



Allison Baker

*Center for Applied Scientific Computing
Lawrence Livermore National Laboratory*

August 21, 2007



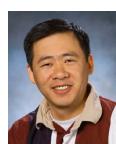
The *hypre* Team



Allison Baker



Rob Falgout



Barry Lee



Tzanio Kolev



Jeff Painter



Charles Tong



Panayot Vassilevski



Ulrike Yang

<http://www.llnl.gov/CASC/hypre/>

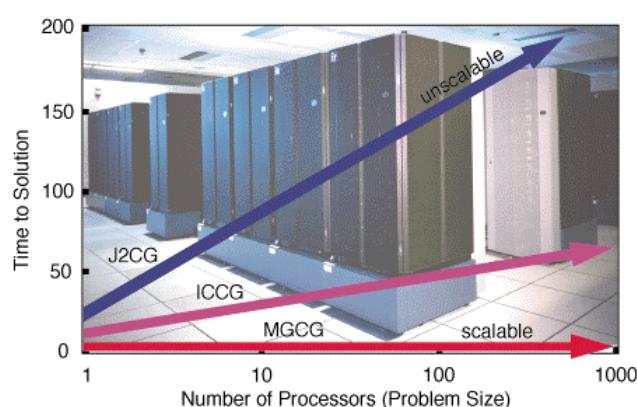
Outline

- Introduction / Motivation
- Getting Started / Conceptual Interfaces
- Structured-Grid Interface (`Struct`)
- Semi-Structured-Grid Interface (`sStruct`)
- Finite Element Interface (`FEI`)
- Linear-Algebraic Interface (`IJ`)
- Solvers and Preconditioners
- Additional Information

CASC

HYPRE 3

Scalability is a central issue for large-scale parallel computing

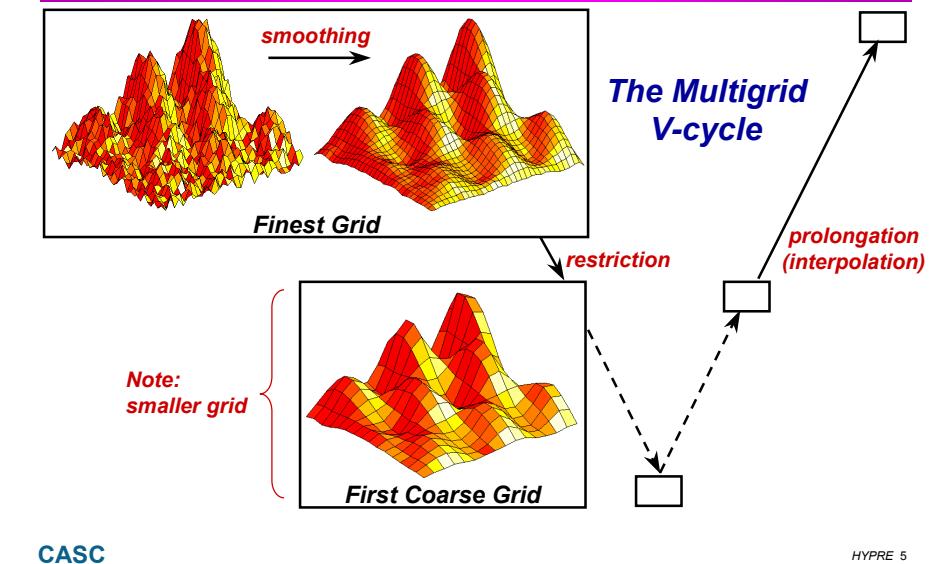


Linear solver convergence is a key scalability issue

CASC

HYPRE 4

Multigrid uses coarse grids to efficiently damp out smooth error components

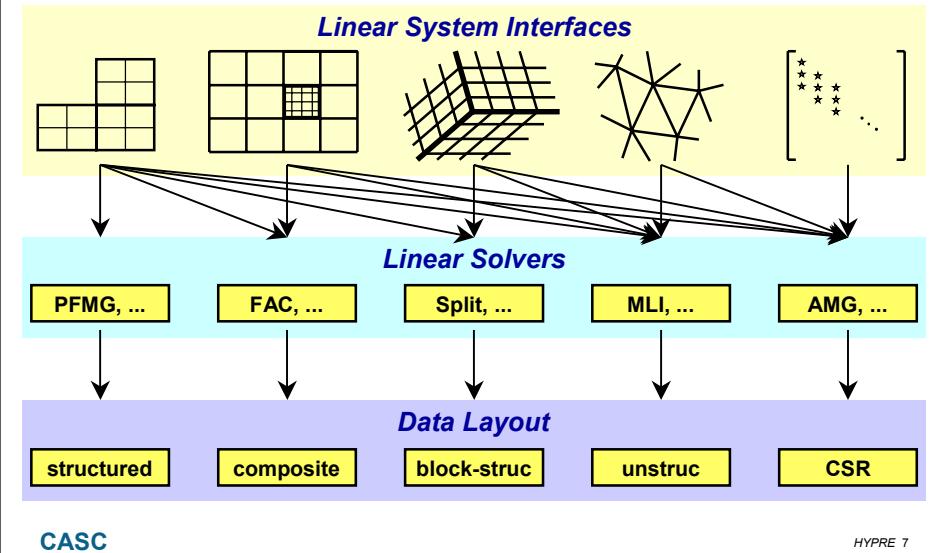


Getting Started

- Before writing your code:
 - choose a conceptual interface
 - choose a solver / preconditioner
 - choose a matrix type that is compatible with your solver / preconditioner and conceptual interface

- Now write your code:
 - build auxiliary structures (e.g., grids, stencils)
 - build matrix/vector through conceptual interface
 - build solver/preconditioner
 - solve the system
 - get desired information from the solver

Multiple interfaces are necessary to provide “best” solvers and data layouts



Why multiple interfaces? The key points

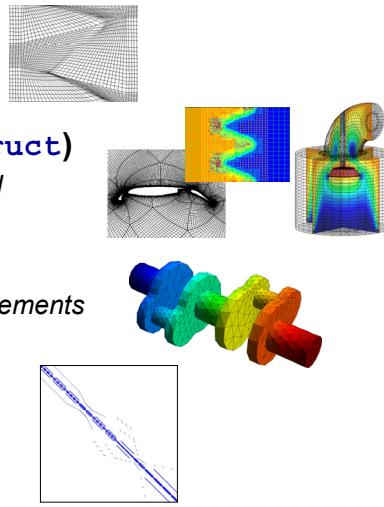
- Provides natural “views” of the linear system
- Eases some of the coding burden for users by eliminating the need to map to rows/columns
- Provides for more efficient (scalable) linear solvers
- Provides for more effective data storage schemes and more efficient computational kernels

CASC

HYPRE 8

Currently, *hypre* supports four conceptual interfaces

- **Structured-Grid (`Struct`)**
— logically rectangular grids
- **Semi-Structured-Grid (`sStruct`)**
— grids that are mostly structured
- **Finite Element (`FEI`)**
— unstructured grids with finite elements
- **Linear-Algebraic (`IJ`)**
— general sparse linear systems
- **More about each next...**

