Composable Streamed Interfaces in PETSc for Extreme-Scale Geometric Multigrid

Society for Industrial and Applied Mathematics

Johann Rudi
Argonne National Laboratory
jrudi@anl.gov

Barry Smith
Argonne National Laboratory
bsmith@mcs.anl.gov

Argonne NATIONAL LABORATORY

Motivation

- Development of parallel scalable implicit solvers and preconditioners
- Support for various models (PDEs, networks) and parallel architectures
- Careful software design with good maintainability in the future

Challenges

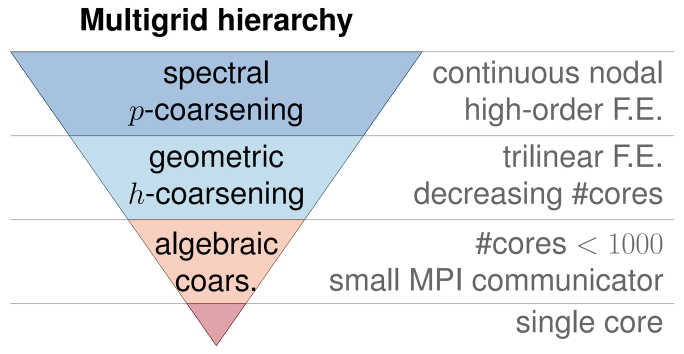
- Parallelization of multi-level methods on adaptively refinement meshes
- Memory handling for high-order finite elements and matrix-free operators
- Heterogeneous architectures demand complex memory transfers: MPI communication, GPU offloading, etc.

Hybrid Spectral–Geometric–Algebraic Multigrid

High-order finite element discretization

- High-order finite elements, nodal and modal shape functions
- Adaptive mesh refinement (AMR) to resolve localized small features
- Hexahedral elements allow exploiting the tensor product structure of shape functions to greatly reduce the number of floating point operations
- Fast, matrix-free application of stiffness and mass matrices reduce memory consumption significantly

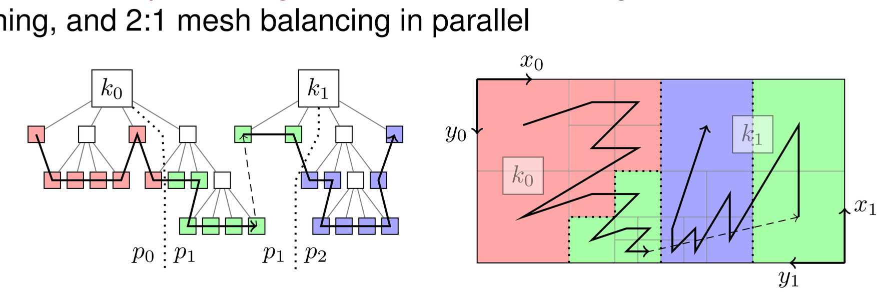
Multigrid hierarchy



- Multigrid hierarchy of nested meshes is generated from an adaptively refined octree-based mesh via spectral–geometric coarsening
- Re-discretization of PDEs at coarser levels
- Parallel repartitioning of coarser meshes for load-balancing (crucial for AMR); sufficiently coarse meshes occupy only subsets of cores
- Coarse grid solver: AMG invoked on small core counts & small communicator

Parallel forest-of-octrees AMR library [p4est.org] Scalable geometric multigrid coarsening due to:

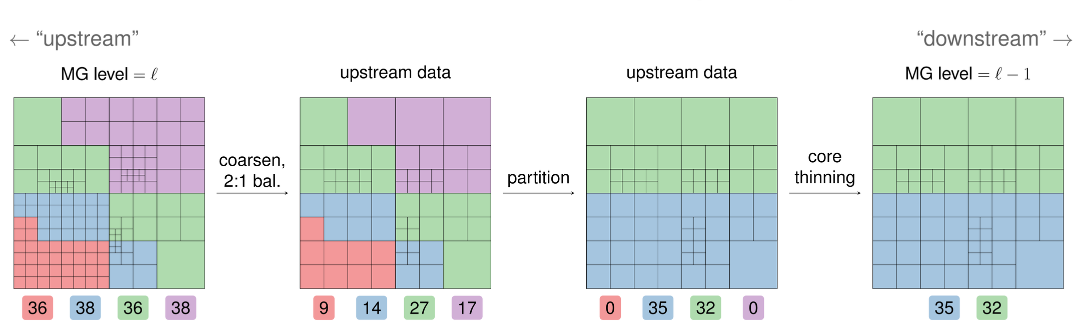
- Forest-of-octree based meshes enable fast refinement/coarsening
- Octrees and space filling curves used for fast neighbor search, mesh repartitioning, and 2:1 mesh balancing in parallel



