SAND2007-7112C

Introduction to FEI (Finite Element Interface to linear solvers)

Alan Williams

Sandia National Laboratories

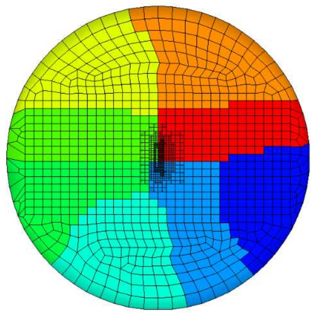
TUG

Nov. 06, 2007

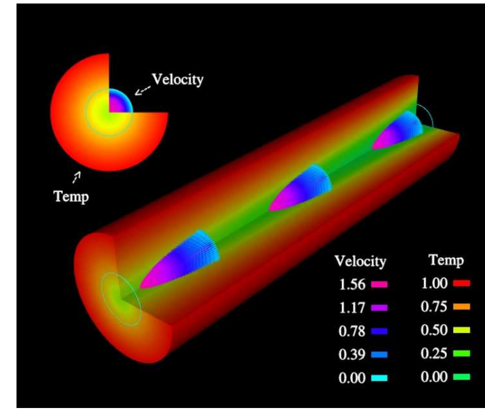
Sandia National Laboratories

Linear systems from implicit FEM/FVM applications

Calore:
Galerkin FEM
heat transfer,
radiation, ...



Fuego:
Pool fires,
turb. flows,...



Linear systems arise in implicit finite element formulations:

