Parallel Multigrid Algorithms and Implementations for Unstructured Finite Element Applications

Mark Adams (madams@cs.berkeley.edu)

James Demmel

R.L.Taylor

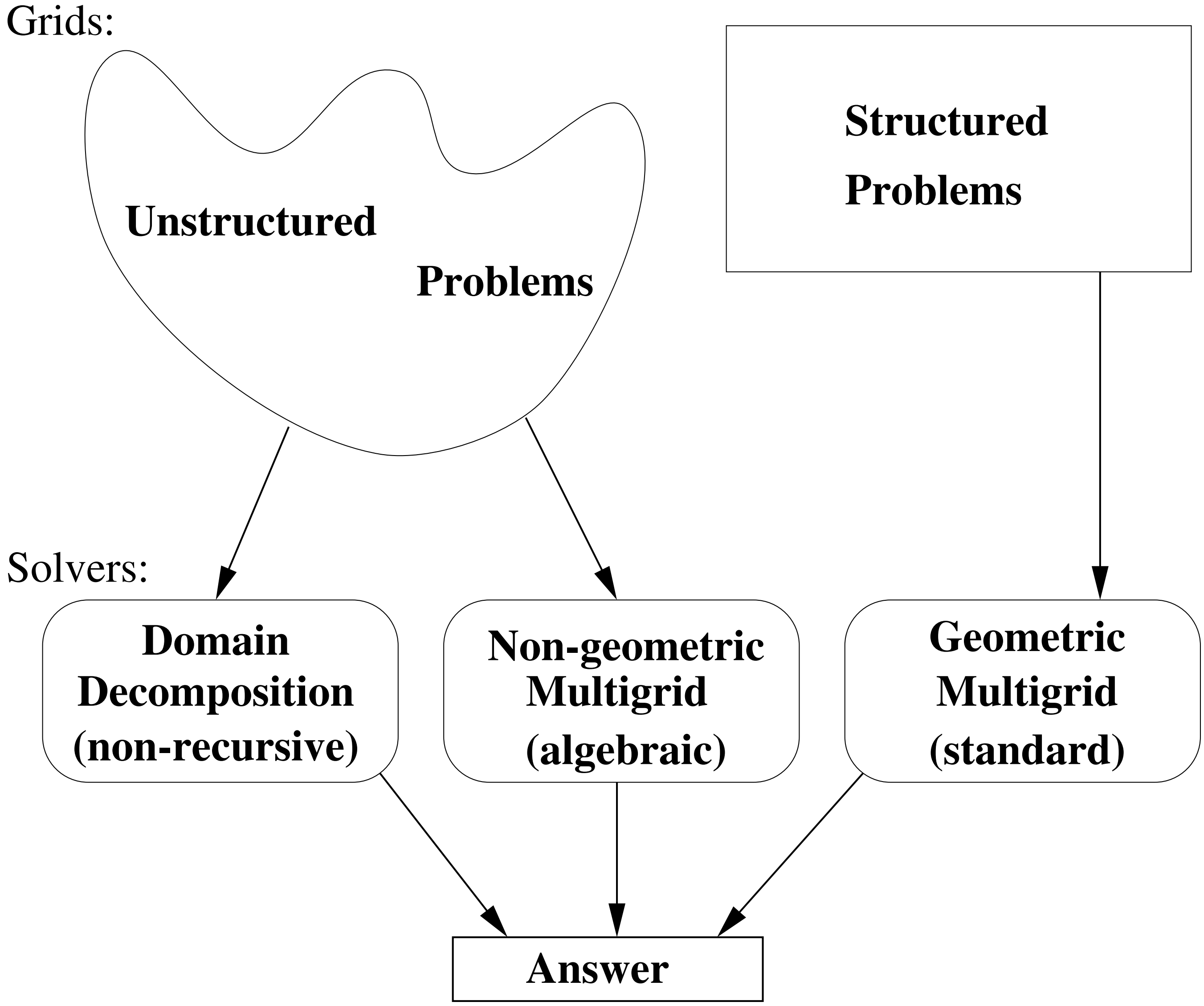
Funded by DOE ASCI

March 22, 1998

Talk outline

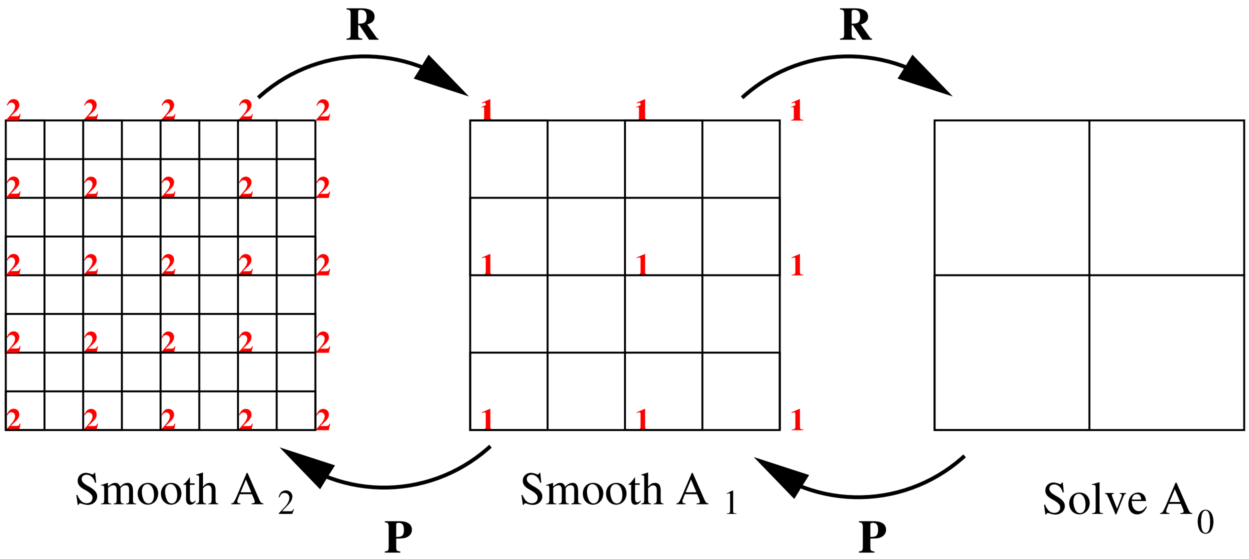
- Introduction to multigrid
- Coarse grid construction
- Our parallel FE architecture - Athena
- Performance results on large deformation elasticity and plasticity problems with more than 26 million degrees of freedom
- Conclusion/ future work

Approximate PDE solver taxonomy



Multigrid

Multigrid on structured mesh



Our Method and Implementation

- Given an unstructured FE mesh automatically generate
  - Coarse Meshes
  - Interpolation and restriction operators
- Given a stiffness matrix and RHS
  - Automatically generate coarse grid operators
  - Solve using multigrid
- Used for nonlinear problems and contact problems
- All phases scalable
- Solved up to 26M dof problems on 640 PE Cray T3E with 55% parallel efficiency
- 26M dof on 4x160 PE ASCI blue-pacific with parallel 50% efficiency

Prometheus Algorithm Outline

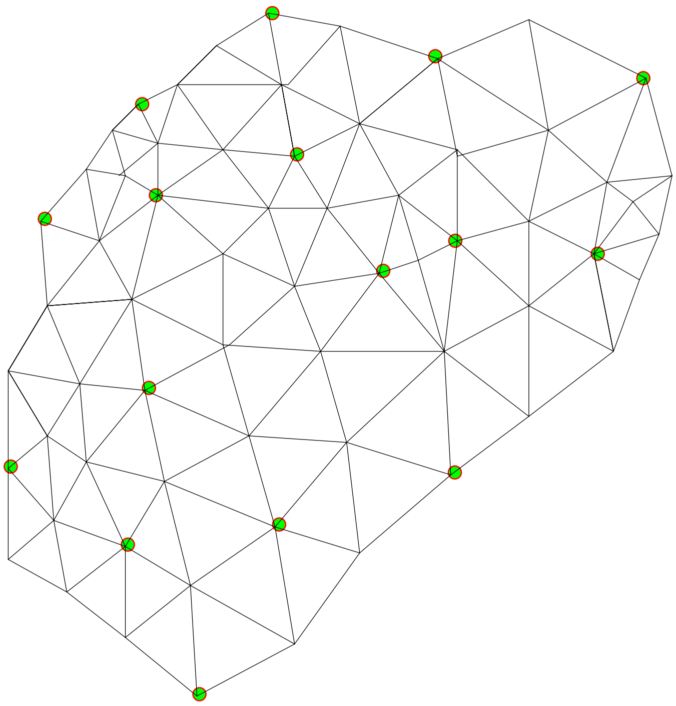
- Based on Guillard (1992)
- Chan and Smith (1994)
- Mixed geometric/algebraic MG

1. Coarse Meshes
   - Maximal Independent Sets (MIS) to get vertex set - New algorithm, $O(1)$ PRAM complexity
   - MIS heuristics to preserve geometry in coarse meshes
   - (Delaunay) Mesh the vertex set. Very hard to parallelize “right”, but not necessary (can just do locally)

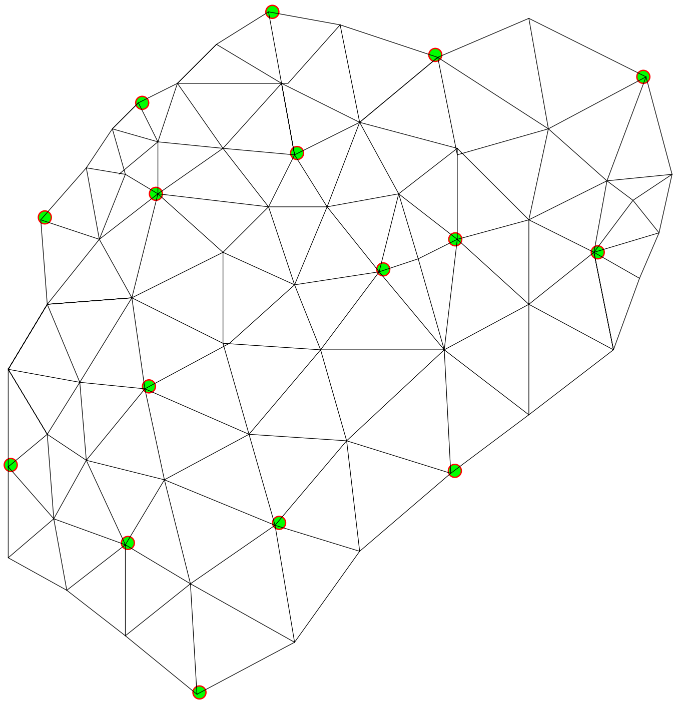
2. Interpolation, Restriction operators
   - Use FE shape functions, easy to parallelize
3. Coarse Grid Operators
   - Galerkin: $A_c = R \cdot A_f \cdot R^T$
   - Hard to make fast (lots of indirection), easy to parallelize
4. Solve using multigrid
   - Load Balance using ParMetis
   - Accommodates CLUMPs (Cluster of SMPs)
   - Coarse Grid Agglomeration
   - Use “MG Harness” from PETSc

1. Maximal independent sets

- Independent set I of Graph $G = (V, E)$:
  If no two vertices in I are connected by an edge
- Maximal Independent Set (MIS) I of G:
  No independent, proper superset of I exists
- **Not** a Maximum Independent Set:
  No larger independent set exists -
  NP hard problem



Serial greedy algorithm



forall $v \in V$

  if v is not done **then**

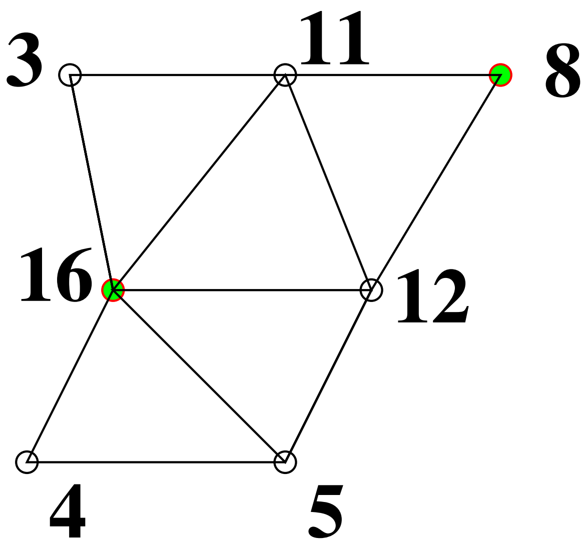
    Select v

    forall neighbors $v1$ of v

      Delete $v1$

$MIS \leftarrow$ all *selected* vertices

Randomized parallel algorithms - Luby



while(any vertex v is not done)
  forall (vertices on proc p)
    if v is not done **then**
      if all neighbors of v are: deleted **or**
        have lower random number **then**
        $Select(v)$

- $I \leftarrow \{$ selected $v \in V\}$

Complexity results

- P-RAM model.
- $\frac{n}{P} = O(1)$ - thus $O(n)$ processors.