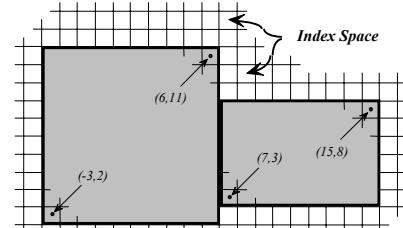


CASC

HYPRE 9

Structured-Grid System Interface (`Struct`)

- Appropriate for scalar applications on structured grids with a fixed stencil pattern
- Grids are described via a global d -dimensional **index space** (singles in 1D, tuples in 2D, and triples in 3D)
- A **box** is a collection of cell-centered indices, described by its “lower” and “upper” corners



- The scalar grid data is always associated with **cell centers** (unlike the more general `sStruct` interface)

CASC

HYPRE 10

Structured-Grid System Interface (Struct)

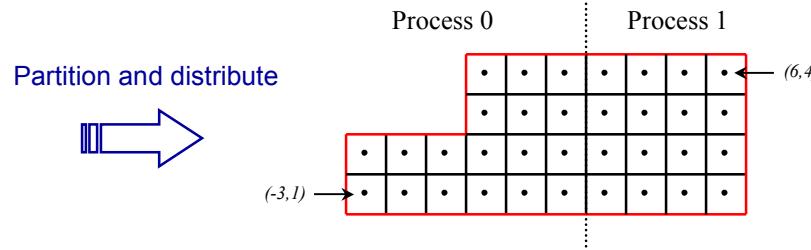
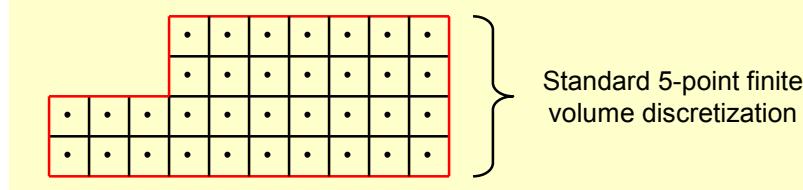
- There are four basic steps involved:
 - set up the Grid
 - set up the Stencil
 - set up the Matrix
 - set up the right-hand-side vector
- Consider the following 2D Laplacian problem

$$\begin{cases} \nabla^2 u = f, & \text{in the domain} \\ u = 0, & \text{on the boundary} \end{cases}$$

CASC

HYPRE 11

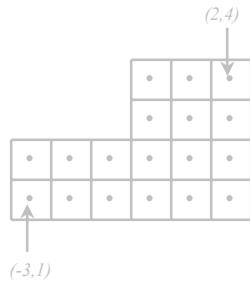
Structured-grid finite volume example:



CASC

HYPRE 12

Structured-grid finite volume example: Setting up the grid on process 0



Create the grid object

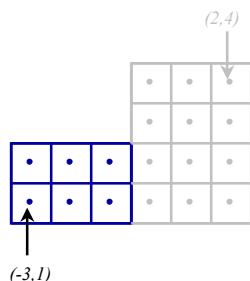
```
HYPRE_StructGrid grid;
int ndim    = 2;

HYPRE_StructGridCreate(MPI_COMM_WORLD, ndim, &grid);
```

CASC

HYPRE 13

Structured-grid finite volume example: Setting up the grid on process 0



Set grid extents for
first box

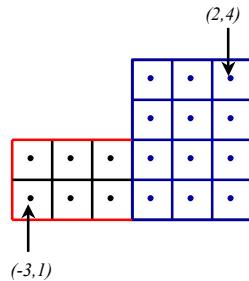
```
int ilo0[2] = {-3,1};
int iup0[2] = {-1,2};

HYPRE_StructGridSetExtents(grid, ilo0, iup0);
```

CASC

HYPRE 14

Structured-grid finite volume example: Setting up the grid on process 0



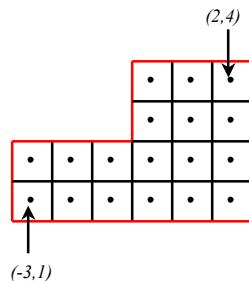
Set grid extents for
second box

```
int ilo1[2] = {0,1};  
int iup1[2] = {2,4};  
  
HYPRE_StructGridSetExtents(grid, ilo1, iup1);
```

CASC

HYPRE 15

Structured-grid finite volume example: Setting up the grid on process 0



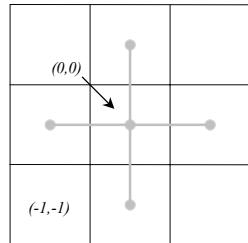
Assemble the grid

```
HYPRE_StructGridAssemble(grid);
```

CASC

HYPRE 16

Structured-grid finite volume example: Setting up the stencil (all processes)



stencil entries	geometries
0	$(0, 0)$
1	$(-1, 0)$
2	$(1, 0)$
3	$(0, -1)$
4	$(0, 1)$

Create the stencil
object

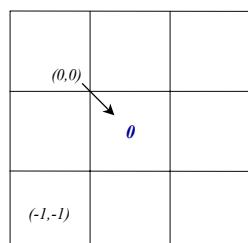
```
HYPRE_StructStencil stencil;
int ndim = 2;
int size = 5;

HYPRE_StructStencilCreate(ndim, size, &stencil);
```

CASC

HYPRE 17

Structured-grid finite volume example: Setting up the stencil (all processes)



stencil entries	geometries
0	$(0, 0)$
1	$(-1, 0)$
2	$(1, 0)$
3	$(0, -1)$
4	$(0, 1)$

Set stencil entries

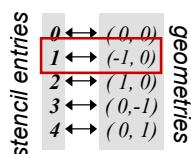
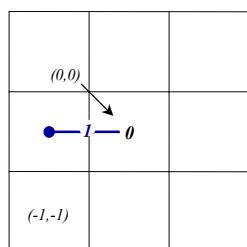
```
int entry = 0;
int offset[2] = {0,0};

HYPRE_StructStencilSetElement(stencil, entry, offset);
```

CASC

HYPRE 18

Structured-grid finite volume example: Setting up the stencil (all processes)



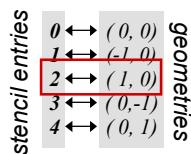
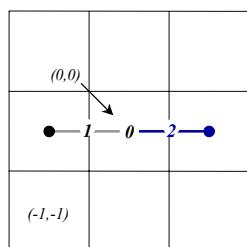
Set stencil entries

```
int entry = 1;  
int offset[2] = {-1, 0};  
  
HYPRE_StructStencilSetElement(stencil, entry, offset);
```

CASC

HYPRE 19

Structured-grid finite volume example: Setting up the stencil (all processes)



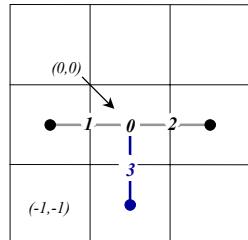
Set stencil entries

```
int entry = 2;  
int offset[2] = {1, 0};  
  
HYPRE_StructStencilSetElement(stencil, entry, offset);
```

CASC

HYPRE 20

Structured-grid finite volume example: Setting up the stencil (all processes)



stencil entries	geometries
0	$(0, 0)$
1	$(-1, 0)$
2	$(1, 0)$
3	$(0, -1)$
4	$(0, 1)$

Set stencil entries

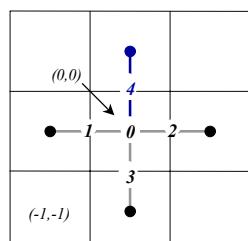
```
int entry = 3;
int offset[2] = {0, -1};

HYPRE_StructStencilSetElement(stencil, entry, offset);
```