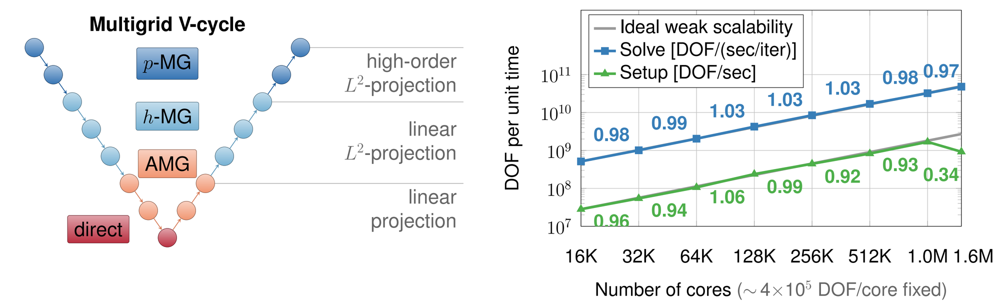
Colors depict different processor cores. (Credit: Burstedde, et al.)

Geometric coarsening (h-MG): Repartitioning & core-thinning

- Parallel repartitioning of locally refined meshes for load balancing
- Core-thinning to avoid excessive communication in multigrid cycle
- Reduced MPI communicators containing only non-empty cores
- Ensure coarsening across core boundaries: Partition families of octants/elements on same core for next coarsening sweep

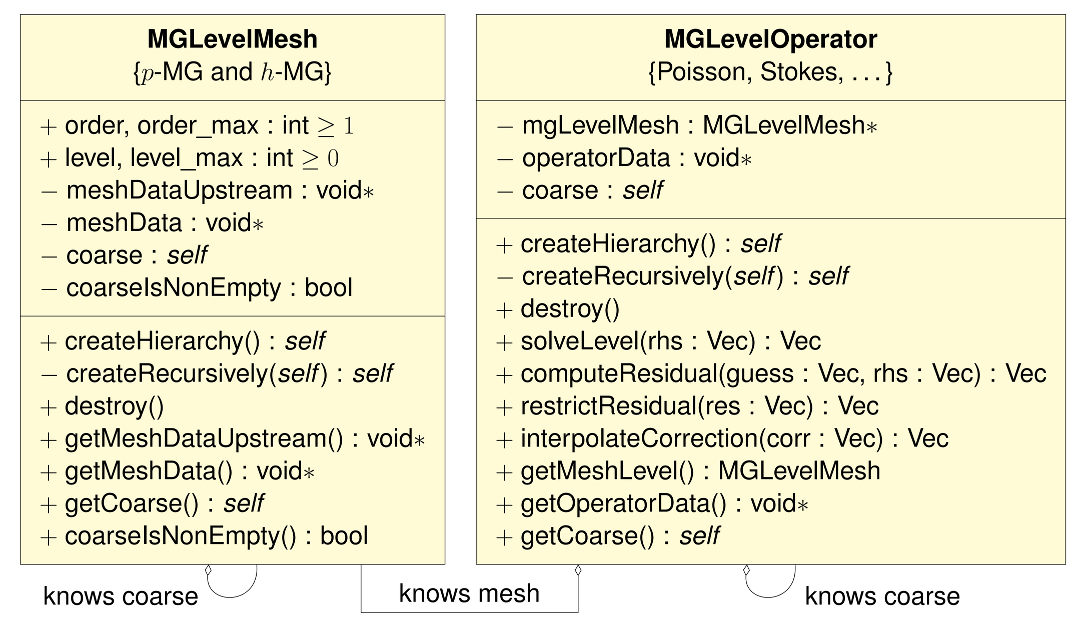


Colors depict different processor cores, numbers indicate element count on each core.
[Sundar, Biros, Burstedde, Rudi, Ghattas, Stadler, 2012]



- High-order L^2-projection onto coarser levels; restriction & interpolation are adjoints of each other in L^2-sense
- Chebyshev accelerated Jacobi smoother with tensorized matrix-free high-order stiffness apply; assembly of high-order diagonal only
- Error reduction per MG V-cycles is independent of core count
- No collective communication needed in spectral–geometric MG cycles