Brief history and our contribution

- Luby 86 - (Random) general graphs - $O(\log^2 n)$
- Goldberg - (bits) bounded degree - $O(\log^* n)$
- Jones - (Random) FE graphs - $O(\log n / \log\log n)$
- Optimal - FE graphs - $O(1)$
- Adams 98 - (Partitioned) FE graphs - $O(1)$*

* Our Additional Assumptions

- $\frac{n}{P} > C$ (e.g. $C \approx 50$ for 3D hex. meshes)
- "Partitioned well" - feasible for FE graphs

Our parallel algorithm - same as "random" algorithms except ...

while(any vertex v is not done)
  forall (vertices on proc p)
    if v is not done **then**
      if all (undeleted) neighbors, of v, have
        lower or equal processor number
        $\langle$ lower random number $\rangle$
        then
          $Select(v)$

- $I \leftarrow \{$ selected $v \in V\}$

Complexity

- Bounded degree graphs, complexity of each outer iteration - $O(1)$
- Total complexity: bounded by number of outer iterations.
  - Randomized: length of Monotonic Paths.
  - Our algorithm: length of Weaving Monotonic Paths

Weaving monotonic path (WMP)

Definition: A weaving monotonic path: $(v_1, v_2, ..., v_i, v_j, ...)$ in which each consecutive pair (v_i, v_j) satisfies

$$v_i.proc < v_j.proc$$

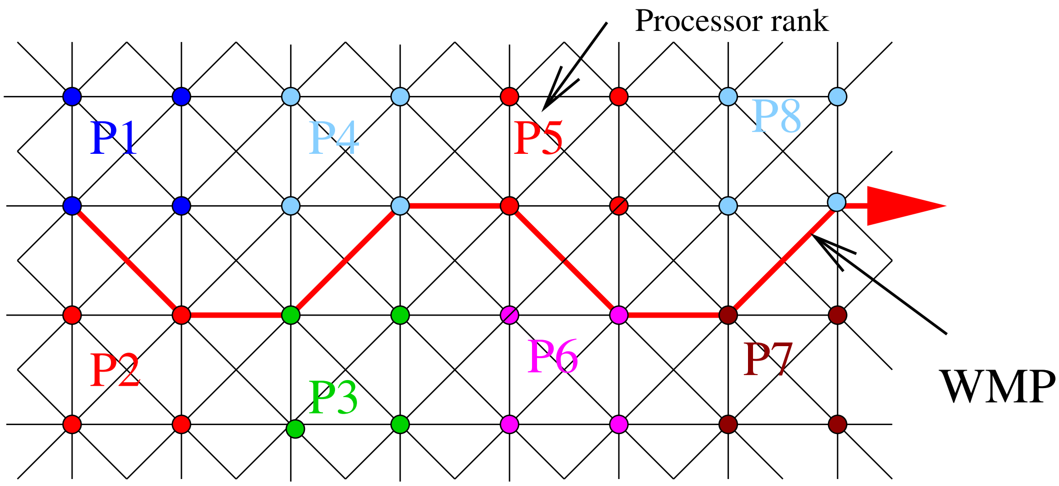


Figure 1: WMP with 2D regular 2 by 2 partitions

Our WMPs

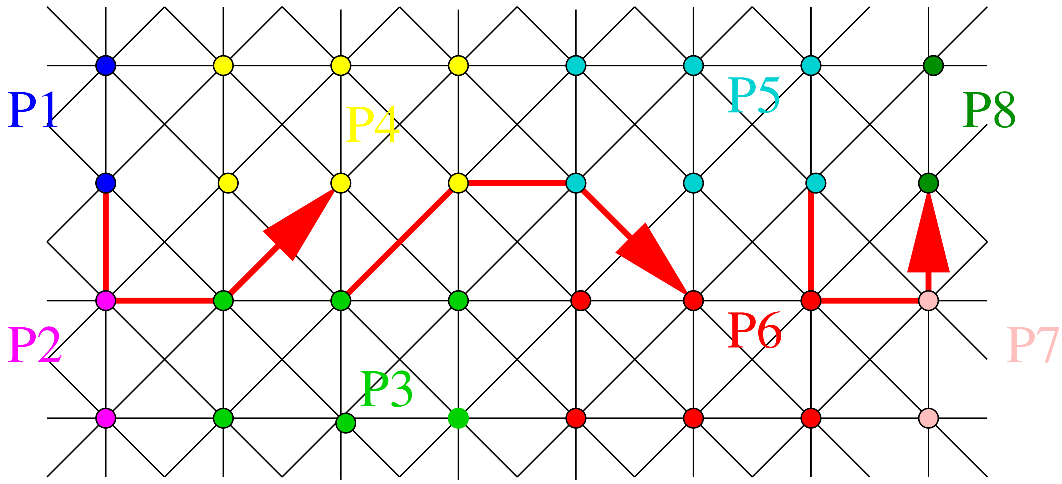


Figure 2: WMP on 3 by 3 partitions

THEOREM. The running time in the P-RAM computational model, of our "partitioned" algorithm, is bounded from above by the maximum length weaving monotonic path in G.

OBSERVATION: "Good" partitionings will have short WMPs.

Bound on WMPs

- Notice: partitioners for FE meshes minimize Surface to Volume ratio of partitions, e.g. by minimizing edge cuts.
- Lower Bound on $\frac{n}{P}$ to keep WMPs from "punching" through partitions:
  - WMPs live along partition interface lines.
- Require Lower bound on $\frac{n}{P}$ to keep interface lines longer than $D + 1$:
  - WMPs die and can only live around partition intersections: length $D+1$ for "well partitioned" graphs.
- Partitions used in practice easily satisfy "well partitioned".

Mis performance results

	Processors									
Vertices	8	16	24	32	40	48	56	64	72	80
427	3	3	3	3	4	4	4	4	6	6
1270	2	4	3	3	4	4	4	3	4	3
2821	3	3	4	3	3	3	4	3	4	4
5296	2	2	3	3	3	3	4	3	3	3
8911	3	3	4	3	4	3	4	3	3	3
13882	3	3	3	3	3	3	3	3	3	3

Figure 3: Average number of iterations

Expected number of iterations = 4 (with "optimal" partitions)

Coarse grids (MIS heuristics)

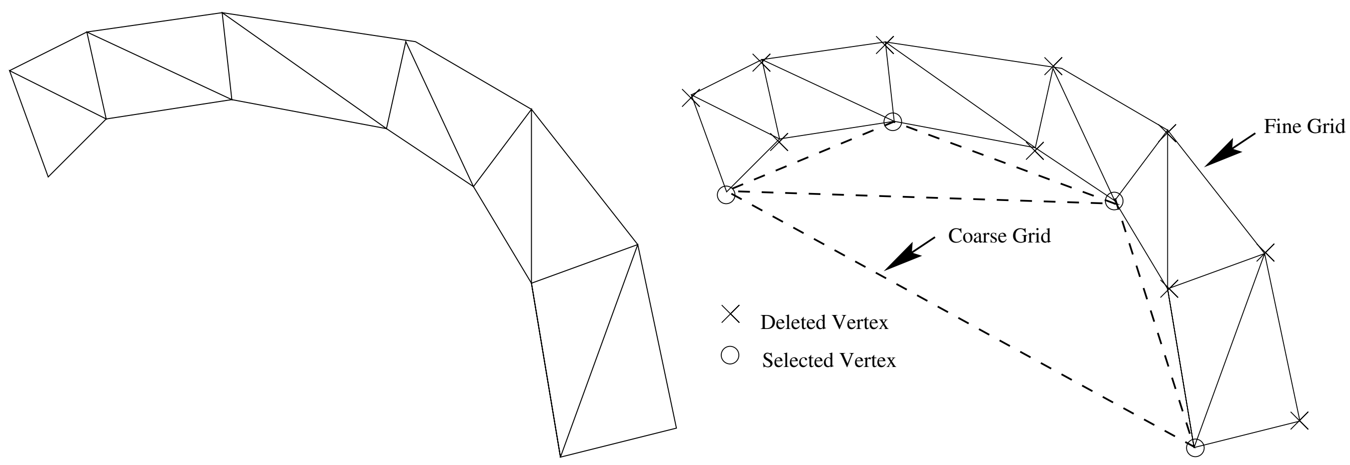
- **Our approach:**
  - Find an accurate coarse representation of the **geometry**
  - Possibly add heuristics for particular applications (eg, semi-coarsening for CFD)
- Heuristics for MIS
  - Order vertices for MIS: { corner, edge, surface, interior }
  - Modify graph by deleting graph edges between different features within a class
  - Move nodes to cover "lost" vertices

Identify important geometric features and (try to) preserve them ...

- Identify "faces"
- Classify vertices: corner, edge, surface and interior
  - Corner vertex: touch 3, or more faces
  - Edge vertex: touch 2 faces
  - Surface vertex: touch only one face
  - Interior vertex: all the rest

- Heuristics for MIS
  - Order vertices for MIS:
    { corner, edge, surface, interior }
  - Modify graph by deleting graph edges between different features within a class
  - Move nodes to cover “lost” vertices

Pathological example:



Example - classify vertices - modify graph

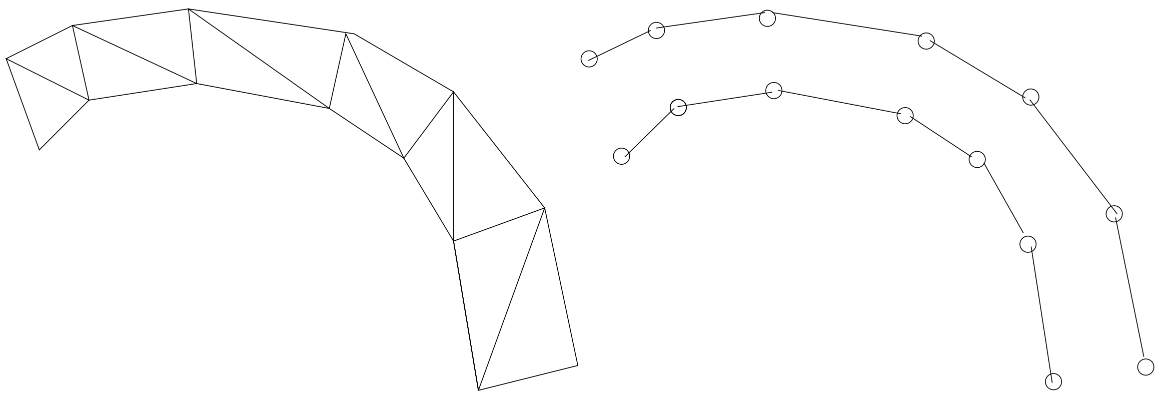


Figure 4: Modify graph

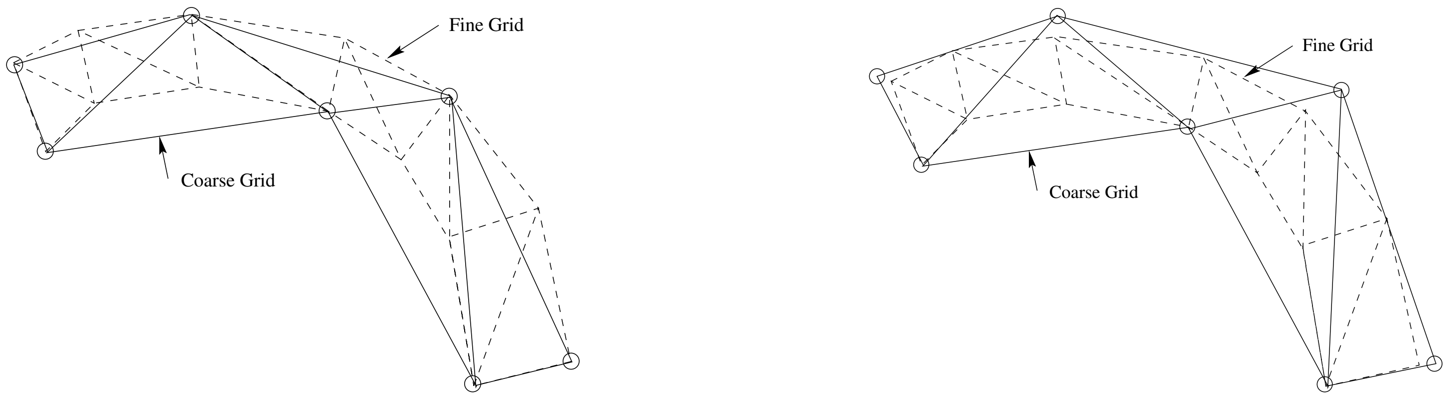
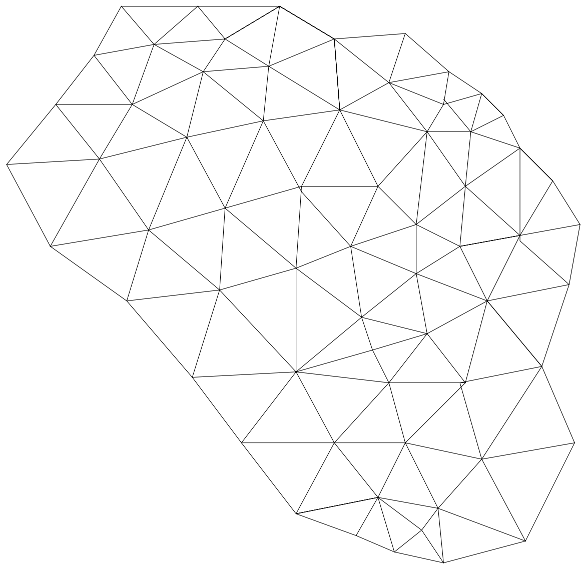