CASC

HYPRE 21

Structured-grid finite volume example: Setting up the stencil (all processes)



stencil entries	geometries
0	$(0, 0)$
1	$(-1, 0)$
2	$(1, 0)$
3	$(0, -1)$
4	$(0, 1)$

Set stencil entries

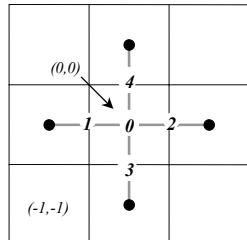
```
int entry = 4;
int offset[2] = {0, 1};

HYPRE_StructStencilSetElement(stencil, entry, offset);
```

CASC

HYPRE 22

Structured-grid finite volume example: Setting up the stencil (all processes)



stencil entries geometries

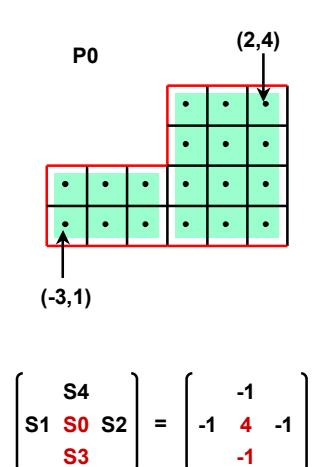
0	$(0, 0)$
1	$(-1, 0)$
2	$(1, 0)$
3	$(0, -1)$
4	$(0, 1)$

That's it!
There is no assemble
routine

CASC

HYPRE 23

Structured-grid finite volume example : Setting up the matrix on process 0



```

HYPRE_StructMatrix A;
double vals[24] = {4, -1, 4, -1, ...};
int nentries = 2;
int entries[2] = {0,3};

HYPRE_StructMatrixCreate(MPI_COMM_WORLD,
    grid, stencil, &A);
HYPRE_StructMatrixInitialize(A);

HYPRE_StructMatrixSetBoxValues(A,
    ilo0, iup0, nentries, entries, vals);
HYPRE_StructMatrixSetBoxValues(A,
    ilol, iupl, nentries, entries, vals);
/* set boundary conditions */
...
HYPRE_StructMatrixAssemble(A);

```

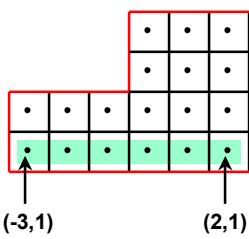
CASC

HYPRE 24

Structured-grid finite volume example :

Setting up the matrix bc's on process 0

P0



CASC

$$\begin{bmatrix} S_4 \\ S_1 \ S_0 \ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \ 4 \ -1 \\ 0 \end{bmatrix}$$

```

int ilo[2] = {-3, 1};
int iup[2] = { 2, 1};
double vals[6] = {0, 0, ...};
int nentries = 1;

/* set interior coefficients */
...
/* implement boundary conditions */
...
i = 3;
HYPRE_StructMatrixSetBoxValues(A,
    ilo, iup, nentries, &i, vals);
/* complete implementation of bc's */
...

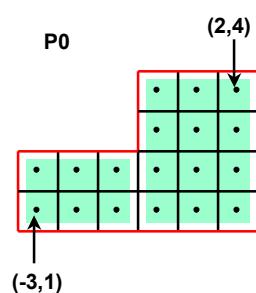
```

HYPRE 25

A structured-grid finite volume example :

Setting up the right-hand-side vector

P0



CASC

```

HYPRE_StructVector b;
double vals[12] = {0, 0, ...};

HYPRE_StructVectorCreate(MPI_COMM_WORLD,
    grid, &b);
HYPRE_StructVectorInitialize(b);

HYPRE_StructVectorSetBoxValues(b,
    ilo0, iup0, vals);
HYPRE_StructVectorSetBoxValues(b,
    ilo1, iup1, vals);

HYPRE_StructVectorAssemble(b);

```

HYPRE 26

Symmetric Matrices

- Some solvers support symmetric storage
- Between Create() and Initialize(), call:

```
HYPRE_StructMatrixSetSymmetric(A, 1);
```

- For best efficiency, only set half of the coefficients

$$\begin{bmatrix} & (0,1) \\ (0,0) & (1,0) \end{bmatrix} \Leftrightarrow \begin{bmatrix} & s_2 \\ s_0 & s_1 \end{bmatrix}$$

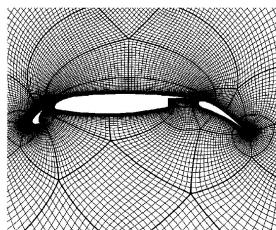
- This is enough info to recover the full 5-pt stencil

CASC

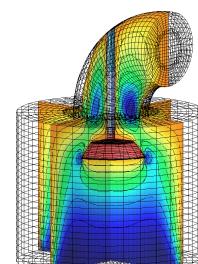
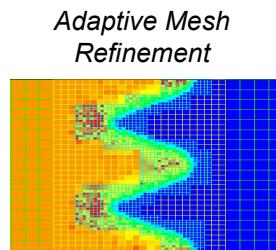
HYPRE 27

Semi-Structured-Grid System Interface (SStruct)

- Allows more general grids:
 - Grids that are mostly (but not entirely) structured
 - Examples: *block-structured grids, structured adaptive mesh refinement grids, overset grids*



Block-Structured



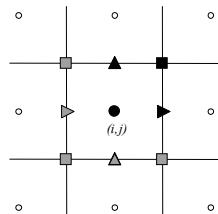
Overset

CASC

HYPRE 28

Semi-Structured-Grid System Interface (**sStruct**)

- Allows more general PDE's
 - Multiple variables (system PDE's)
 - Multiple variable types (**cell centered, face centered, vertex centered, ...**)



Variables are referenced by
the abstract cell-centered
index to the left and down

CASC

HYPRE 29

Semi-Structured-Grid System Interface (**sStruct**)

- The **sStruct** grid is composed out of a number of structured grid **parts**
- The interface uses a **graph** to allow nearly arbitrary relationships between part data
- The graph is constructed from stencils plus some additional data-coupling information set either
 - directly with `GraphAddEntries()`, or
 - by relating parts with `GridSetNeighborBox()`
- We will consider two examples:
 - block-structured grid
 - structured adaptive mesh refinement

CASC

HYPRE 30

Semi-Structured-Grid System Interface (SStruct)

- There are five basic steps involved:
 - set up the Grid
 - set up the Stencils
 - **set up the Graph**
 - set up the Matrix
 - set up the right-hand-side Vector

CASC

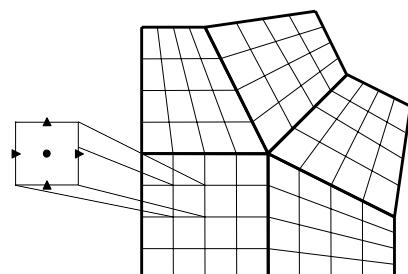
HYPRE 31

Block-structured grid example (SStruct)