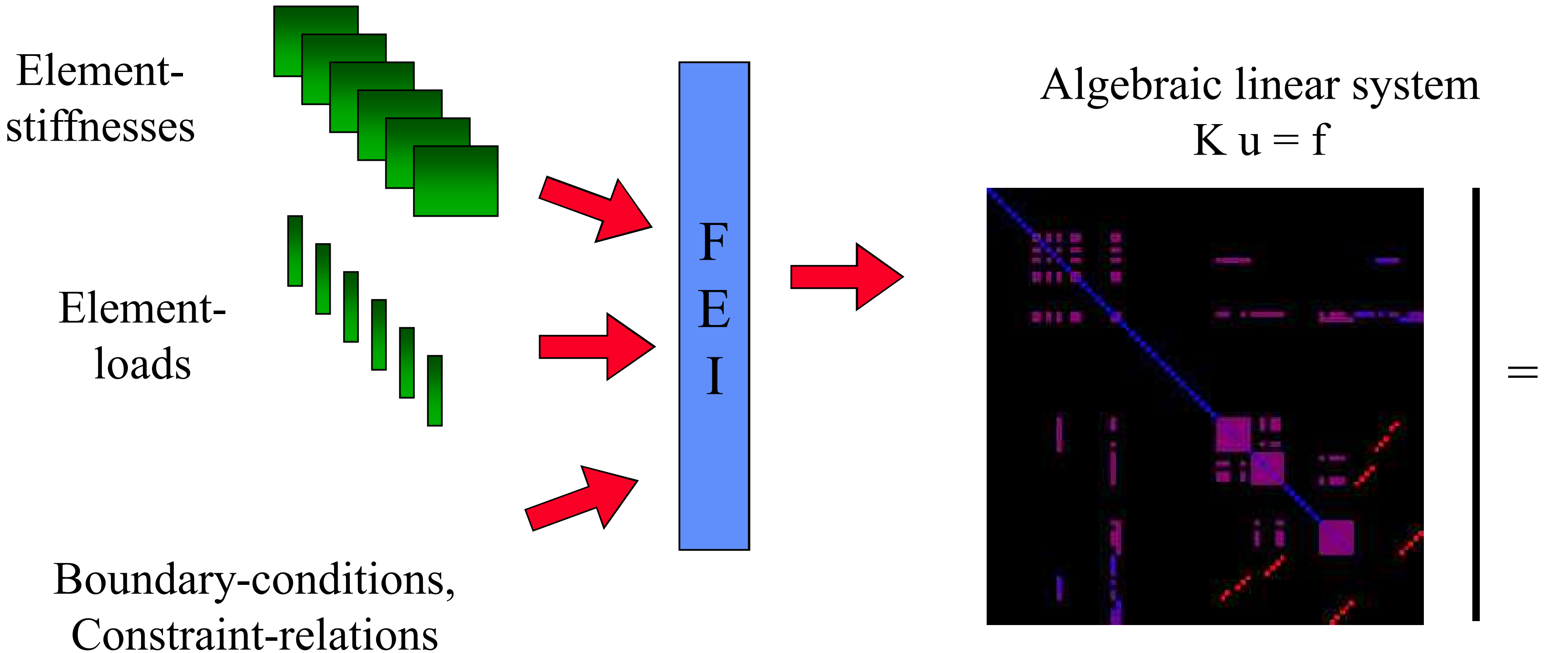
$$Ku = f$$

$$K \in \Re^{nxn}; u, f \in \Re^{n}$$

FEI is a linear system assembly library

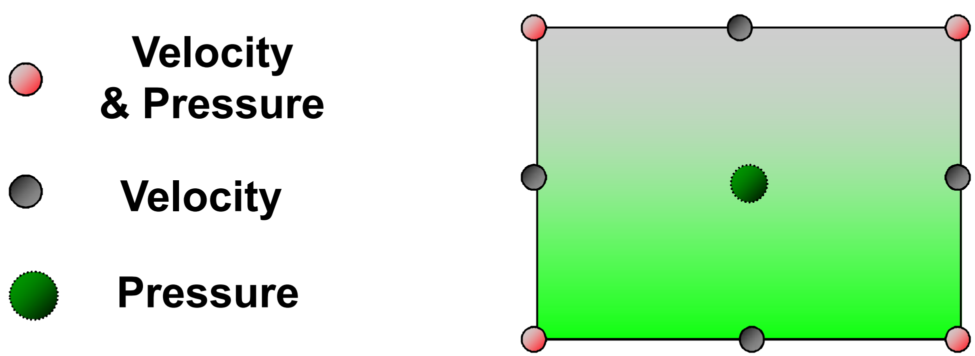
- Mediates between finite-element view (nodes, degrees of freedom) and algebraic view (equations, indices).
- Assists with parallel communications (e.g. for shared-contributions)
- Provides abstraction layer, putting a common interface on various solver libraries