Figure 5: New mesh - fixed mesh

Simple face identification algorithm



Example of Prometheus meshes

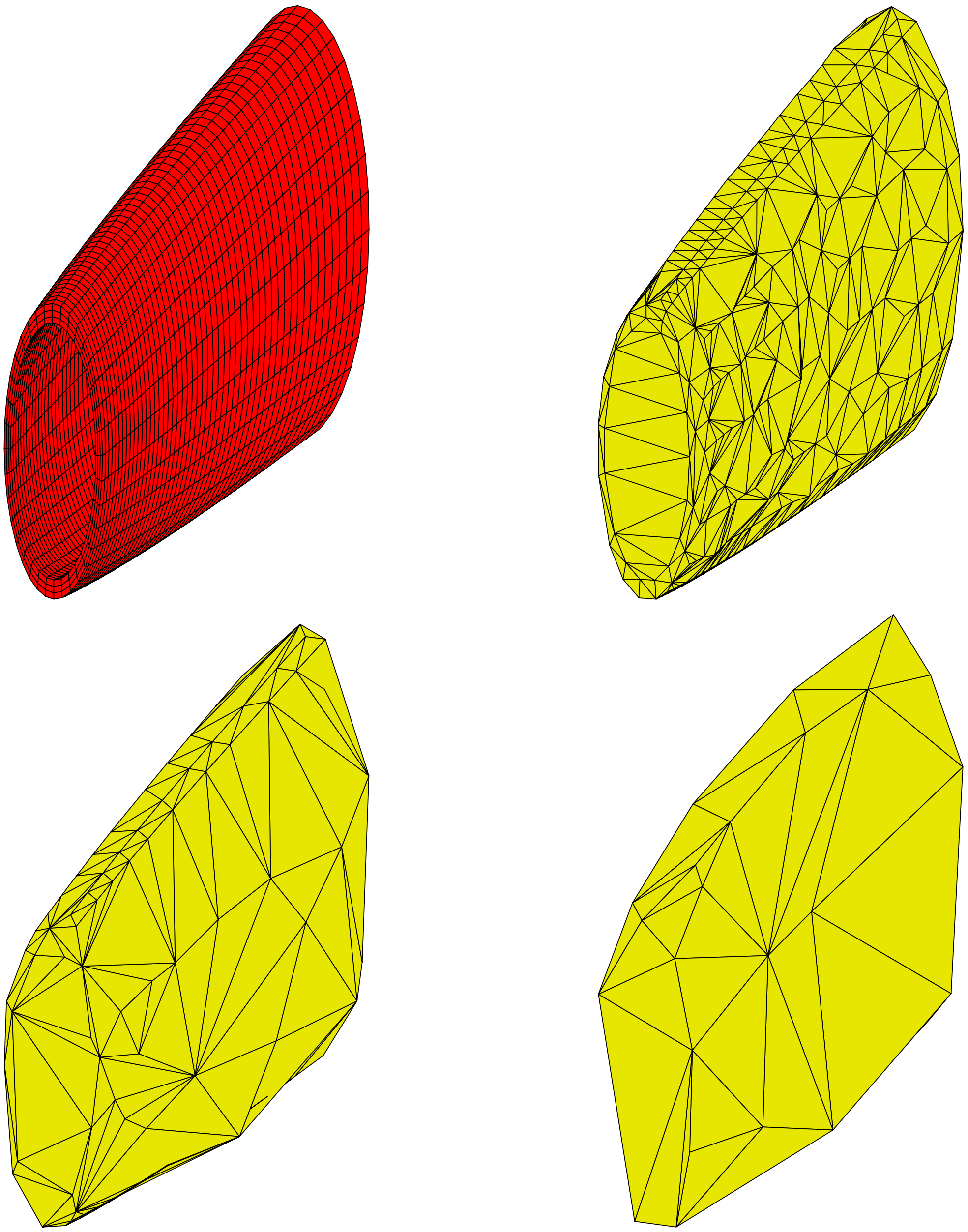
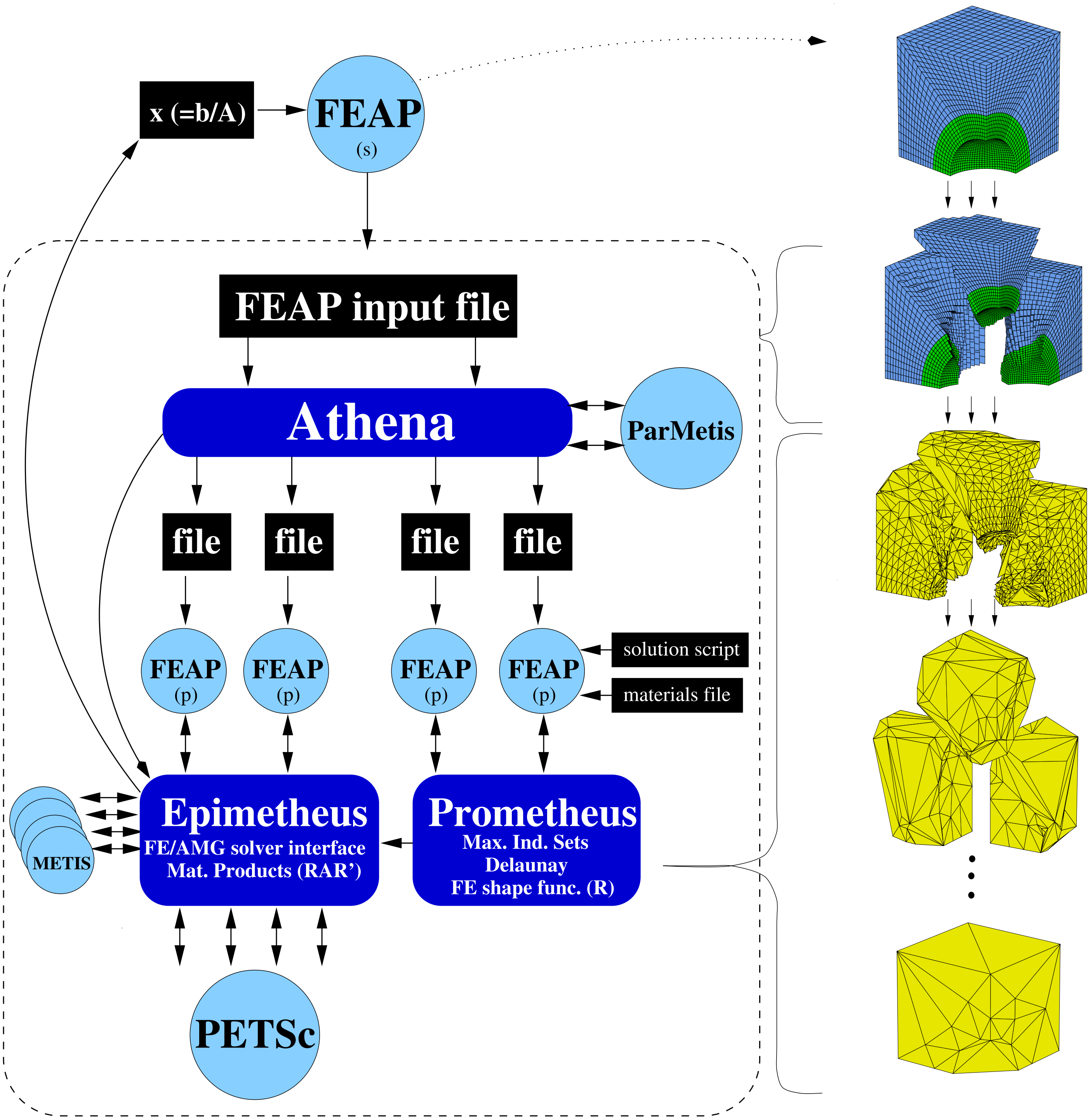


Figure 6: Sample input grid and coarse grids

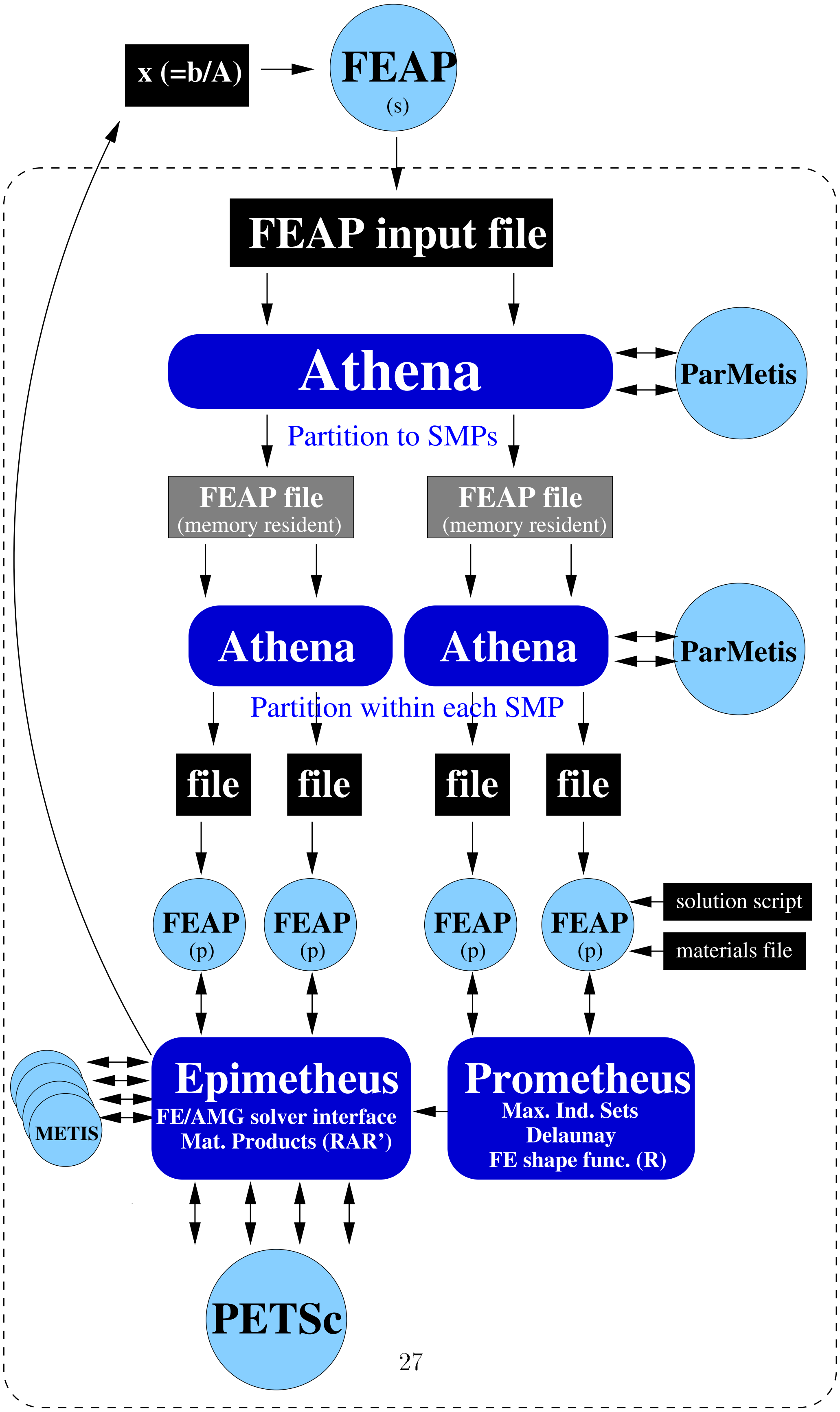
4. Athena components

- FEAP (U.C. Berkeley): Serial research FE implementation
  - Run on each processor, in parallel to form A and b - calls Prometheus to solve for x
  - State of the art FE technology
  - Used serially to create input meshes, visualize results, and verify correctness
- PETSc (ANL): Parallel programming environment for PDE solvers.
  - Unstructured parallel numerical kernels
  - Parallel (MPI) development environment
  - Multigrid solver framework
  - Krylov subspace methods
- ParMetis/METIS (University of Minnesota)
  - Parallel mesh partitioning

Athena architecture
(Athena/Epimetheus/Prometheus)



Athena architecture for CLUMPs



Coarse grid agglomeration

- How many processors to use on each grid?
  - Top grid: one processor as only have serial direct solver (now)
  - Fine grid will use all processors
  - How many processors used in each middle grid?

Issue: on grid i with few vertices per processor: $\frac{n_i}{P}$ <??? - faster to use **fewer** processors

How many processors to use ? … model method:

If relative cost in a grid, with n vertices, is: two dot products and a matrix-vector product

$$C_{MG}(P,n) = 2 \cdot C_{dot}(P,n) + C_{matrix-vector}(P,n)$$

dot cost:

$$C_{dot}(P,n) = \frac{2 \cdot b \cdot n}{P \cdot fp_1} + (\alpha + \beta)\log(P)$$

with b the dof per vertex, and matrix-vector product cost:

$$C_{matrix-vector}(P,n) = \frac{2 \cdot 27 \cdot b^2 \cdot n}{P \cdot fp_2} + 6 \cdot b \cdot \beta \cdot \frac{n^{\frac{2}{3}}}{P} + 22 \cdot \alpha$$

$\frac{d}{dP} C_{MG}(P,n) = 0$

$$\frac{d}{dP} C_{MG}(P,n) = \frac{2(\alpha+\beta)}{P} - 4 \cdot b \cdot \frac{n}{P^2 \cdot fp_1} - \frac{27 \cdot b^2 \cdot n}{P^2 \cdot fp_2} - \frac{4 \cdot b \cdot \beta \cdot n^{2/3}}{P^{5/3}}$$

Get an expression

$$P_{Opt}(n) = \frac{\frac{2 \cdot b \cdot n}{fp_1} + \frac{13 \cdot b^2 \cdot n}{fp_2} + 2 \cdot \beta \cdot n^{2/3} \cdot P_{Opt}^{1/3}}{\alpha + \beta}$$

Solve it … but models are hard so … use curve fitting

Coarse grids go faster with fewer processors ...how few?