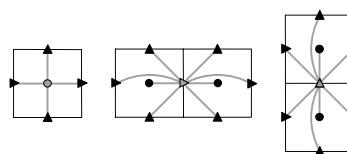
- Consider the following block-structured grid discretization of the diffusion equation

$$-\nabla \cdot (D \nabla u) + \sigma u = f$$

A block-structured grid with
3 variable types



The 3 discretization stencils

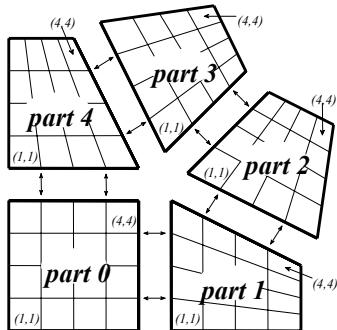


CASC

HYPRE 32

Block-structured grid example (SStruct)

- The Grid is described in terms of 5 separate logically-rectangular parts

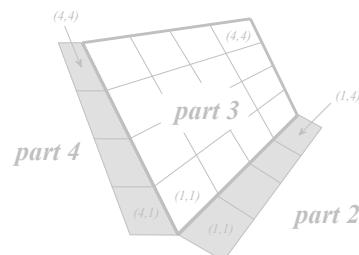


- Here, we assume 5 processes such that process p owns part p (note: user determines the distribution)
- We consider the interface calls made by process 3

CASC

HYPRE 33

Block-structured grid example: Setting up the grid on process 3



Create the grid object

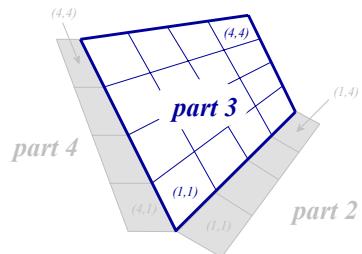
```
HYPRE_SStructGrid grid;
int ndim    = 2;
int nparts  = 5;

HYPRE_SStructGridCreate(MPI_COMM_WORLD, ndim, nparts, &grid);
```

CASC

HYPRE 34

Block-structured grid example: Setting up the grid on process 3



Set grid extents for
part 3

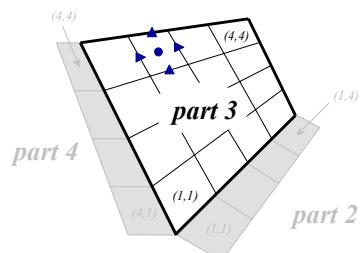
```
int part = 3;
int ilower[2] = {1,1};
int iupper[2] = {4,4};

HYPRE_SStructGridSetExtents(grid, part, ilower, iupper);
```

CASC

HYPRE 35

Block-structured grid example: Setting up the grid on process 3



Set grid variables for
part 3

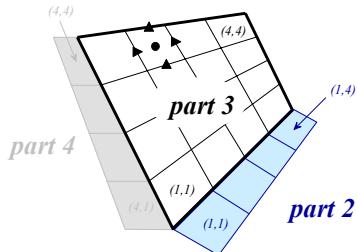
```
int part = 3;
int nvars = 3;
int vartypes[3] = {HYPRE_SSTRUCT_VARIABLE_CELL,
                   HYPRE_SSTRUCT_VARIABLE_XFACE,
                   HYPRE_SSTRUCT_VARIABLE_YFACE};

HYPRE_SStructGridSetVariables(grid, part, nvars, vartypes);
```

CASC

HYPRE 36

Block-structured grid example: Setting up the grid on process 3



Set spatial relationship
between parts 3 and 2

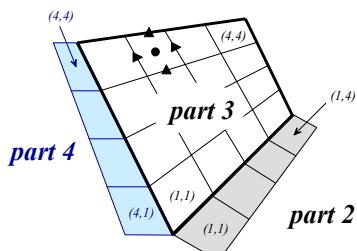
```
int part = 3, nbor_part = 2;
int ilower[2] = {1,0}, iupper[2] = {4,0};
int nbor_ilower[2] = {1,1}, nbor_iupper[2] = {1,4};
int index_map[2] = {1,0};

HYPRE_SStructGridSetNeighborBox(grid, part, ilower, iupper,
nbor_part, nbor_ilower, nbor_iupper, index_map);
```

CASC

HYPRE 37

Block-structured grid example: Setting up the grid on process 3



Set spatial relationship
between parts 3 and 4

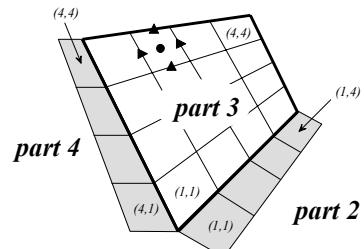
```
int part = 3, nbor_part = 4;
int ilower[2] = {0,1}, iupper[2] = {0,4};
int nbor_ilower[2] = {4,1}, nbor_iupper[2] = {4,4};
int index_map[2] = {0,1};

HYPRE_SStructGridSetNeighborBox(grid, part, ilower, iupper,
nbor_part, nbor_ilower, nbor_iupper, index_map);
```

CASC

HYPRE 38

Block-structured grid example: Setting up the grid on process 3



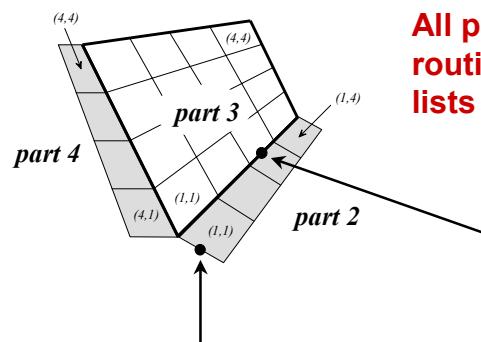
Assemble the grid

```
HYPRE_SStructGridAssemble(grid);
```

CASC

HYPRE 39

Block-structured grid example: some comments on SetNeighborBox()



All parts related via this
routine must have consistent
lists of variables and types

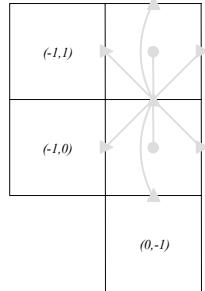
Variables may have
different types on
different parts (e.g.,
y-face on part 3 and
x-face on part 2)

Some variables on different
parts become "the same"

CASC

HYPRE 40

Block-structured grid example: Setting up the y-face stencil (all processes)



stencil entries	geometries
0	$(0,0); \blacktriangle$
1	$(0,-1); \blacktriangle$
2	$(0,1); \blacktriangle$
3	$(0,0); \bullet$
4	$(0,1); \bullet$
5	$(-1,0); \blacktriangleright$
6	$(0,0); \blacktriangleright$
7	$(-1,1); \blacktriangleright$
8	$(0,1); \blacktriangleright$

Create the stencil object

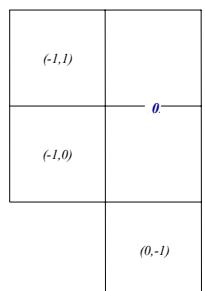
```
HYPRE_SStructStencil stencil;
int ndim = 2;
int size = 9;

HYPRE_SStructStencilCreate(ndim, size, &stencil);
```

CASC

HYPRE 41

Block-structured grid example: Setting up the y-face stencil (all processes)



stencil entries	geometries
0	$(0,0); \blacktriangle$
1	$(0,-1); \blacktriangle$
2	$(0,1); \blacktriangle$
3	$(0,0); \bullet$
4	$(0,1); \bullet$
5	$(-1,0); \blacktriangleright$
6	$(0,0); \blacktriangleright$
7	$(-1,1); \blacktriangleright$
8	$(0,1); \blacktriangleright$