FEI is a filter -- finite-element data to linear algebra data



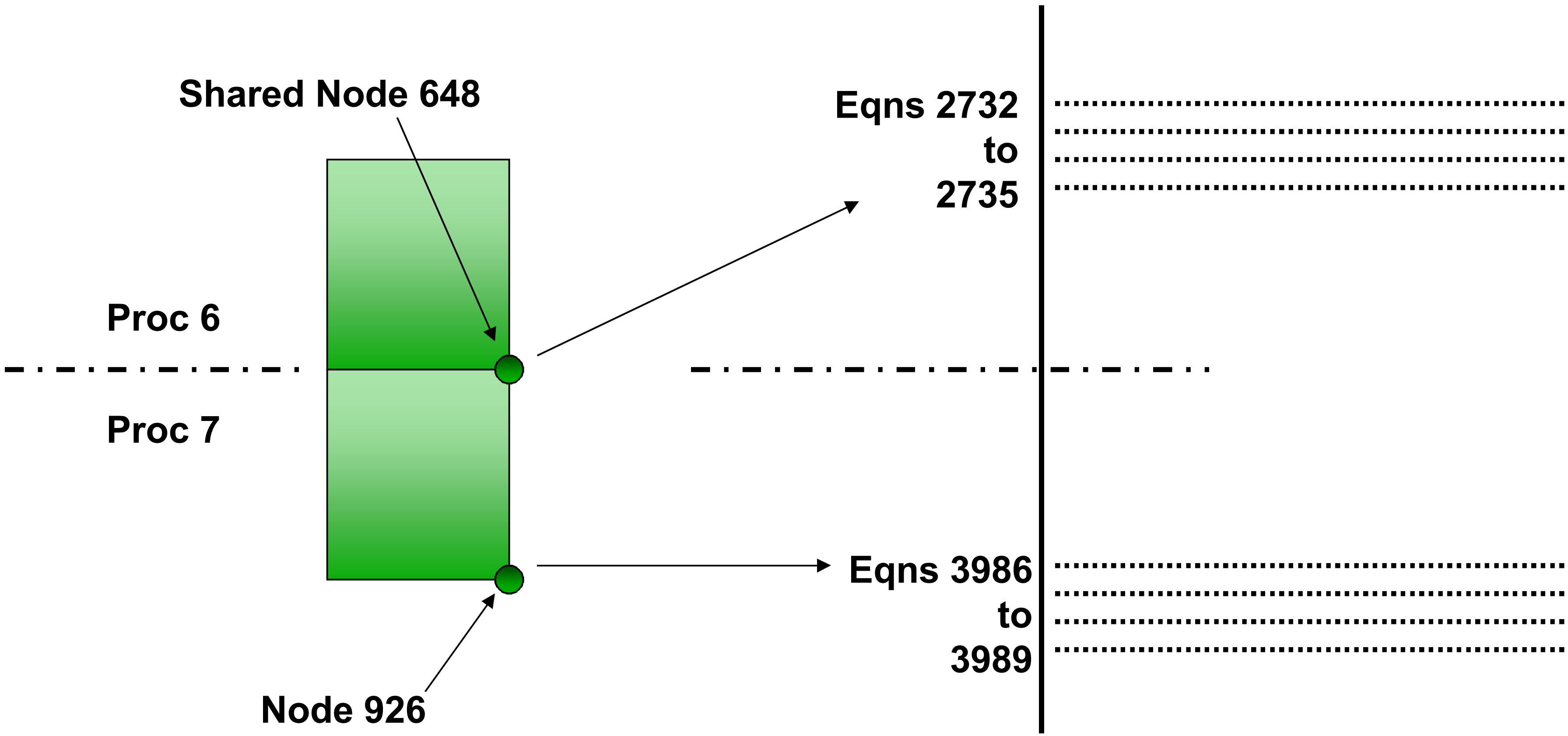
Assembly from multi-physics problems

- Solution ‘fields’ are collections of scalars
  - e.g., temperature scalar, displacement vector
- Define arbitrary mixture of fields-per-node on elements
- Define element-topologies for blocks of elements