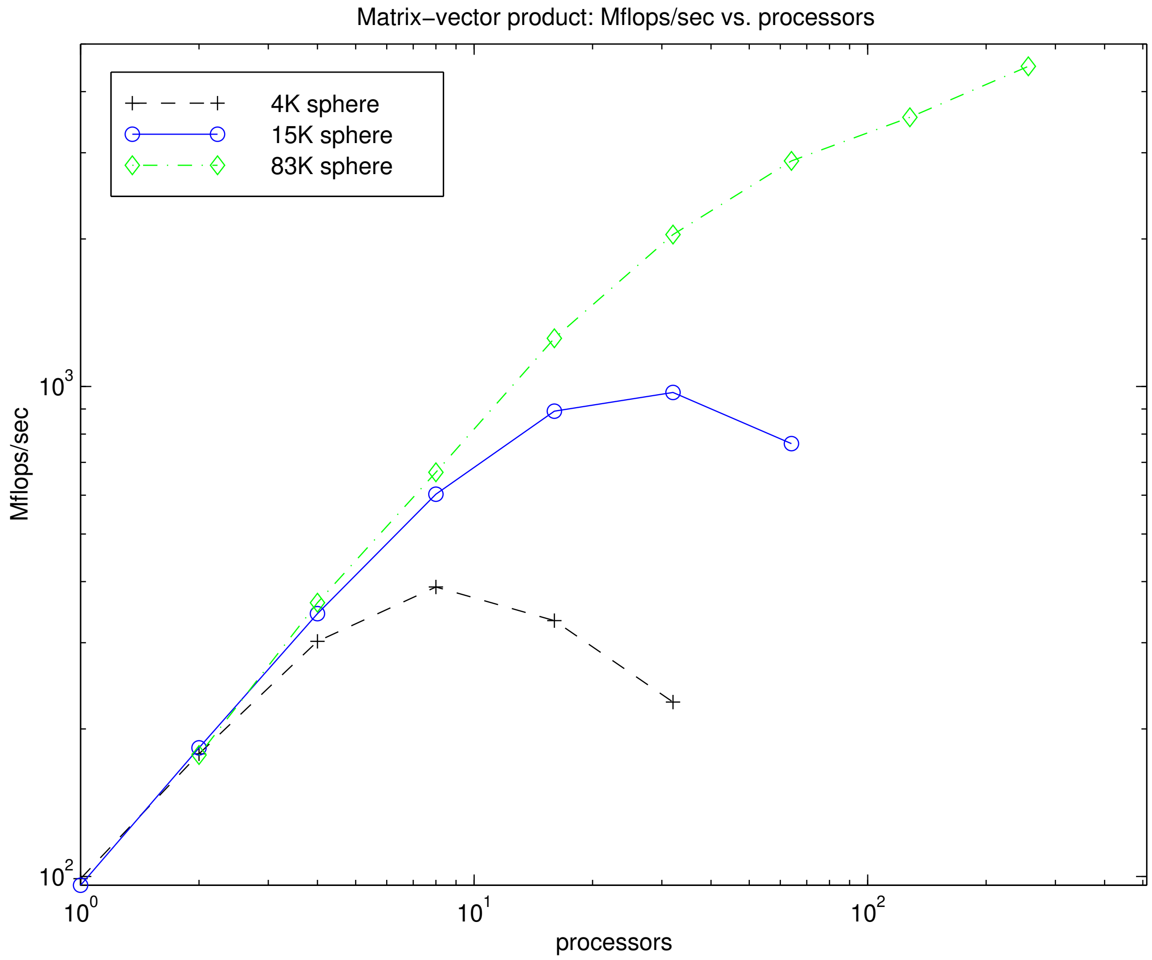
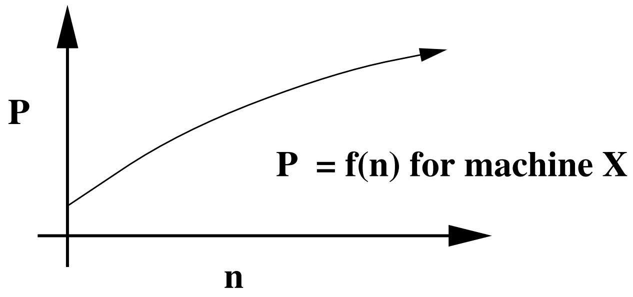
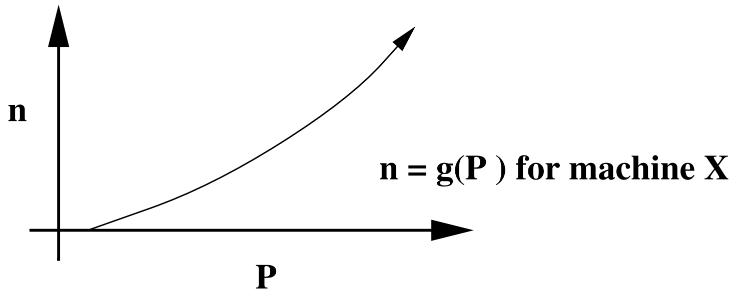


Figure 7: Mflops / sec for matrix vector product on 83K, 15K, and 4K matrix on Cray T3E

Want function: $P = f(G(V,E))$, or simply $P = f(|V|) = f(n)$



or



Approximate with a quadratic polynomial

$$AP^2 + BP = n$$

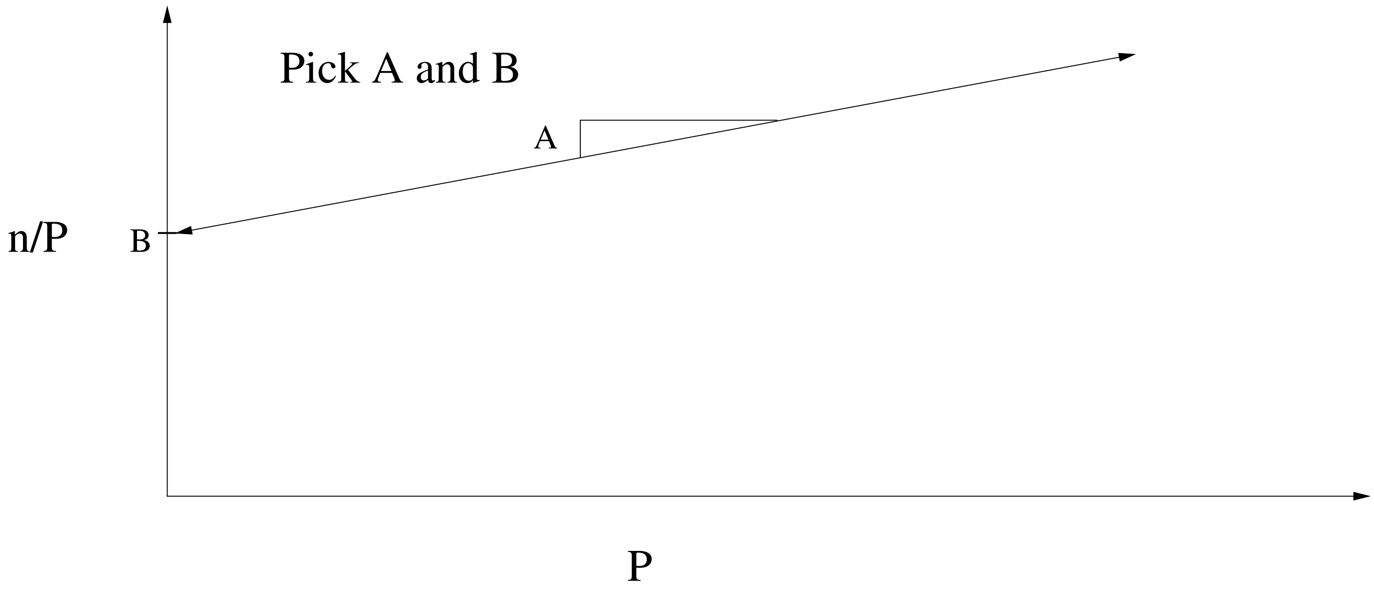
or

$$AP + B = \frac{n}{P}$$

Now pick A and B for machine X

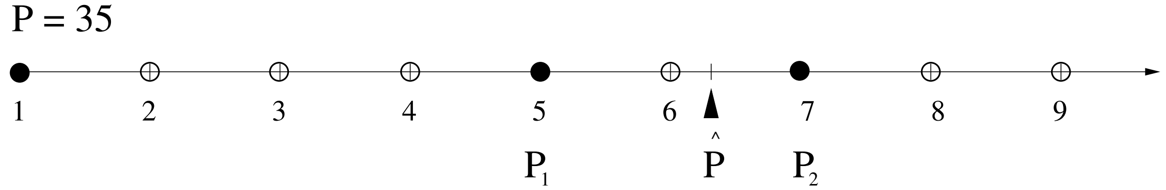
Problem: non-determinate results with current implementation on current platforms

Thus: Experiment and pick A and B informally



Given n, solve for P (pick closest feasible integer for P

$$AP^2 + BP = n$$



Performance results - Blue

3,594 K dof		Total Mflop per sec. for SLESSolve					
Included Sphere		1509			249		
grid	≈ equations	np	mat-vect.	dot	np	mat-vect.	dot
1	3,600K	**240**	3947	35	**240**	5239	42
2	314K	**120**	2505	11	**240**	2597	3
3	45K	**30**	549	8	**240**	412	0
4	6K	**10**	266	2	**240**	29	0
5	600	**1**	NA	NA	**1**	NA	NA

Table 1: Flat and "graded" processor groups, IBM PowerPC cluster

Does condition number matter?

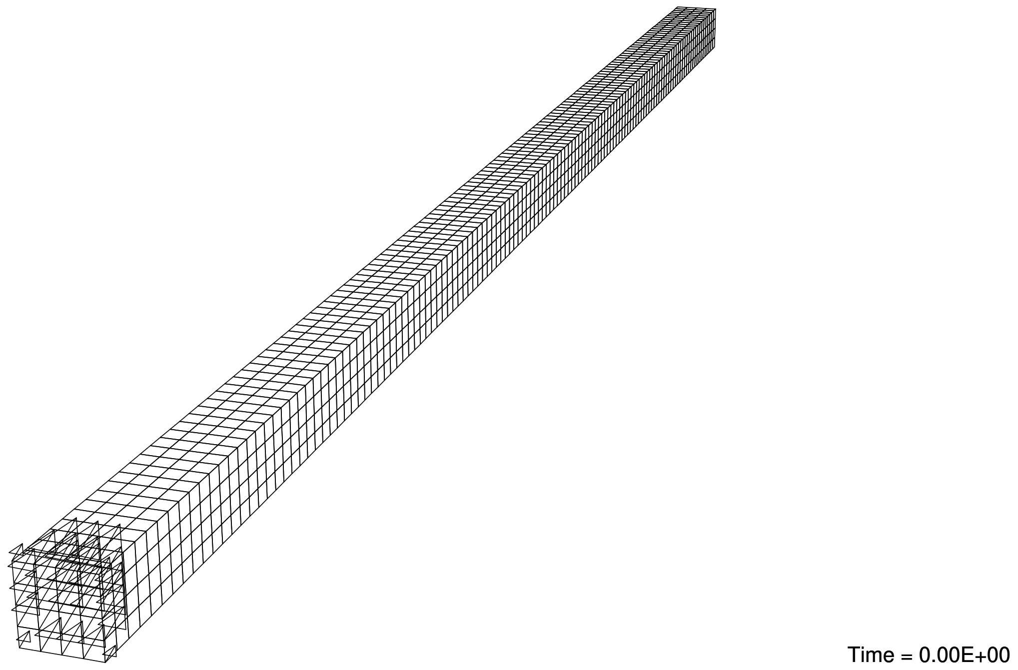


Figure 8: Soft section cantilever with uniform mesh

Parameter: $\log_{10} E_{soft}$	0	-2	-4	-6	-8
Iterations	11	11	12	13	14
Condition	$4.3 \cdot 10^8$	$5.6 \cdot 10^8$	$1.5 \cdot 10^{10}$	-	-
Unpreconditioned CG iterations	2193	2717	6738	-	-

Table 2: 60,000 dof Cantilever linear elasticity with soft section

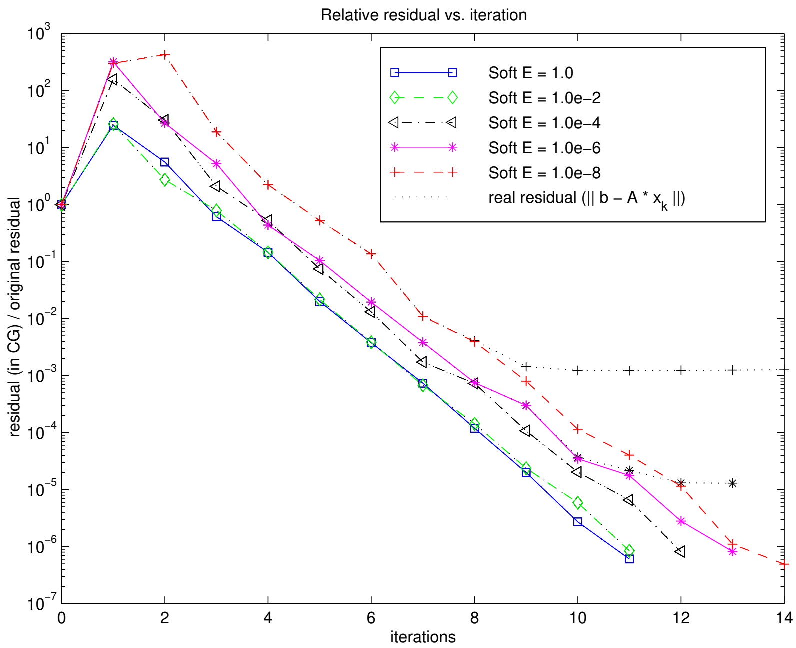
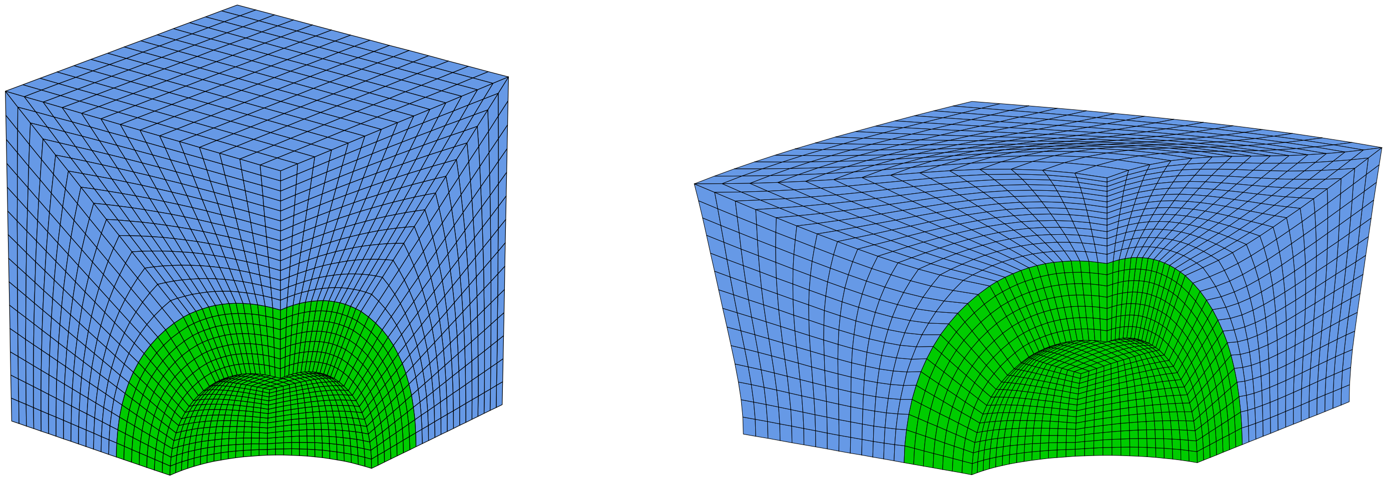


Figure 9: Residual convergence for a cantilever with soft a section

Conclusion: poor condition does not necessarily hurt convergence rate

Can Prometheus solve any "well" conditioned matrix?