Software design of multigrid Separate hierarchy of meshes and operators:

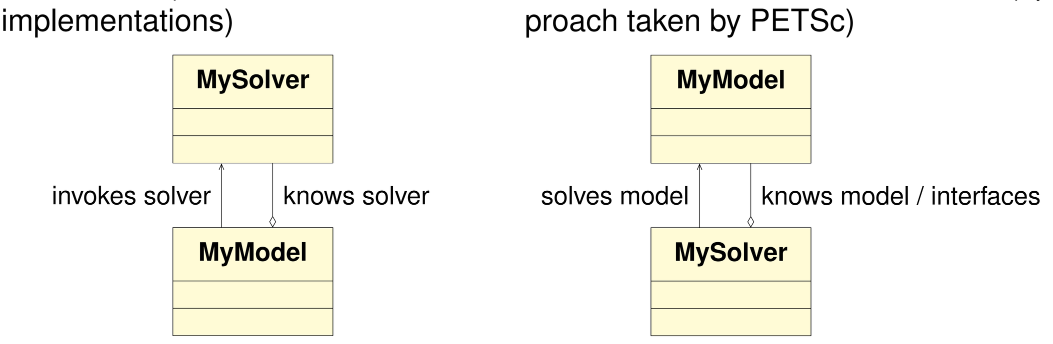


Design Patterns for DM Objects in PETSc

A high-level perspective on design patterns for model–solver interactions.

Monolithic: model knows and controls solver (common in traditional implementations)

Flexible: model is agnostic of solver; solver accesses interfaces of model (approach taken by PETSc)



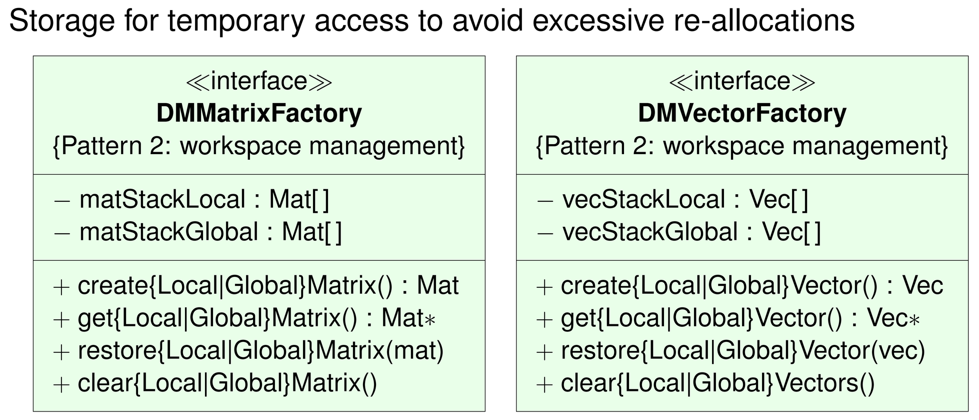
A DM in PETSc represents a model and follows five design patterns.

Pattern 1: Mapping between global and local representations of DOFs.

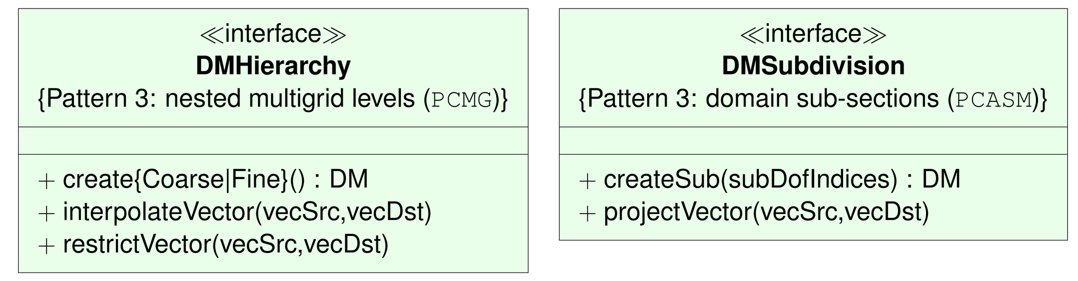
- Local representations are used to perform processor-local computations
- Global representations are distributed across memory without duplication
- Global-to-local maps can include communication of shared or "ghosted" data from other processors and enforcement of algebraic constraints

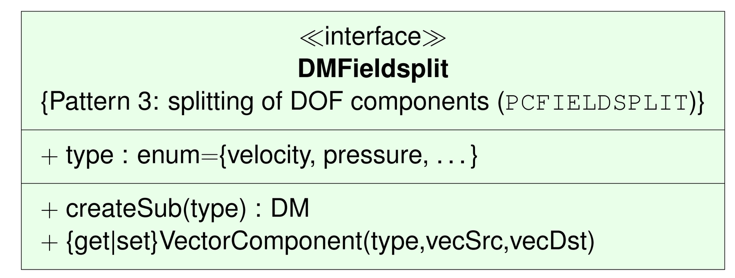
Pattern 2: Allocation and management of workspace vectors.