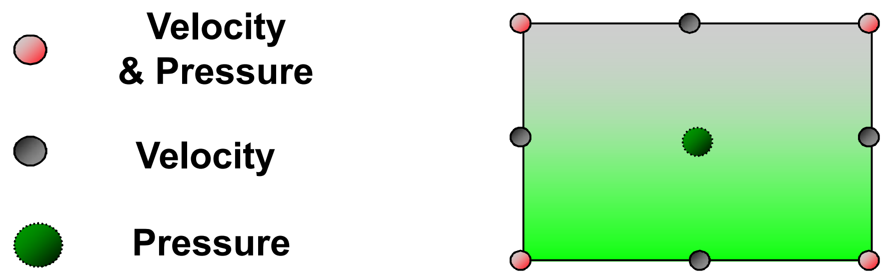


Parallel mappings

Element-based mesh decomposition ⟺ **Equation-based algebraic decomposition**

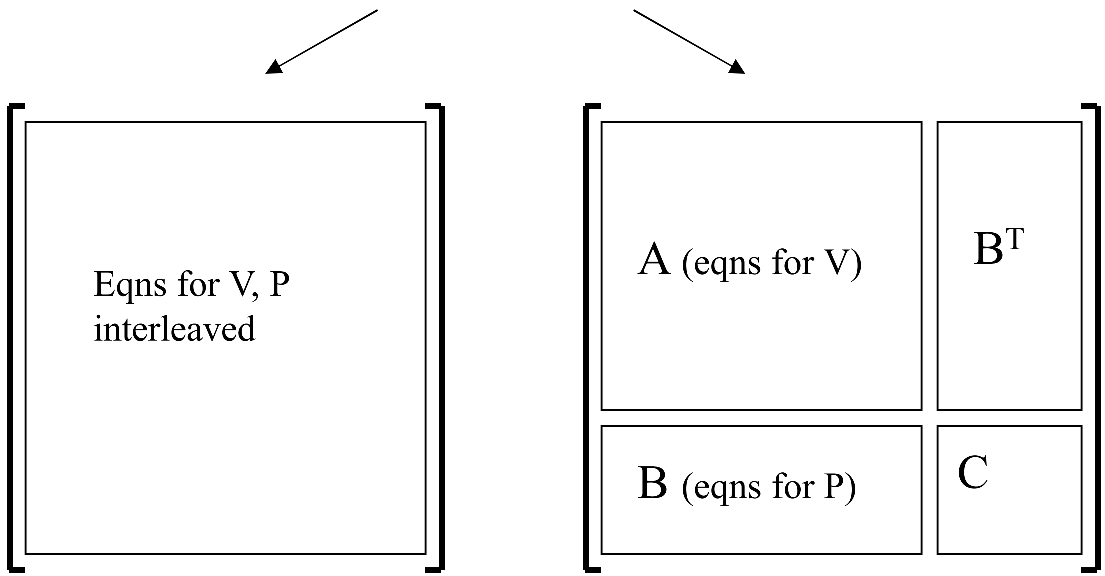


Assembly from multi-physics problems



Filtering options:

Single global matrix



Separate partitioned matrix blocks

Constraint Relations

Solve system subject to algebraic constraints,
enforced by Penalty or Lagrange Multiplier formulation

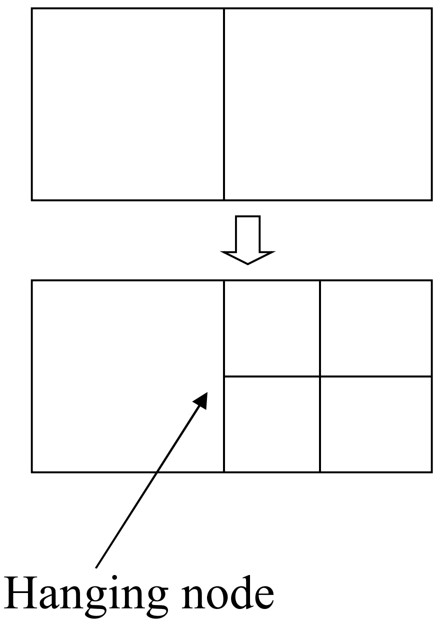
$Ku = f, \quad K$ is $n \times n$

$Cu = g, \quad C$ is $c \times n$, c is num-constraints

Penalty formulation: contributions to existing matrix structure.
Lagrange Multiplier formulation results in partitioned system

$$\begin{vmatrix} K & C^T \\ C & 0 \end{vmatrix} \begin{vmatrix} u \\ v \end{vmatrix} = \begin{vmatrix} f \\ g \end{vmatrix}$$

Common source of constraints:
Hanging nodes from adaptive mesh refinement:
2 quad elements:



Constraint Relations (continued)

Slave-constraint reduction (from paper by St. Georges et al)

If constraints represent master-slave relations,

$$C = \begin{bmatrix} D & -I \end{bmatrix}$$

Split solution space 'u' into dependent and independent unknowns

$$u_d = D u_i + h \qquad K = \begin{matrix} K_{ii} & K_{id} \\ K_{di} & K_{dd} \end{matrix}$$

Then reduced matrix is given by:

$$K_R = K_{ii} + K_{id} D + D^T K_{dd} D$$

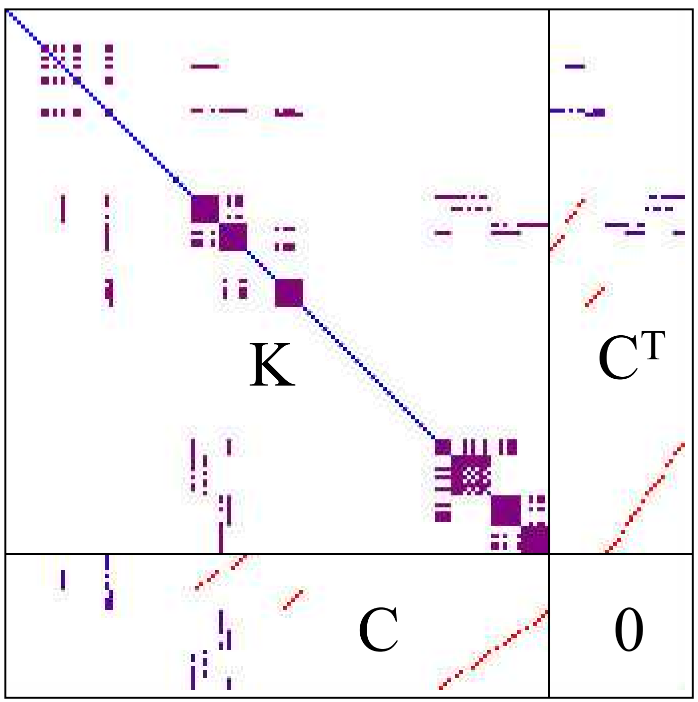
FEI can create reduced matrix from element-level contributions