Set stencil entries

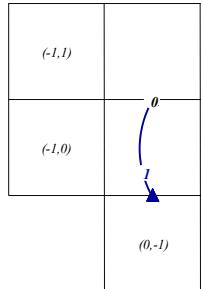
```
int entry = 0;
int offset[2] = {0,0};
int var = 2; /* the y-face variable number */

HYPRE_SStructSetStencilEntry(stencil, entry, offset, var);
```

CASC

HYPRE 42

Block-structured grid example: Setting up the y-face stencil (all processes)



stencil entries	Geometries
0 \leftrightarrow (0,0);▲	
1 \leftrightarrow (0,-1);▲	
2 \leftrightarrow (0,1);●	
3 \leftrightarrow (0,0);●	
4 \leftrightarrow (0,1);●	
5 \leftrightarrow (-1,0);▶	
6 \leftrightarrow (0,0);▶	
7 \leftrightarrow (-1,1);▶	
8 \leftrightarrow (0,1);▶	

Set stencil entries

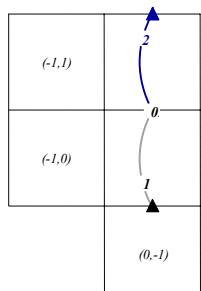
```
int entry = 1;
int offset[2] = {0,-1};
int var = 2; /* the y-face variable number */

HYPRE_SStructSetStencilEntry(stencil, entry, offset, var);
```

CASC

HYPRE 43

Block-structured grid example: Setting up the y-face stencil (all processes)



stencil entries	Geometries
0 \leftrightarrow (0,0);▲	
1 \leftrightarrow (0,-1);▲	
2 \leftrightarrow (0,1);▲	
3 \leftrightarrow (0,0);●	
4 \leftrightarrow (0,1);●	
5 \leftrightarrow (-1,0);▶	
6 \leftrightarrow (0,0);▶	
7 \leftrightarrow (-1,1);▶	
8 \leftrightarrow (0,1);▶	

Set stencil entries

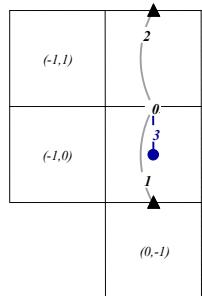
```
int entry = 2;
int offset[2] = {0,1};
int var = 2; /* the y-face variable number */

HYPRE_SStructSetStencilEntry(stencil, entry, offset, var);
```

CASC

HYPRE 44

Block-structured grid example: Setting up the y-face stencil (all processes)



stencil entries	elements
0	$(0,0); \blacktriangle$
1	$(0,-1); \blacktriangle$
2	$(0,1); \blacktriangle$
3	$(0,0); \bullet$
4	$(0,1); \blacktriangleright$
5	$(-1,0); \blacktriangleright$
6	$(0,0); \blacktriangleright$
7	$(-1,1); \blacktriangleright$
8	$(0,1); \blacktriangleright$

Set stencil entries

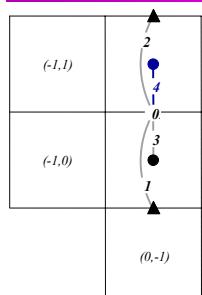
```
int entry = 3;
int offset[2] = {0,0};
int var = 0; /* the cell-centered variable number */

HYPRE_SStructSetStencilEntry(stencil, entry, offset, var);
```

CASC

HYPRE 45

Block-structured grid example: Setting up the y-face stencil (all processes)



stencil entries	elements
0	$(0,0); \blacktriangle$
1	$(0,-1); \blacktriangle$
2	$(0,1); \blacktriangle$
3	$(0,0); \bullet$
4	$(0,1); \bullet$
5	$(-1,0); \blacktriangleright$
6	$(0,0); \blacktriangleright$
7	$(-1,1); \blacktriangleright$
8	$(0,1); \blacktriangleright$

Set stencil entries

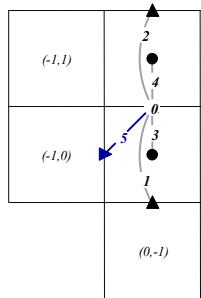
```
int entry = 4;
int offset[2] = {0,1};
int var = 0; /* the cell-centered variable number */

HYPRE_SStructSetStencilEntry(stencil, entry, offset, var);
```

CASC

HYPRE 46

Block-structured grid example: Setting up the y-face stencil (all processes)



stencil entries	geometries
0	$(0,0); \blacktriangle$
1	$(0,-1); \blacktriangle$
2	$(0,1); \blacktriangle$
3	$(0,0); \bullet$
4	$(0,1); \bullet$
5	$\text{HYPRE_SStructSetStencilEntry}(\text{stencil}, \text{entry}, \text{offset}, \text{var});$
6	$(0,0); \triangleright$
7	$(-1,1); \triangleright$
8	$(0,1); \triangleright$

Set stencil entries

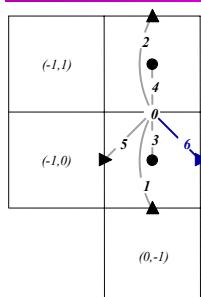
```
int entry = 5;
int offset[2] = {-1,0};
int var = 1; /* the x-face variable number */

HYPRE_SStructSetStencilEntry(stencil, entry, offset, var);
```

CASC

HYPRE 47

Block-structured grid example: Setting up the y-face stencil (all processes)



stencil entries	geometries
0	$(0,0); \blacktriangle$
1	$(0,-1); \blacktriangle$
2	$(0,1); \blacktriangle$
3	$(0,0); \bullet$
4	$(0,1); \bullet$
5	$\text{HYPRE_SStructSetStencilEntry}(\text{stencil}, \text{entry}, \text{offset}, \text{var});$
6	$(0,0); \triangleright$
7	$(-1,1); \triangleright$
8	$(0,1); \triangleright$

Set stencil entries

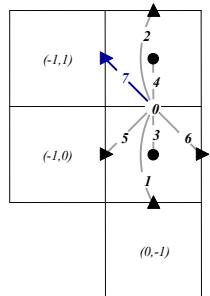
```
int entry = 6;
int offset[2] = {0,0};
int var = 1; /* the x-face variable number */

HYPRE_SStructSetStencilEntry(stencil, entry, offset, var);
```

CASC

HYPRE 48

Block-structured grid example: Setting up the y-face stencil (all processes)



stencil entries	geometries
0	$(0,0); \blacktriangle$
1	$(0,-1); \blacktriangle$
2	$(0,1); \blacktriangle$
3	$(0,0); \bullet$
4	$(0,1); \bullet$
5	$(-1,0); \blacktriangleright$
6	$(0,0); \blacktriangleright$
7	$(-1,1); \blacktriangleright$
8	$(0,1); \blacktriangleright$

Set stencil entries

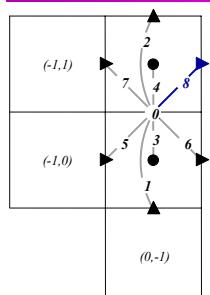
```
int entry = 7;
int offset[2] = {-1,1};
int var = 1; /* the x-face variable number */

HYPRE_SStructSetStencilEntry(stencil, entry, offset, var);
```