- Creates matrices / vectors required by solvers or inside application code
- Storage for temporary access to avoid excessive re-allocations



Pattern 3: Derivatives of DM's with corresponding operators to map DOFs.

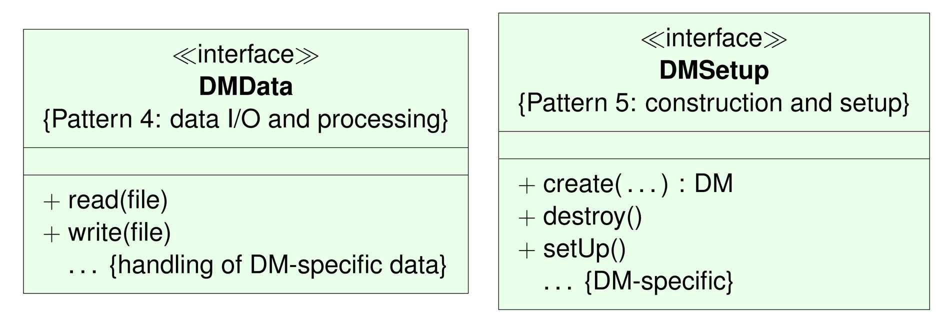




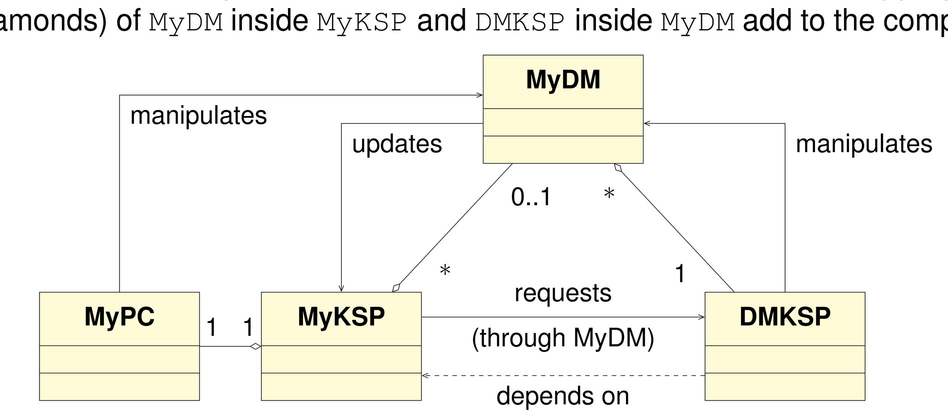
Pattern 4: Data storage, access, and processing.

Pattern 5: Construction of a fully working DM. Allows configuration of DM, e.g.:

- Input of geometry information and construction of a mesh and discretization
- Setup of differential operators and boundary conditions



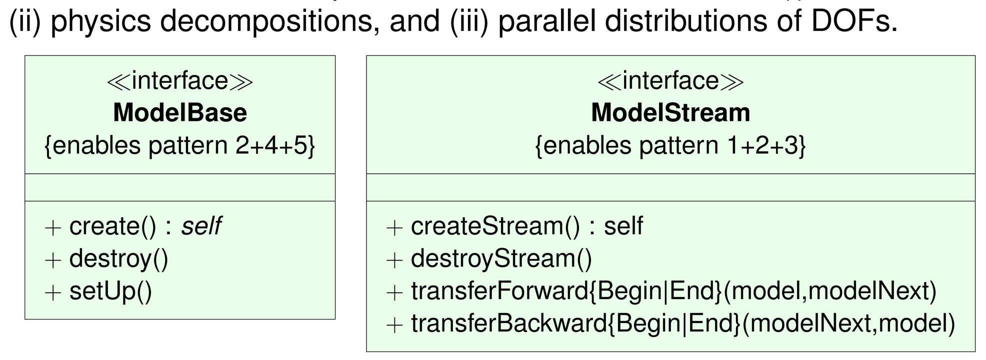
Example interactions between DM and KSP solver: Currently, the access patterns (arrows) create many complicated dependencies, which are challenging for software implementation and maintenance. The nested aggregation (diamonds) of MyDM inside MyKSP and DMKSP inside MyDM add to the complexity.