13882 Vertex 3D FE mesh - Deformed Shape

Solver

- Conjugate gradient solver (accelerator for SPD)
- Full Multigrid preconditioner
- Conjugate gradient smoother (2 iterations)
- Diagonal, Block Jacobi, and additive overlapped Schwarz preconditioners
- Convergence tolerance on the relative norm of the residual 10^{-6}

Parameter: ν_{soft}	0.3	0.4	0.45	0.49	0.499
Diagonal	16	18	21	1210	339
Block Jacobi	11	13	15	28	360
Overlapping additive Schwarz	12	12	16	19	35
Condition ($3,420$ dof version)	$1.52 \cdot 10^6$	$1.64 \cdot 10^6$	$1.70 \cdot 10^6$	$1.75 \cdot 10^6$	$1.76 \cdot 10^6$

Table 3: Iterations for included sphere with common preconditioners in CG smoother

Conclusion: good condition is not sufficient for success

What's happening? ... the "incompressible" volumetric stiffness hides behind the steel

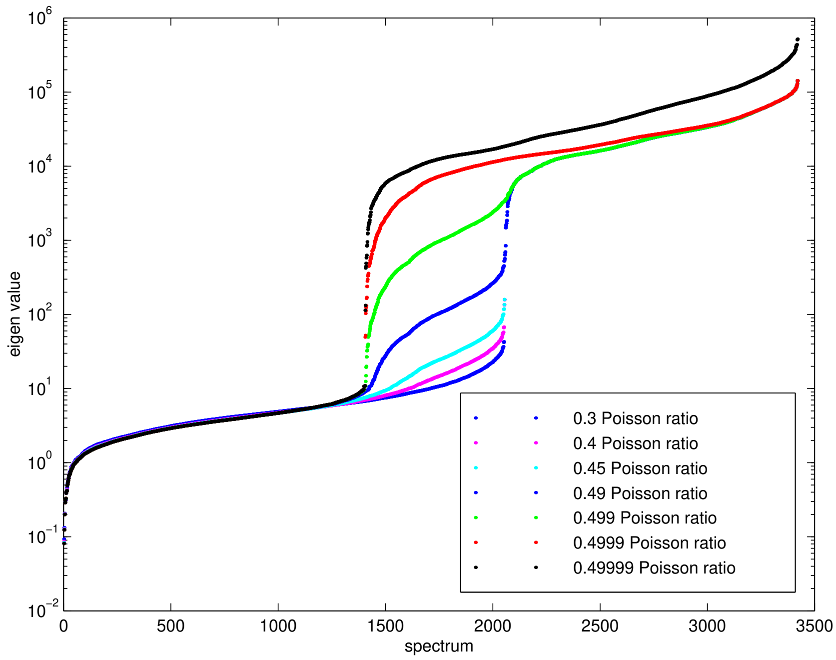


Figure 10: Spectrum of 3,810 dof included sphere with incompressible cover material

Proposition: classical multigrid takes the *scale* out

Corollary: multigrid is *not* solver ... the *smoother* is ...

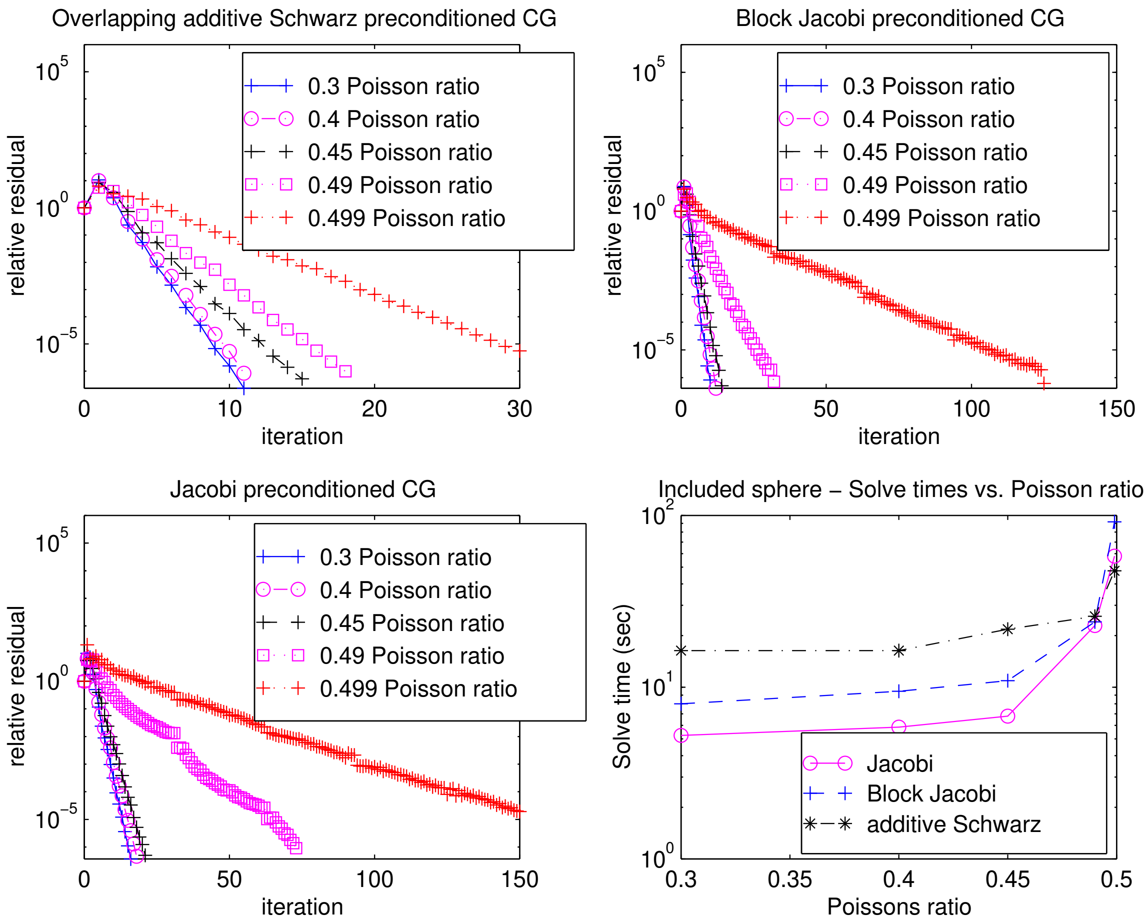
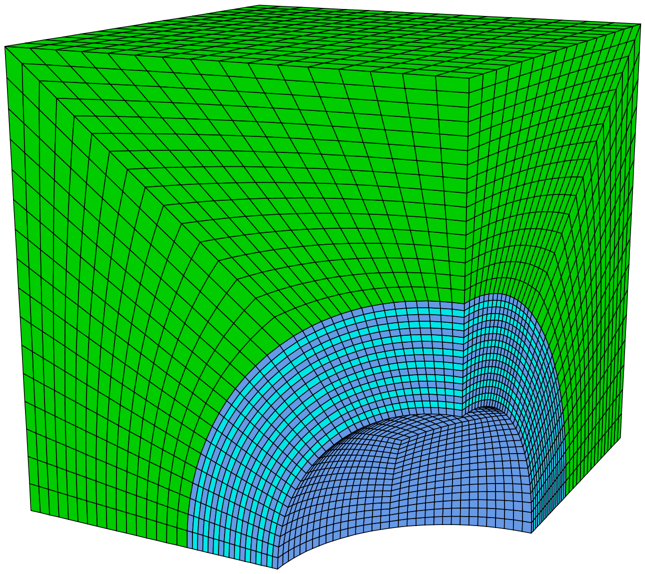


Figure 11: Residual convergence for included sphere with incompressible cover material

Who Needs Multigrid?

Ill-conditioning sources	MG	2 level methods	1 level methods
scale - large problems	X	1/2	
grading of discretization	X	X	
large jumps in coef.	1/2	= 1/2	
operator (e.g. incompressibility)		X	X

Performance results

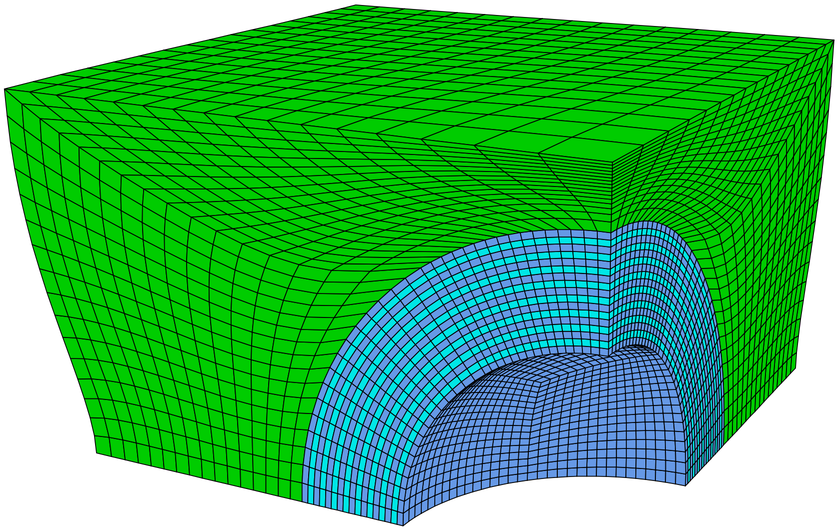


Problem

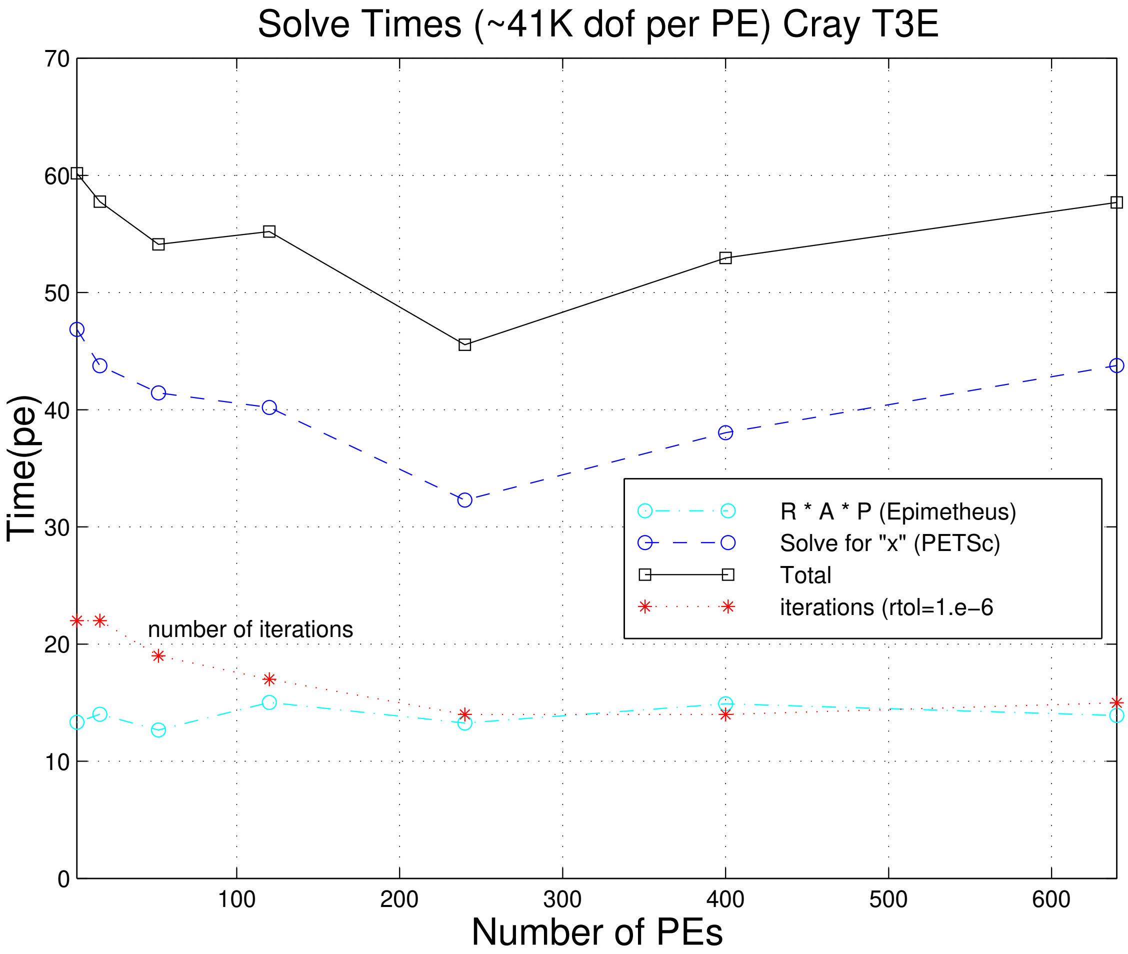
- Parameterized mesh - 79,679 to 26,257,055 dof problems used
- Hard sphere covered by soft ($E_s = 10^{-4} E_h$) material
- Poisson ratio 0.49 for soft material
- About 41,000 dof per processor Cray T3E, and 41,000 dof per IBM 4-way SMP node (blue-pacific)
- "Roller" boundary conditions symmetry plains and top surfaces
- Finite deformation (geometric NL) mixed soft material
- Plasticity (material NL) hard material w/o hardening
- $E_h = 1000$, $\sigma_y = 0.5 = 0.0005 \cdot E_h$

Solver

- Newton solver with energy relative tolerance $= 10^{-16}$, and
- Linear solver: CG with Epimetheus multigrid preconditioner.
- CG smoother, preconditioned with Block Jacobi preconditioner
- Epimetheus relative tolerance $= 10^{-4}$ first iteration ($k = 1$), and
- relative tolerance $= min\left(10^{-3}, 10^{-1} \cdot \frac{r_k}{r_{k-1}}\right)$ for $k \geq 2$