CASC

HYPRE 49

Block-structured grid example: Setting up the y-face stencil (all processes)



stencil entries	geometries
0	$(0,0); \blacktriangle$
1	$(0,-1); \blacktriangle$
2	$(0,1); \blacktriangle$
3	$(0,0); \bullet$
4	$(0,1); \bullet$
5	$(-1,0); \blacktriangleright$
6	$(0,0); \blacktriangleright$
7	$(-1,1); \blacktriangleright$
8	$(0,1); \blacktriangleright$

Set stencil entries

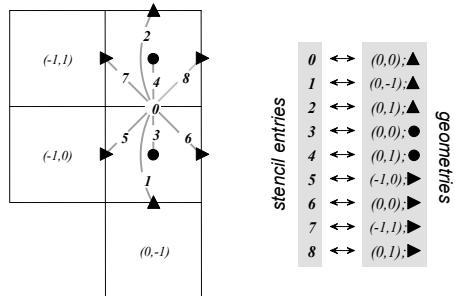
```
int entry = 8;
int offset[2] = {0,1};
int var = 1; /* the x-face variable number */

HYPRE_SStructSetStencilEntry(stencil, entry, offset, var);
```

CASC

HYPRE 50

Block-structured grid example: Setting up the y-face stencil (all processes)

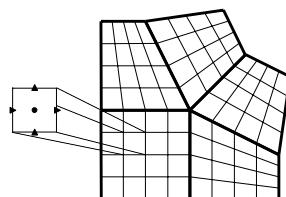


That's it!
There is no assemble
routine

CASC

HYPRE 51

Block-structured grid example: Setting up the graph on process 3



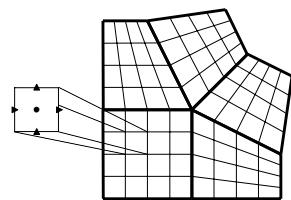
Create the graph
object

```
HYPRE_SStructGraph graph;  
  
HYPRE_SStructGraphCreate(MPI_COMM_WORLD, grid, &graph);
```

CASC

HYPRE 52

Block-structured grid example: Setting up the graph on process 3



Set the cell-centered
stencil for each part

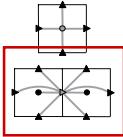
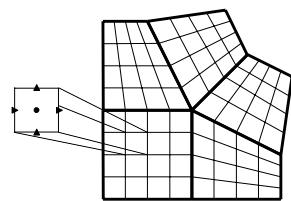
```
int part;
int var = 0;
HYPRE_SStructStencil cell_stencil;

HYPRE_SStructGraphSetStencil(graph, part, var, cell_stencil);
```

CASC

HYPRE 53

Block-structured grid example: Setting up the graph on process 3



Set the x-face stencil
for each part

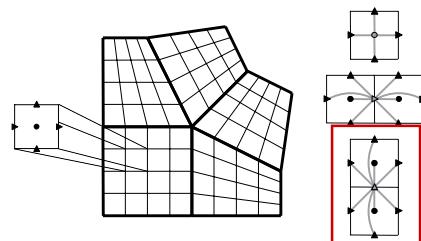
```
int part;
int var = 1;
HYPRE_SStructStencil x_stencil;

HYPRE_SStructGraphSetStencil(graph, part, var, x_stencil);
```

CASC

HYPRE 54

Block-structured grid example: Setting up the graph on process 3



Set the y-face stencil
for each part

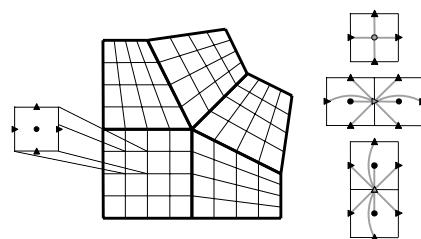
```
int part;
int var = 2;
HYPRE_SStructStencil y_stencil;

HYPRE_SStructGraphSetStencil(graph, part, var, y_stencil);
```

CASC

HYPRE 55

Block-structured grid example: Setting up the graph on process 3



Assemble the graph

```
/* No need to add non-stencil entries
 * with HYPRE_SStructGraphAddEntries */

HYPRE_SStructGraphAssemble(graph);
```

CASC

HYPRE 56

Block-structured grid example: Setting up the matrix and vector

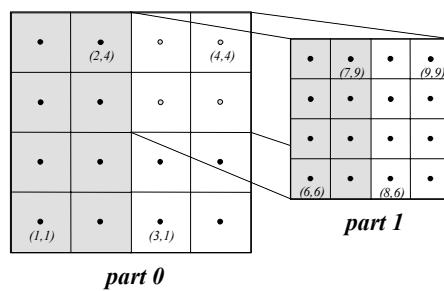
- The matrix and vector objects are constructed in a manner similar to the `struct` interface
- Matrix coefficients are set with the routines
 - `HYPRE_SStructMatrixSetValues`
 - `HYPRE_SStructMatrixAddToValues`
- Vector values are set with similar routines
 - `HYPRE_SStructVectorSetValues`
 - `HYPRE_SStructVectorAddToValues`

CASC

HYPRE 57

Structured AMR example (`SStruct`)

- Consider a simple cell-centered discretization of the Laplacian on the following structured AMR grid



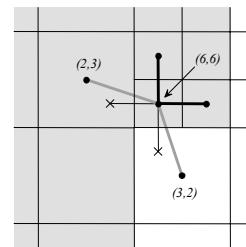
- Each AMR grid level is defined as a separate part
- Assume 2 processes with shaded regions on process 0 and unshaded regions on process 1

CASC

HYPRE 58

Structured AMR example (SStruct)

- The **grid** is constructed using straightforward calls to the routines `HYPRE_SStructGridSetExtents` and `HYPRE_SStructGridSetVariables` as in the previous block-structured grid example
- The **graph** is constructed from a cell-centered stencil plus additional **non-stencil entries** at coarse-fine interfaces
- These non-stencil entries are set one variable at a time using `HYPRE_SStructGraphAddEntries()`



CASC

HYPRE 59

Building different matrix/vector storage formats with the SStruct interface

- Efficient preconditioners often require specific matrix/vector storage schemes
- Between `Create()` and `Initialize()`, call:

```
HYPRE_SStructMatrixSetObjectType(A, HYPRE_PARCSR);
```
- After `Assemble()`, call:

```
HYPRE_SStructMatrixGetObject(A, &parcsr_A);
```
- Now, use the ParCSR matrix with compatible solvers such as BoomerAMG (algebraic multigrid)

CASC

HYPRE 60

Comments on SStruct interface

- The routine `HYPRE_SStructGridAddVariables` allows additional variables to be added to individual cells, but is not yet implemented
- The routine `HYPRE_SStructGridSetNeighborBox` only supports cell-centered variable types currently
- New FAC solver in *hypre* for AMR

CASC

HYPRE 61

Finite Element Interface (FEI)

- The FEI interface is designed for finite element discretizations on unstructured grids
- See the following for information on how to use it

R. L. Clay et al. An annotated reference guide to the Finite Element Interface (FEI) Specification, Version 1.0. Sandia National Laboratories, Livermore, CA, Technical Report SAND99-8229, 1999.

