for these techniques
- The last generation of code frameworks have converged to very similar architectures - the solutions for next generation framework design might do the same

# Considerations for Architecting