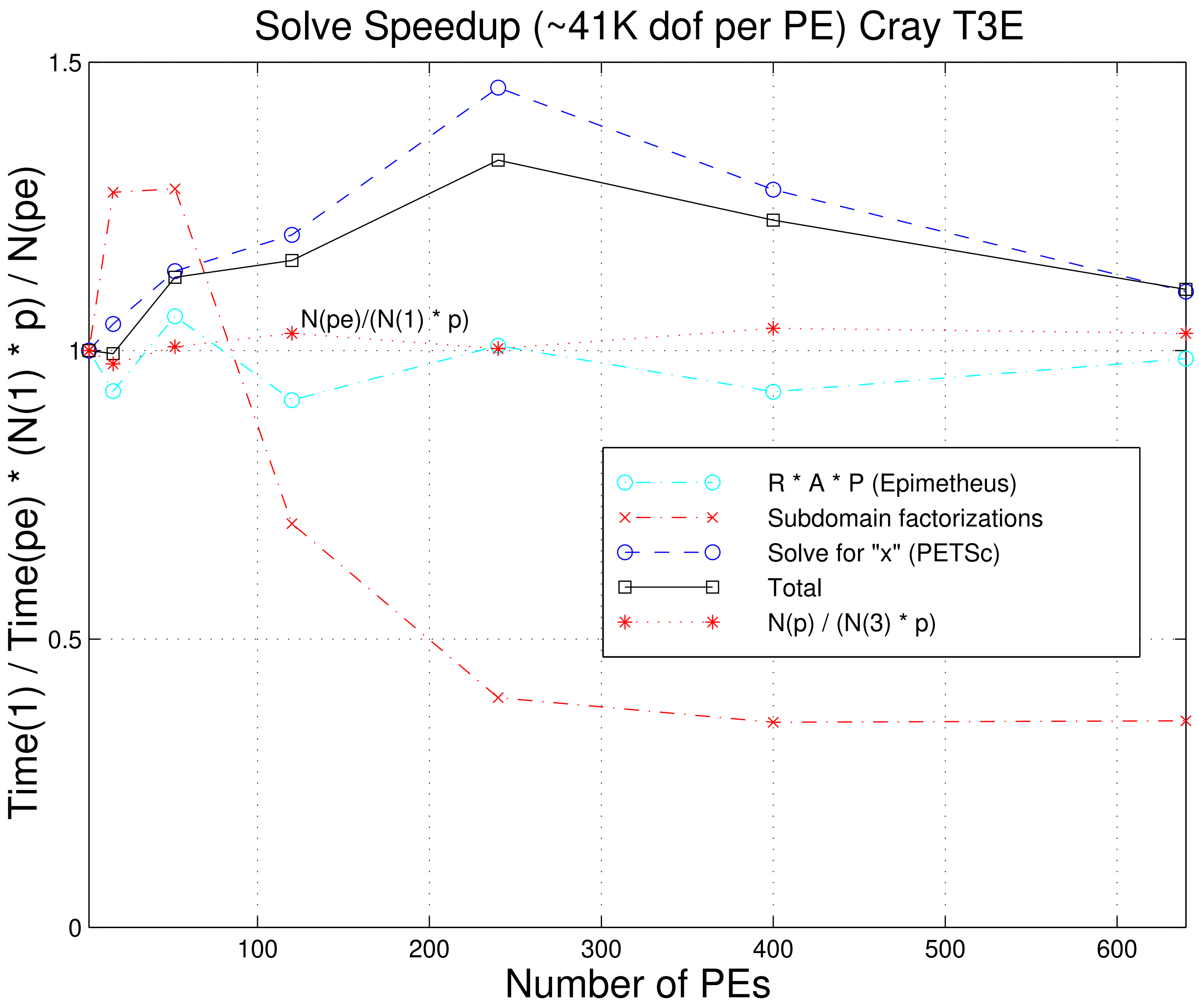


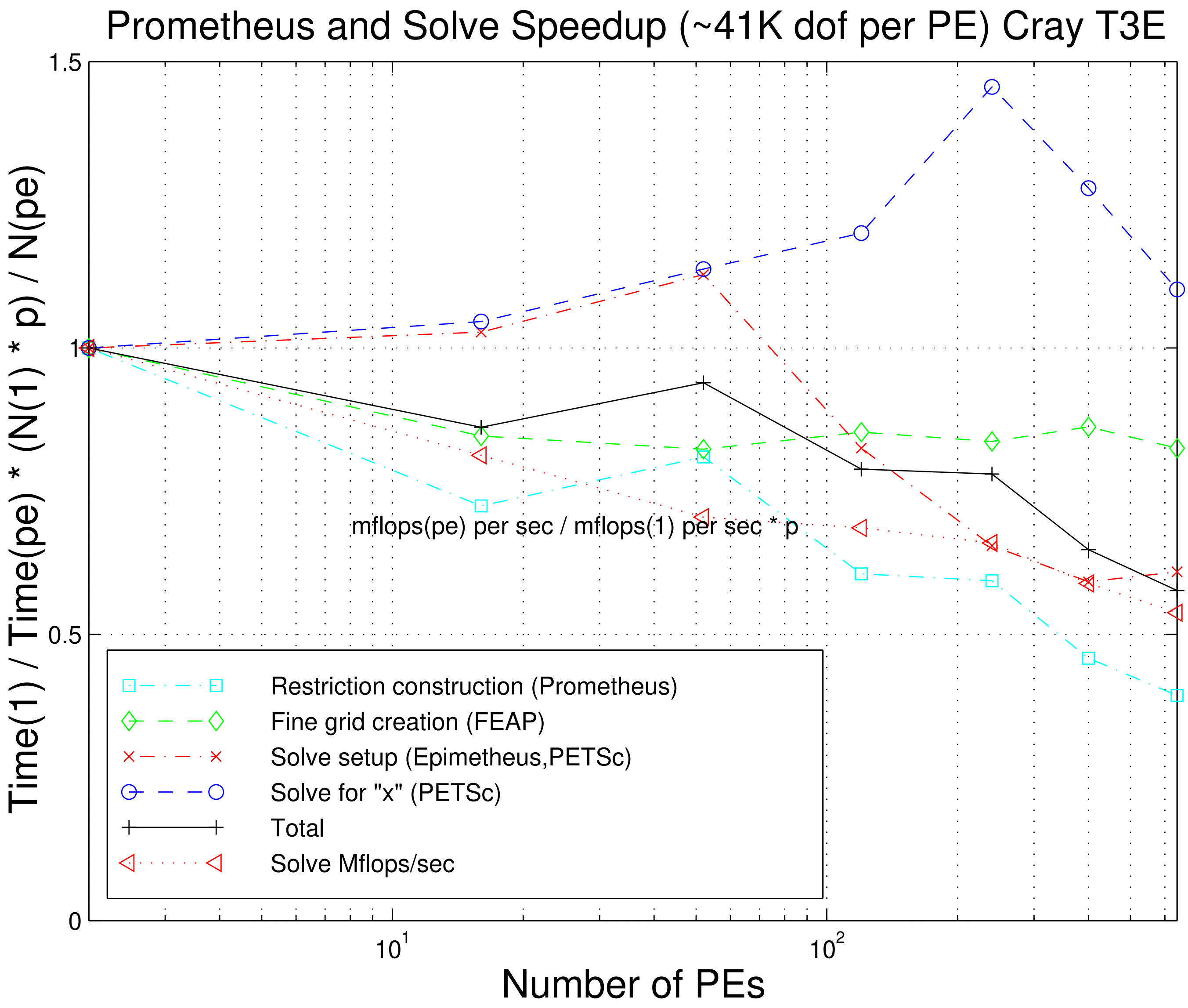
Cray T3E: Times for one solve (80 K to 26,257 K dof)

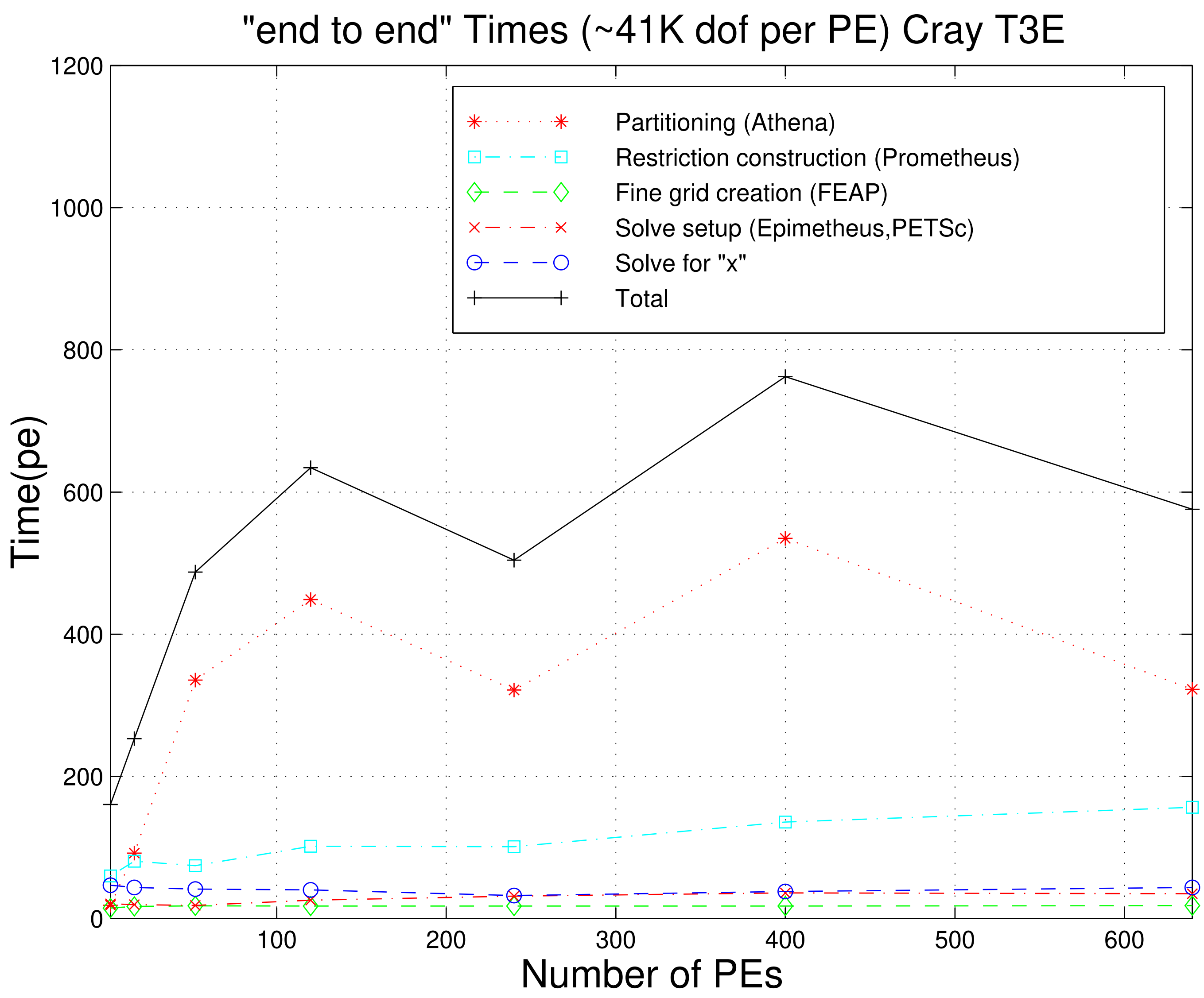


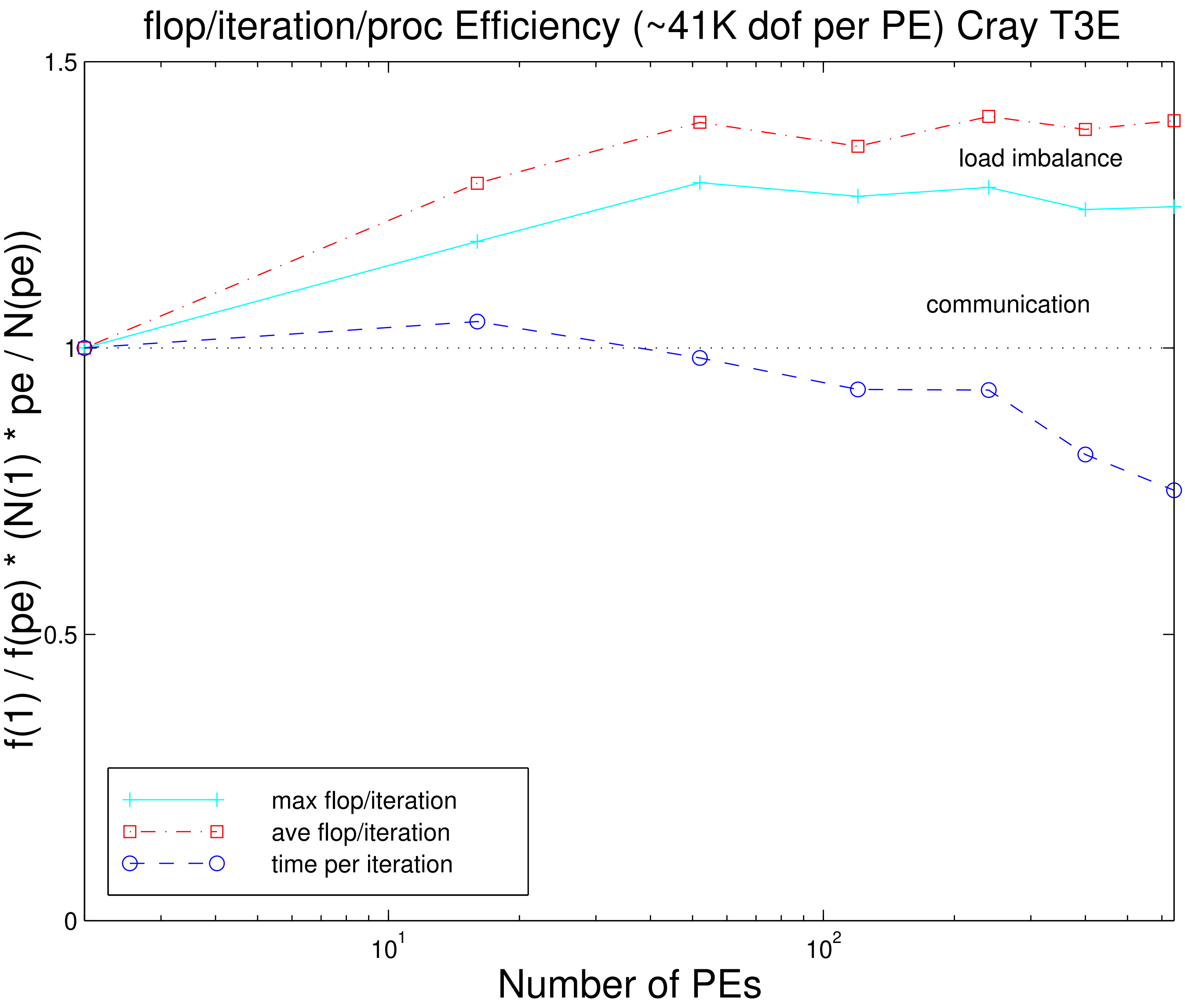
N. dof	79,679	622,815	2,085,599	4,924,223	9,594,879	16,553,759	26,257,055
N. Procs.	2	15	50	120	240	405	640
Iterations	22	22	19	17	14	14	15