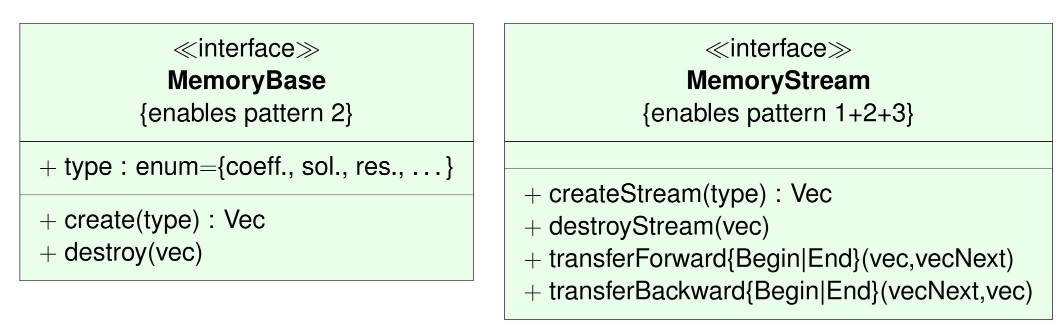


Propose: Model–Controller–Solver Design Patterns

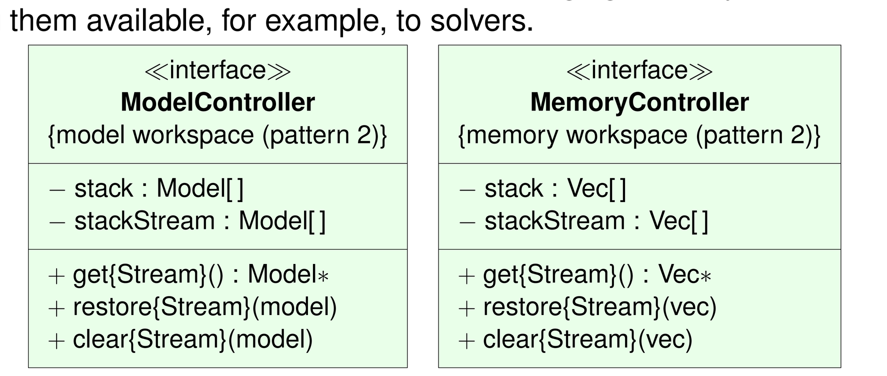
The new design is based on the five DM patterns introduced before and has the goal to realize geometric multigrid implementations (like the MG shown before) and accommodate different parallelism models (distributed & shared memory; MPI+OpenMP or MPI+GPU) via unifying interfaces.

New "Model" and "Memory" interfaces: Explicitly target separate designs for *memory* and a structure given to this memory via a *model.* We introduce the concept of *streaming* that enables the composition of objects that implement these interfaces. Examples of a stream of models are: (i) coarser MG levels, (ii) physics decompositions, and (iii) parallel distributions of DOFs.

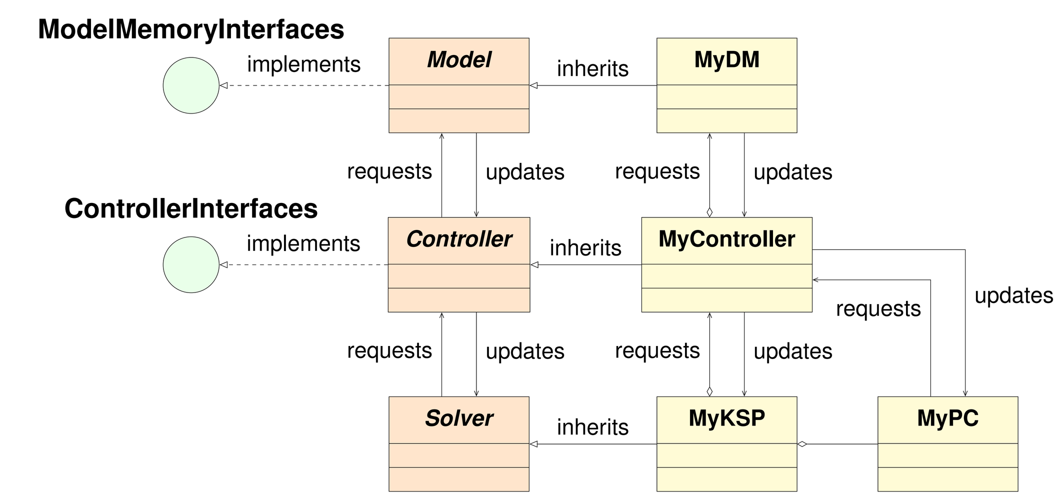




New Controller interfaces: Intended for managing memory and models and making them available, for example, to solvers.