Sandia National Laboratories

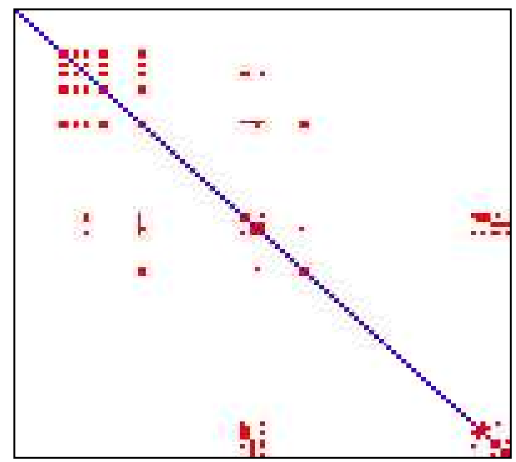
Constraint Relations (continued)

FEI filtering option:
Lagrange Multipliers or Slave reduction

139 DOF, 38 constraints
Matrix: 169x169

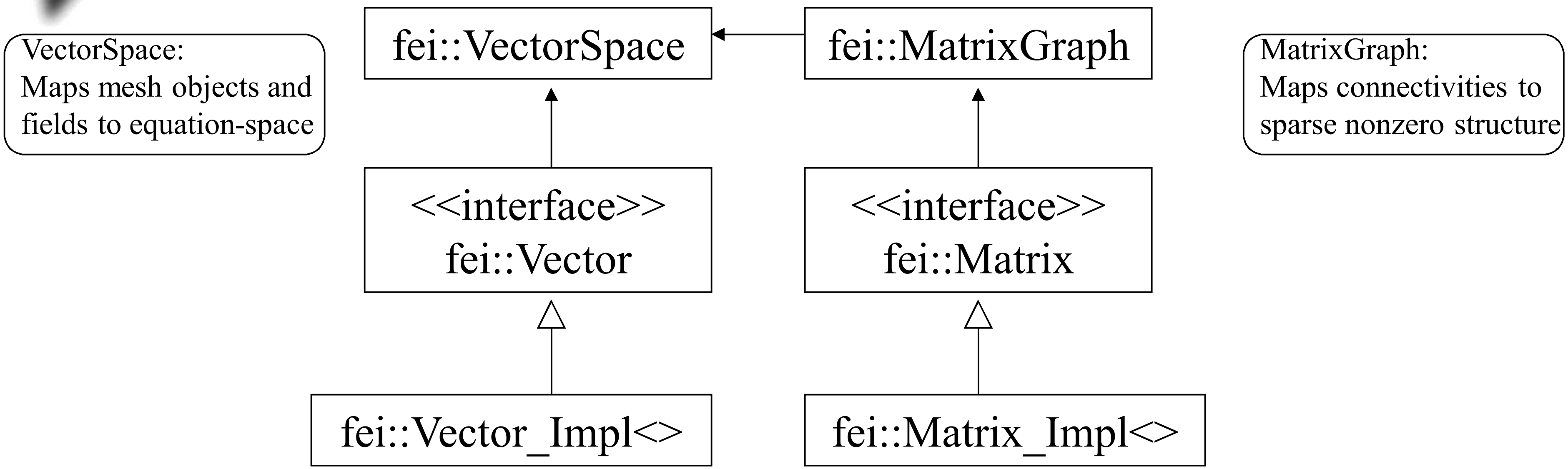


139 DOF, 38 'slaves'
Matrix: 101x101



Key point: reduced matrix is SPD, augmented matrix is indefinite

FEI classes & interfaces



Example: instantiation for a Trilinos matrix object:

```
fei::Matrix* matrix = new fei_impl::Matrix<Epetra_CrsMatrix>
```

The connection between fei_impl::Matrix<> and Epetra_CrsMatrix is made using a Traits class.