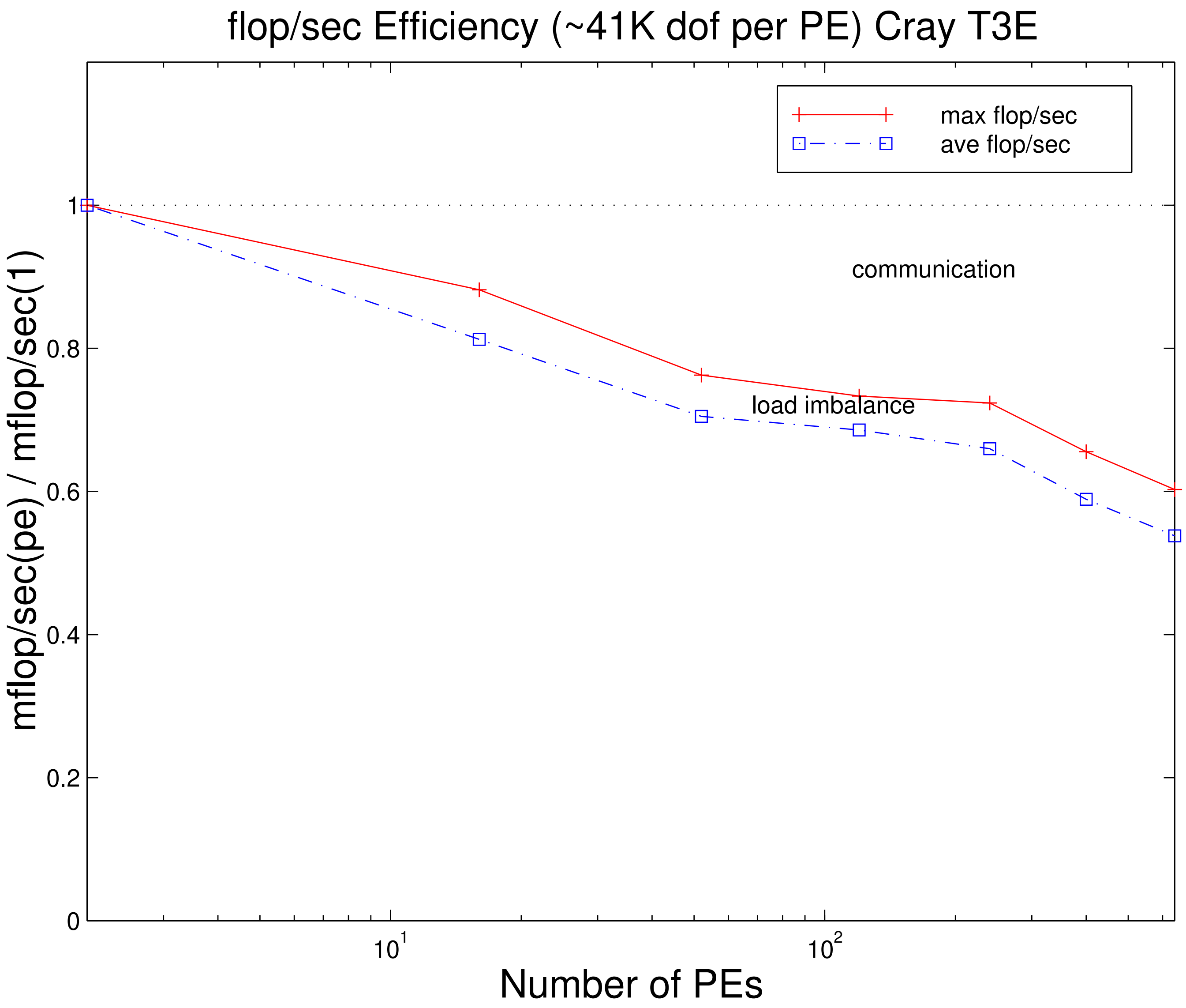
Table 4: Number of iterations and problem size


Solve Speedup (~41K dof per PE) Cray T3E
Time(1) / Time(pe) * (N(1) * p) / N(pe)
Number of PEs
1.5
1
0.5
0
100
200
300
400
500
600
N(pe)/(N(1) * p)
R * A * P (Epimetheus)
Subdomain factorizations
Solve for "x" (PETSc)
Total
N(p) / (N(3) * p)


Prometheus and Solve Speedup (~41K dof per PE) Cray T3E
Time(1) / Time(pe) * (N(1) * p) / N(pe)
Number of PEs
1.5
1
0.5
0
10^1
10^2
mflops(pe) per sec / mflops(1) per sec * p
Restriction construction (Prometheus)
Fine grid creation (FEAP)
Solve setup (Epimetheus,PETSc)
Solve for "x" (PETSc)
Total
Solve Mflops/sec

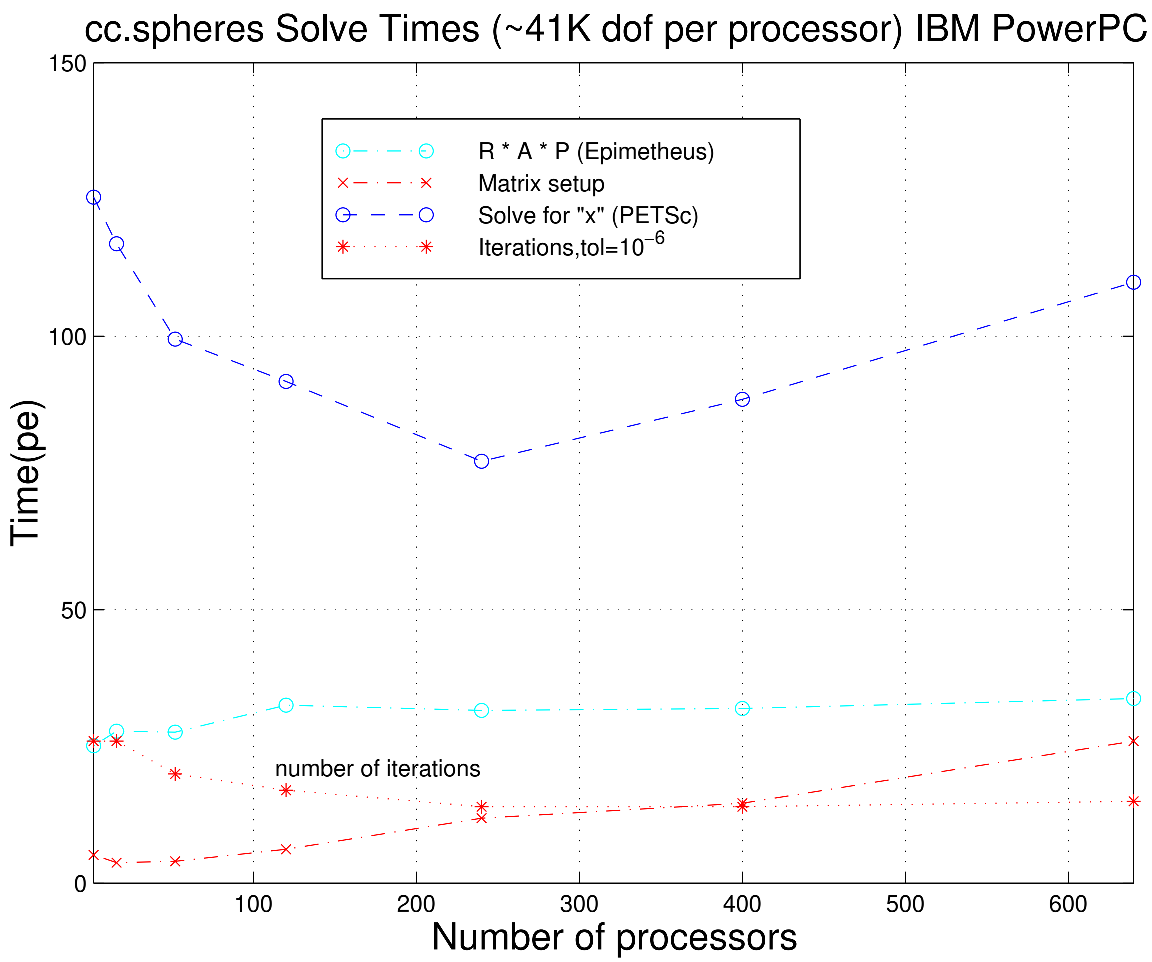

"end to end" Times (~41K dof per PE) Cray T3E
Partitioning (Athena)
Restriction construction (Prometheus)
Fine grid creation (FEAP)
Solve setup (Epimetheus,PETSc)
Solve for "x"
Total
Time(pe)
Number of PEs
0
200
400
600
800
1000
1200
100
200
300
400
500
600


flop/iteration/proc Efficiency (~41K dof per PE) Cray T3E
f(1) / f(pe) * (N(1) * pe / N(pe))
1.5
1
0.5
0
10^1
10^2
Number of PEs
load imbalance
communication
max flop/iteration
ave flop/iteration
time per iteration



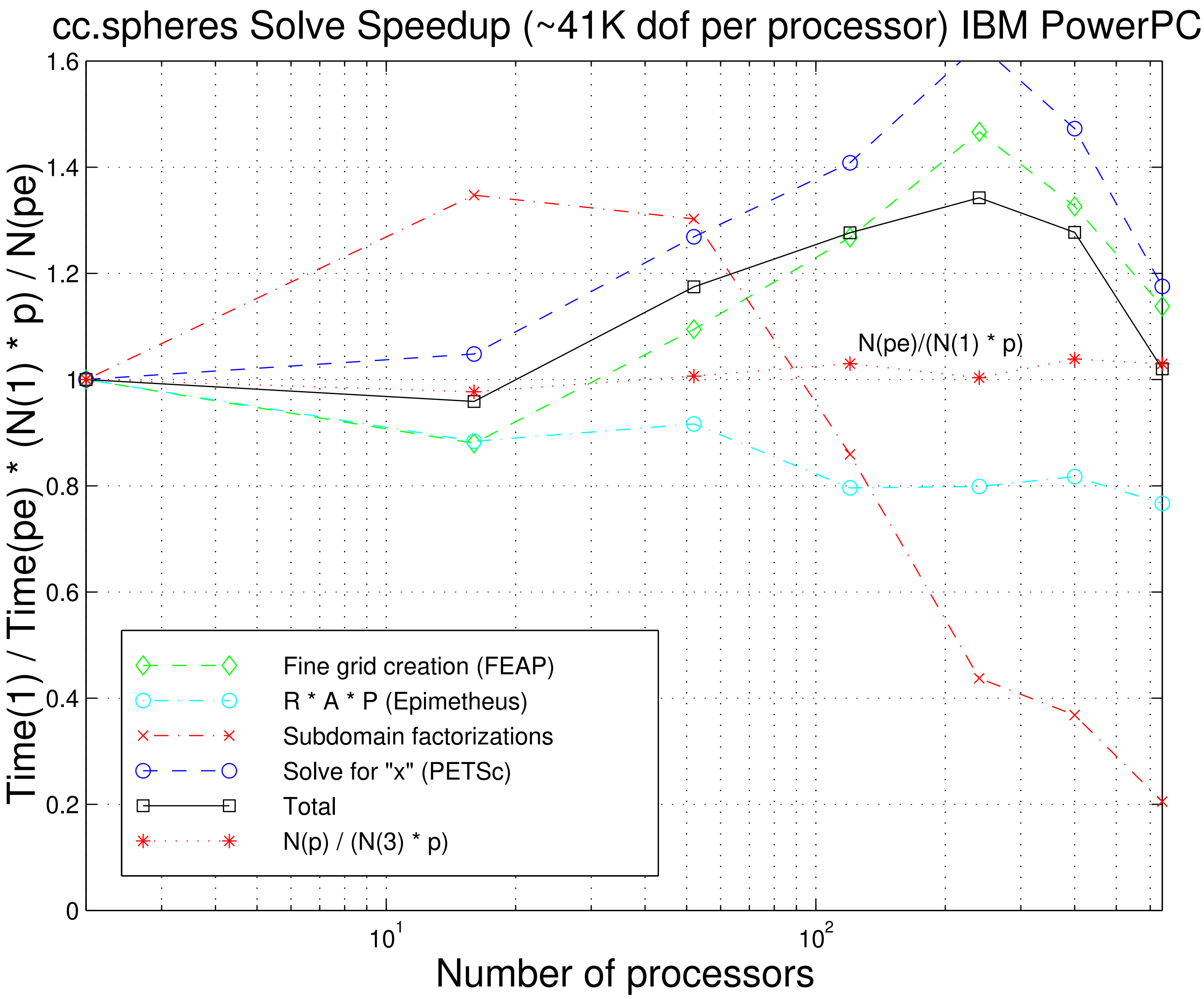
flop/sec Efficiency (~41K dof per PE) Cray T3E
max flop/sec
ave flop/sec
communication
load imbalance
mflop/sec(pe) / mflop/sec(1)
1
0.8
0.6
0.4
0.2
0
10^1
10^2
Number of PEs