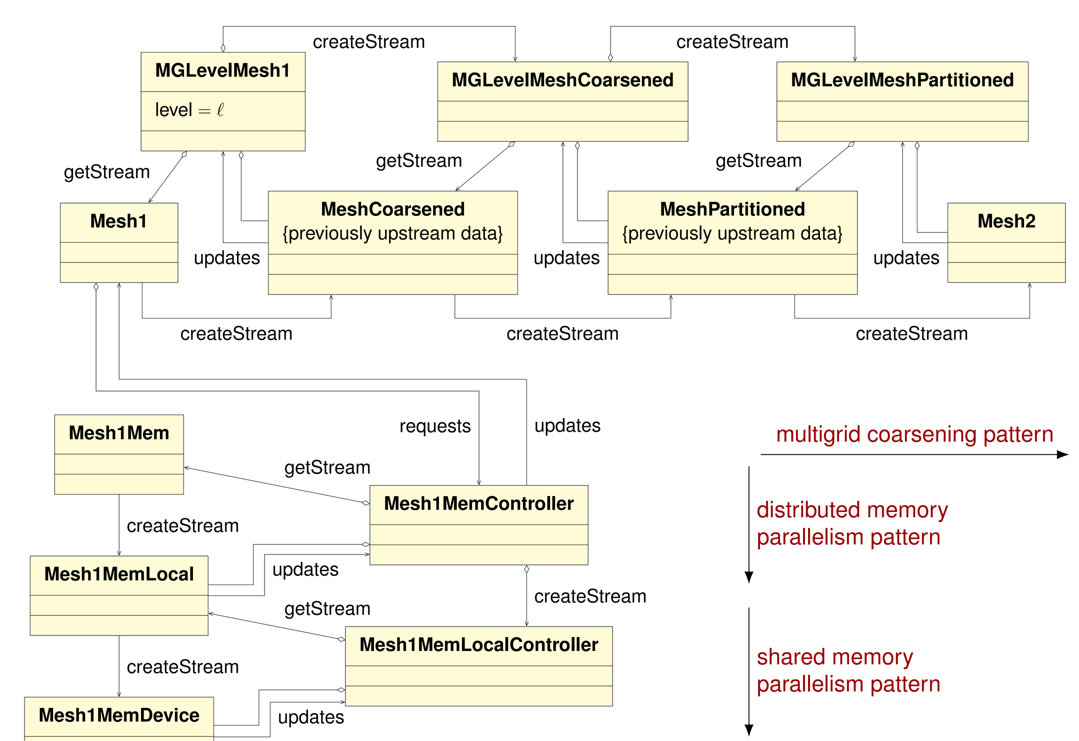


New Model–Controller–Solver (MCS) design pattern: Interactions between a DM and a KSP solver are restructured and the complex dependencies (see example before) do no longer exist.



Example of streamed interfaces for MG coarsening and parallelism: The flexibility and composability of the streamed interfaces allows us to express multiple design paradigms:

- Interactions between model and solver (see example above)
- Nested multigrid hierarchies
- Distributed and shared memory parallelism



Scientific Achievements

- Implicit solvers for complex PDEs with algorithmic scalability to billions of DOFs and parallel scalability to millions of processor cores [Rudi, Malossi, Isaac, et al., 2015]
- Generalization of extreme-scale MG to support a wide range of problems
- Software design to efficiently implement multi-level algorithms and their MPI+GPU parallelization

Significance & Impact

- Software components that encapsulate mathematical, computational science, and application-specific methods and additionally provide clear interfaces, allow for a more productive collaboration of interdisciplinary teams
- Accessibility of extreme-scale multi-level solvers to a large community of domain specialists, able to leverage state-of-the-art numerical methods
- Adaptability of parallel algorithms to a changing landscape of HPC hardware architectures