CASC

HYPRE 62

Linear-Algebraic System Interface (IJ)

- The IJ interface provides access to general sparse-matrix solvers, but not specialized solvers
- There are two basic steps involved:
 - set up the Matrix
 - set up the right-hand-side vector

CASC

HYPRE 63

An example for the IJ interface: *setting up a Matrix*

$$A = \left(\begin{array}{ccc|ccc|c} 4 & -1 & & -1 & & & \\ -1 & 4 & -1 & & -1 & & \\ -1 & & 4 & & & -1 & \\ \hline -1 & & & 4 & -1 & & -1 \\ -1 & & -1 & 4 & -1 & -1 & \\ -1 & & & -1 & 4 & -1 & \\ \hline & & -1 & & 4 & -1 & \\ & & & -1 & -1 & 4 & \\ & & & & -1 & -1 & 4 \end{array} \right) \quad \begin{array}{l} \text{Proc 0:} \\ \text{Rows} \\ 0,1,2,3 \end{array}$$

Proc 1:
Rows
4,5,6

Proc 2:
Rows
7,8



CASC

HYPRE 64

An example for the IJ interface: setting up the Matrix (on Proc 1)

column indices

$$\begin{array}{ccccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
 \begin{matrix} 4 \\ \text{rows} \\ 5 \\ 6 \end{matrix} & \left(\begin{array}{ccccccccc}
 -1 & -1 & 4 & -1 & -1 \\
 -1 & -1 & 4 & -1 & -1 \\
 -1 & & 4 & -1 &
 \end{array} \right)
 \end{array}$$

Create and initialize matrix

```

HYPRE_IJMatrix A;
int ilower = 4, iupper = 6;
int jlower = 4, jupper = 6;

HYPRE_IJMatrixCreate(MPI_COMM_WORLD, ilower, iupper,
                     jlower, jupper, &A);
HYPRE_IJMatrixSetObjectType(A, HYPRE_PARCSR);

HYPRE_IJMatrixInitialize(A);

```

CASC

HYPRE 65

An example for the IJ interface: setting up the Matrix (on Proc 1)

column indices

$$\begin{array}{ccccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
 \begin{matrix} 4 \\ \text{rows} \\ 5 \\ 6 \end{matrix} & \left(\begin{array}{ccccccccc}
 -1 & -1 & 4 & -1 & -1 \\
 -1 & -1 & 4 & -1 & -1 \\
 -1 & & 4 & -1 &
 \end{array} \right)
 \end{array}$$

Set matrix coefficients

```

HYPRE_IJMatrix A;
int nrows = 3, ncols[3] = {5, 4, 3}, rows[4] = {4, 5, 6};
int cols[12] = {1,3,4,5,7, 2,4,5,8, 3,6,7};
double values[12] = {-1,-1,4,-1,-1, -1,-1,4,-1, -1,4,-1};

/* set matrix coefficients several rows at a time */
HYPRE_IJMatrixSetValues(A, nrows, ncols, rows, cols, values);

```

CASC

HYPRE 66

An example for the IJ interface: setting up the Matrix (on Proc 1)

$$\begin{array}{c} \text{column indices} \\ \hline
 \begin{matrix} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \begin{matrix} 4 \\ 5 \\ 6 \end{matrix} & \left(\begin{matrix} -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & & 4 & -1 & \end{matrix} \right) \end{matrix}
 \end{array}$$

**Assemble matrix
and retrieve object**

```

HYPRE_IJMatrix A;
HYPRE_IJMatrixAssemble(A);
HYPRE_ParCSRMatrix parcsr_A;
HYPRE_IJMatrixGetObject(A, &parcsr_A);

```

CASC

HYPRE 67

Current solver / preconditioner availability via hypre's conceptual interfaces

Solvers	System Interfaces			
	Struct	SStruct	FEI	IJ
Jacobi	✓	✓		
SMG	✓	✓		
PFMG	✓	✓		
Split		✓		
SysPFMG		✓		
FAC		✓		
Maxwell		✓		
AMS		✓	✓	✓
BoomerAMG		✓	✓	✓
MLI		✓	✓	✓
ParaSails		✓	✓	✓
Euclid		✓	✓	✓
PILUT		✓	✓	✓
PCG	✓	✓	✓	✓
GMRES	✓	✓	✓	✓
BiCGSTAB	✓	✓	✓	✓
Hybrid	✓	✓	✓	✓

CASC

HYPRE 68

Setup and use of solvers is largely the same (see Reference Manual for details)

- Create the solver

```
HYPRE_SolverCreate(MPI_COMM_WORLD, &solver);
```

- Set parameters

```
HYPRE_SolverSetTol(solver, 1.0e-06);
```

- Prepare to solve the system

```
HYPRE_SolverSetup(solver, A, b, x);
```

- Solve the system

```
HYPRE_SolverSolve(solver, A, b, x);
```

- Get solution info out via conceptual interface

```
HYPRE_StructVectorGetValues(struct_x, index, values);
```

- Destroy the solver

```
HYPRE_SolverDestroy(solver);
```

CASC

HYPRE 69

Solver example: SMG-PCG

```
/* define preconditioner (one symmetric V(1,1)-cycle) */
HYPRE_StructSMGCreate(MPI_COMM_WORLD, &precond);
HYPRE_StructSMGSetMaxIter(precond, 1);
HYPRE_StructSMGSetTol(precond, 0.0);
HYPRE_StructSMGSetZeroGuess(precond);
HYPRE_StructSMGSetNumPreRelax(precond, 1);
HYPRE_StructSMGSetNumPostRelax(precond, 1);

HYPRE_StructPCGCreate(MPI_COMM_WORLD, &solver);
HYPRE_StructPCGSetTol(solver, 1.0e-06);

/* set preconditioner */
HYPRE_StructPCGSetPrecond(solver,
    HYPRE_StructSMGSolve, HYPRE_StructSMGSetup, precond);

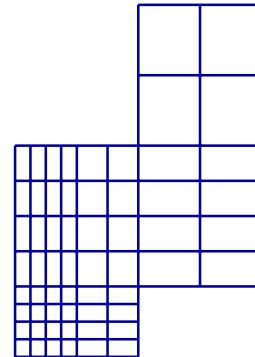
HYPRE_StructPCGSetup(solver, A, b, x);
HYPRE_StructPCGSolve(solver, A, b, x);
```

CASC

HYPRE 70

SMG and PFMG are semicoarsening multigrid methods for structured grids

- Interface: Struct, SStruct
- Matrix Class: Struct
- SMG uses plane smoothing in 3D, where each plane “solve” is effected by one 2D V-cycle
- SMG is very robust
- PFMG uses simple pointwise smoothing, and is less robust
- Constant-coefficient versions!