Libraries available through FEI

Trilinos	SNL	
HYPRE		LLNL/CASC
PETSc		ANL
FETI		SNL
Prometheus		UC-Berkeley, now Columbia?

•For each solver library, a "glue layer" is required to connect it to FEI.

•The support layer for Trilinos, PETSc is included with FEI code distribution.

•HYPRE, FETI and Prometheus provide their own FEI support layer.

Common question: does FEI impose significant overhead?

For most matrix/vector coefficient contributions, FEI simply passes pointers through to underlying solver-library data structures.

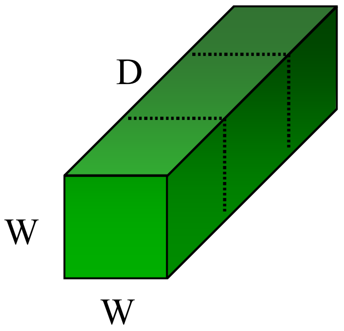
However, there is work done in creating matrix-graph (nonzero sparsity pattern) as well as parallel communication for shared-contributions, etc.

This is work that would have to be done elsewhere if FEI wasn't doing it, but it is worth verifying that FEI does it efficiently.

Test problem: 3-D "beam" of 8-node Hexahedral elements.

Dimension: W x W x D elements

In parallel, sliced across 'D' dimension.
Each proc has W x W x (D/nprocs) elems.



Scalability measurement

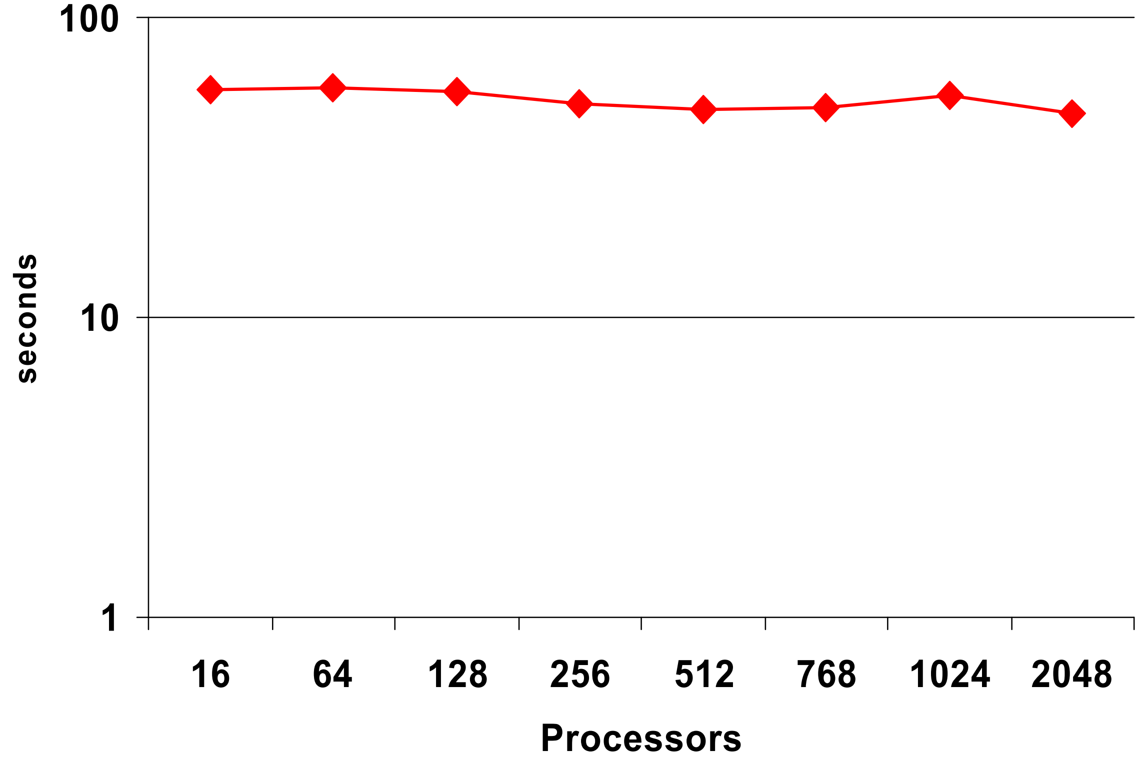
FEI assembly time, ASC "Red Storm"