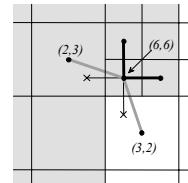


CASC

HYPRE 71

FAC is an algebraic cell-centered fast adaptive composite grid solver in *hypre*

- Interface: SStruct
- Matrix Class: SStruct
- Requires only the composite matrix
 - no coarse underlying matrix needed
- Does not require nested AMR levels in the processor distribution, e.g., 3 levels on 2 procs
 - uses intra- and inter-level communication



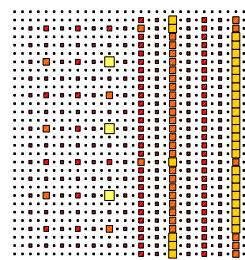
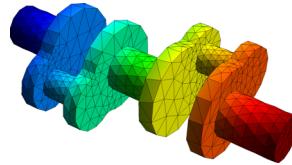
- Designed for smooth-coefficient diffusion problems

CASC

HYPRE 72

BoomerAMG is an algebraic multigrid method for unstructured grids

- Interface: SStruct, FEI, IJ
- Matrix Class: ParCSR
- Originally developed as a general matrix method (i.e., assumes given only A, x, and b)
- Various coarsening, interpolation and relaxation schemes
- Automatically coarsens “grids”
- Can solve systems of PDEs if additional information is provided

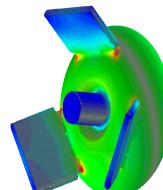


CASC

HYPRE 73

Maxwell is a definite Maxwell solver for (semi)-structured grids

- Interface: SStruct
- Matrix Class: SStruct
- Solves definite problems
 $\nabla \times \alpha \nabla \times E + \beta E = f, \beta > 0$
- Uses multiple coarsening and special relaxation, and a coupled hierarchy to resolve different vector components of the correction
- Requires the linear system and a gradient matrix
- Only for edge finite element discretizations

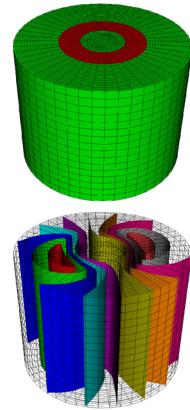


CASC

HYPRE 74

AMS is an auxiliary space Maxwell solver for unstructured grids

- Interface: SStruct, FEI, IJ
- Matrix Class: ParCSR
- Solves definite problems:
 $\nabla \times \alpha \nabla \times E + \beta E = f, \alpha > 0, \beta \geq 0$
- Requires additional gradient matrix and mesh coordinates
- Variational form of Hiptmair-Xu
- Employs BoomerAMG
- Only for FE discretizations



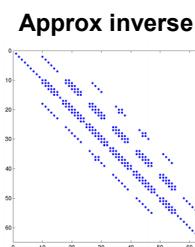
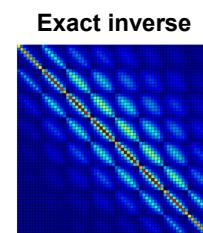
Copper Wire in Air
Conductivity jump of 10^6

CASC

HYPRE 75

ParaSAILS is an approximate inverse method for sparse linear systems

- Interface: SStruct, FEI, IJ
- Matrix Class: ParCSR
- Approximates the inverse of A by a sparse matrix M by minimizing the Frobenius norm of $I - AM$
- Uses graph theory to predict good sparsity patterns for M

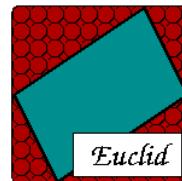


CASC

HYPRE 76

Euclid is a family of Incomplete LU methods for sparse linear systems

- **Interface:** SStruct, FEI, IJ
- **Matrix Class:** ParCSR
- Obtains scalable parallelism via local and global reorderings
- Good for unstructured problems
- <http://www.cs.odu.edu/~hysom/Euclid>



CASC

HYPRE 77

Getting the code

<http://www.llnl.gov/CASC/hypre/>



- The User's Manual and Reference Manual can be downloaded directly
- A short form must be filled out prior to download (This is just for our own records)

CASC

HYPRE 78

Building the library

- Usually, *hypre* can be built by typing `configure` followed by `make`
- Configure supports several options (for usage information, type '`configure --help`'):
`'configure --enable-debug'` - turn on debugging
`'configure --with-openmp'` - use openmp
`'configure --with-CFLAGS=...'` - set compiler flags
- Release now includes example programs!

CASC

HYPRE 79

Calling *hypre* from Fortran

- C code:

```
HYPRE_IJMatrix A;
int          nvalues, row, *cols;
double       *values;

HYPRE_IJMatrixSetValues(A, nvalues, row, cols, values);
```

- Corresponding Fortran code:

```
integer*8      A
integer        nvalues, row, cols(MAX_NVALUES)
double precision values(MAX_NVALUES)

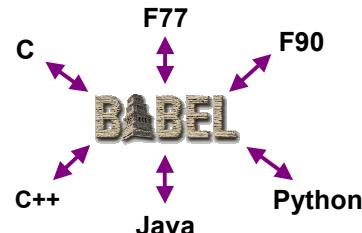
call HYPRE_IJMatrixSetValues(A, nvalues, row, cols, values)
```

CASC

HYPRE 80

The *hypre* library now features the Babel language interoperability tool

- Babel provides:
 - language interoperability
 - OO support
- Releases 1.11.0b and later use Babel for all major interface classes (except FEI)
- Currently supporting C, C++, Fortran, and Python



CASC

HYPRE 81

More about the new Babel interfaces

- Users download, build, and use *hypre* in virtually the same manner as before

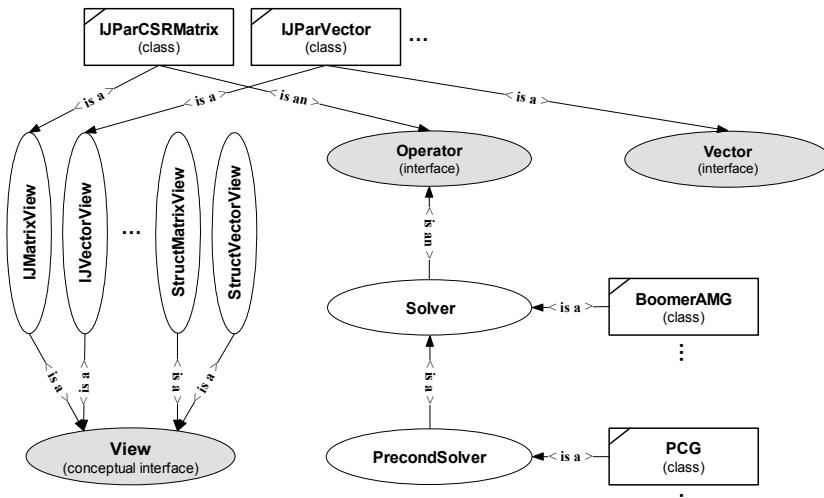
```
/* current interface */
MPI_COMM mpi_comm;
HYPRE_IJMatrix ij_A;
HYPRE_IJMatrixCreate( mpi_comm, i_lower, i_upper,
j_lower, j_upper, &ij_A);
HYPRE_IJMatrixSetObjectType(ij_A, HYPRE_PARCSR);

/* new babel interface */
bHYPRE_MPICommunicator mpi_comm;
bHYPRE_IJParCSRMatrix parcsr_A;
parcsr_A = bHYPRE_IJParCSRMatrix_Create( mpi_comm,
i_lower, i_upper, j_lower, j_upper);
```

CASC

HYPRE 82

Central to hypre's object design is the use of multiple inheritance of interfaces



CASC

HYPRE 83

Reporting bugs, requesting features, general usage questions

- Send email to:
hypre-support@llnl.gov
- We use a tool called Roundup to automatically tag and track issues

CASC

HYPRE 84



This work was performed under the auspices of the U.S.
Department of Energy by Lawrence Livermore National
Laboratory under contract no. W-7405-Eng-48.

UCRL-PRES-231999

[CASC](#)

HYPRE 85