(Includes Trilinos matrix assembly)

"Beam" of 8-node Hex elements

~1M Eqns/proc

Procs	Eqns
16	16.9M
64	67.8M
128	135.5M
256	271.0M
512	542.0M
768	813.0M
1024	1.084B
2048	2.107B



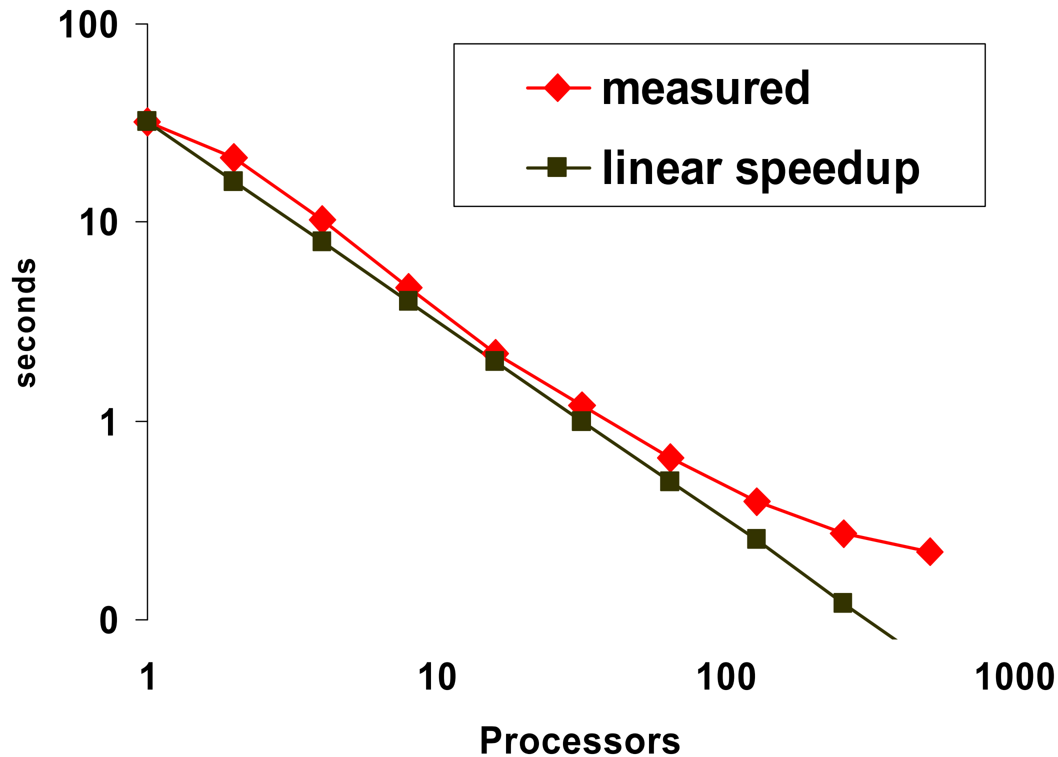
Speedup measurement

FEI assembly time, ASC "Red Storm"

(Includes Trilinos matrix assembly time)

"Beam" of 8-node Hexes. Fixed-size problem, 1M Eqns

Procs	Eqns/proc
1	1.04M
2	521K
4	261K
8	132K
16	67K
32	34K
64	18K
128	10K
256	6K
512	4K



Summary

- FEI:
  - Helps with mappings between finite-element and algebraic points-of-view

  - abstraction layer makes linear system assembly look the same for all solver libraries (not including solver control parameters, of course).

  - filtering operations provide useful solution capabilities for constrained problems, multi-physics problems, etc.

  - parallel communications and mappings have been proven to be efficient and scalable on large numbers of processors.

- Note: Trilinos also contains other abstraction layers:
  - Thyra interfaces abstract operators/vectors from data,
  - Amesos package is an interface to several sparse direct solvers such as UMFPACK, SuperLU, etc.