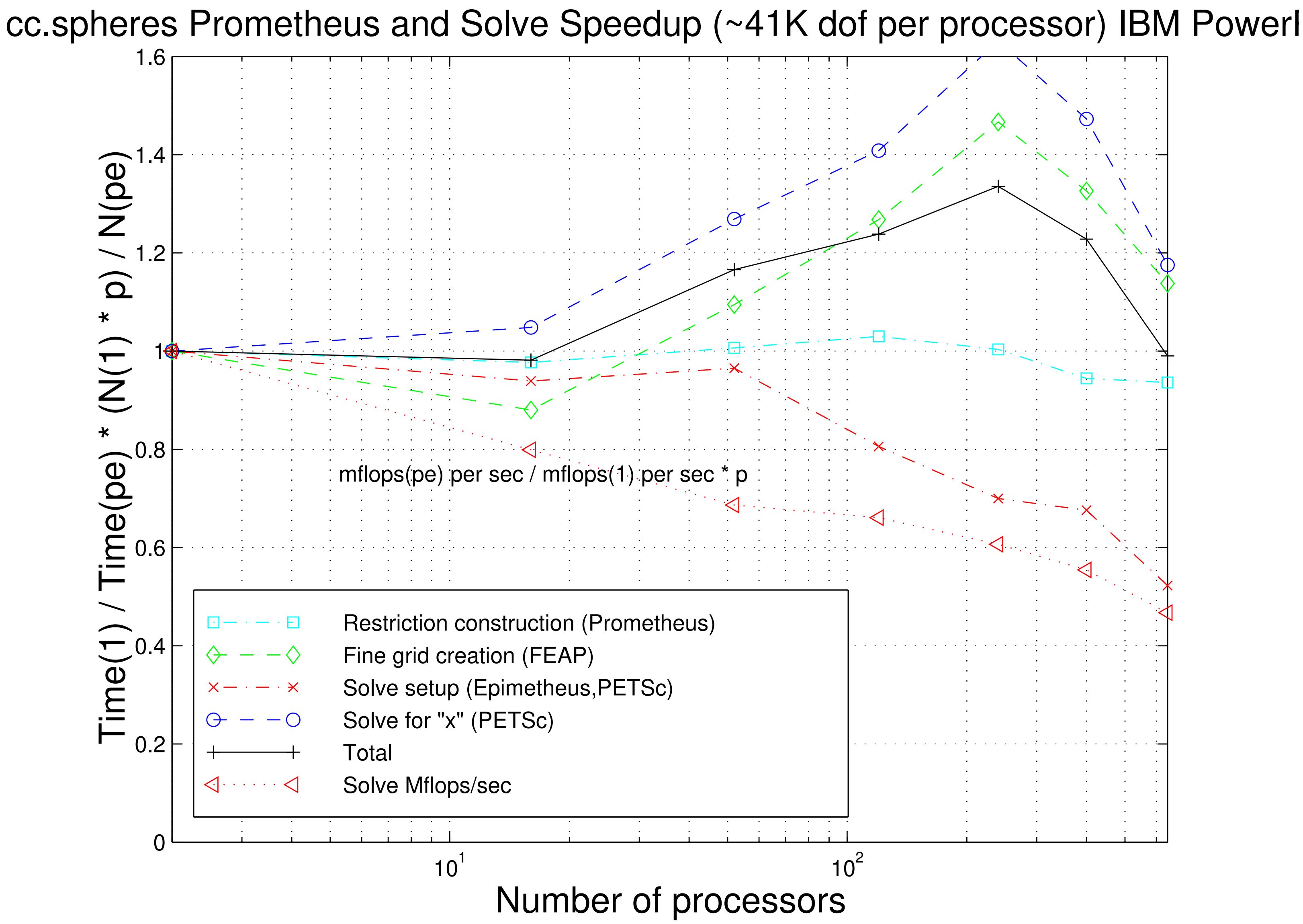
(Preliminary) Times on IMB PowerPC cluster



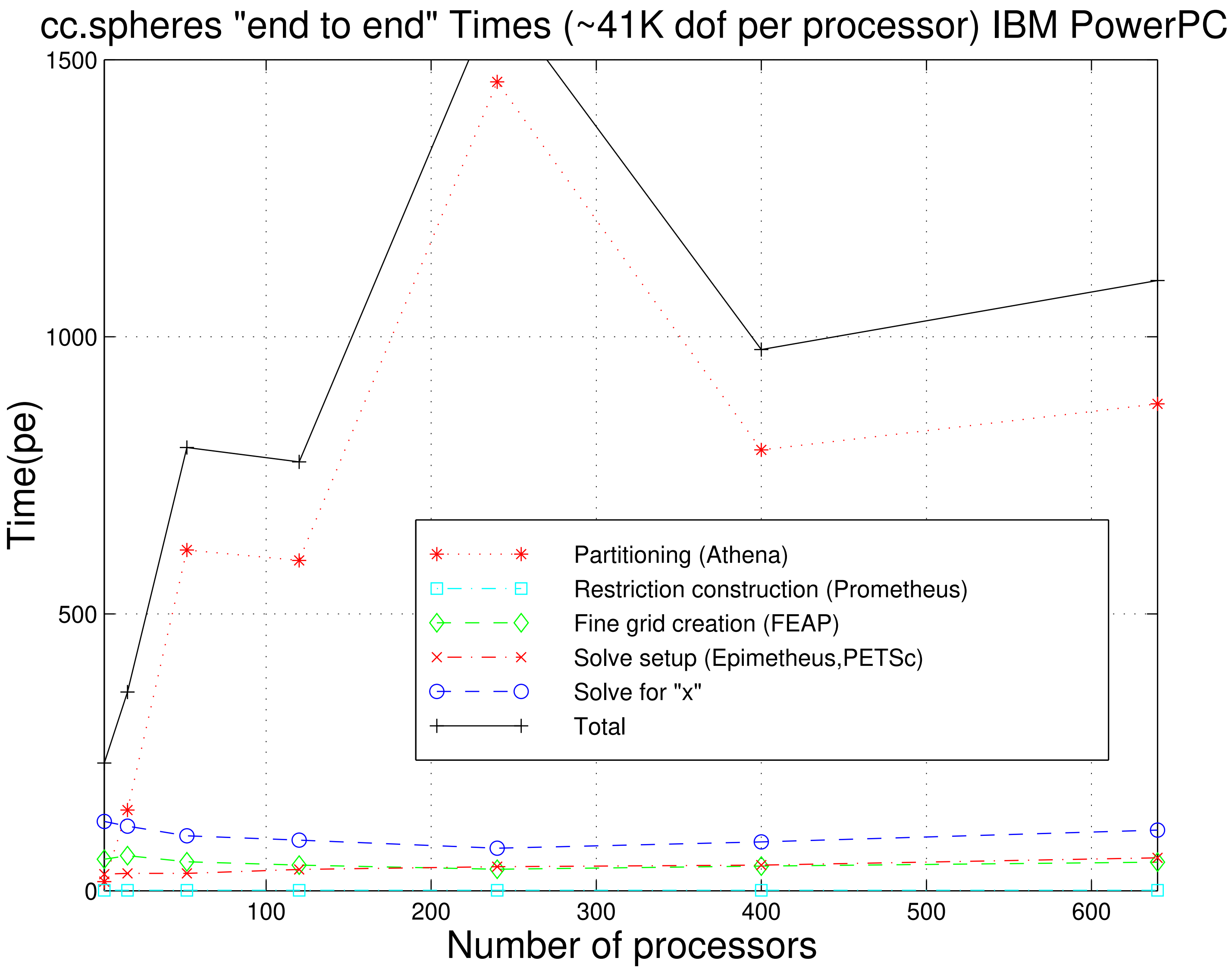
N. dof	79,679	622,815	2,085,599	4,924,223	9,594,879	16,553,759	26,257,055
N. Procs.	2	16	52	120	240	400	640
Iterations	26	26	20	17	14	14	15

Table 5: Number of iterations and problem size

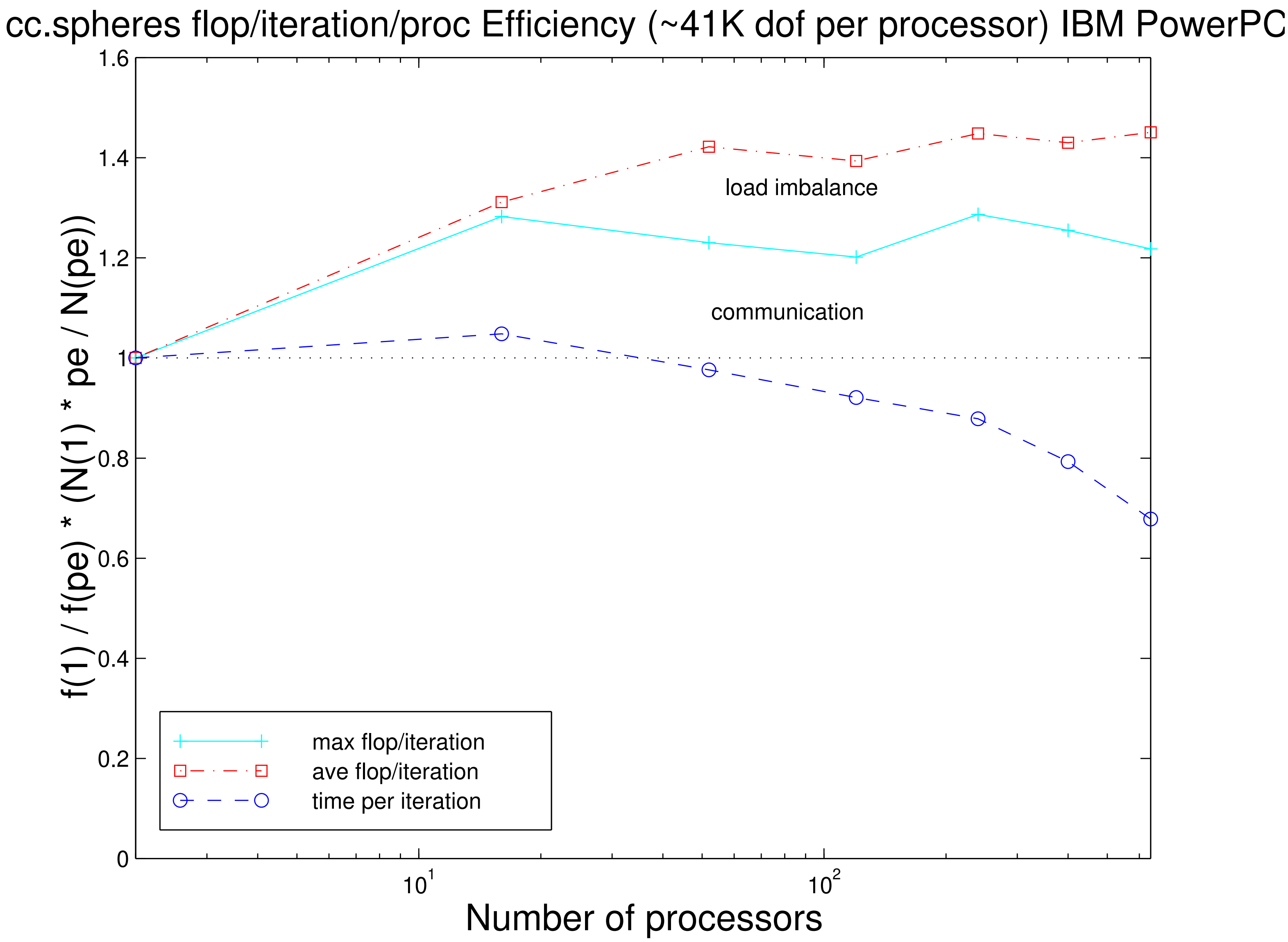

cc.spheres Solve Speedup (~41K dof per processor) IBM PowerPC
Time(1) / Time(pe) * (N(1) * p) / N(pe)
Number of processors
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
10^1
10^2
N(pe)/(N(1) * p)
Fine grid creation (FEAP)
R * A * P (Epimetheus)
Subdomain factorizations
Solve for "x" (PETSc)
Total
N(p) / (N(3) * p)



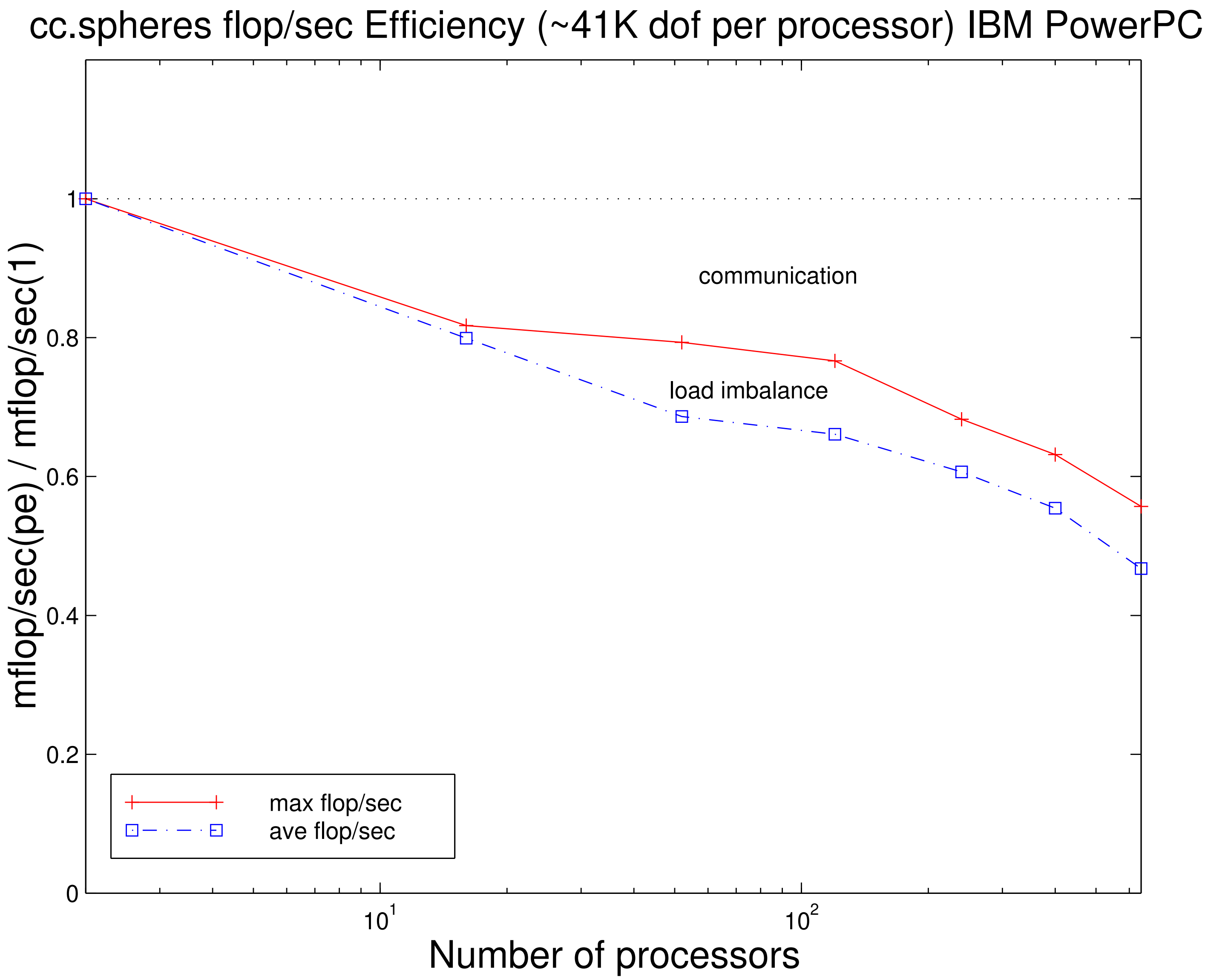
cc.spheres Prometheus and Solve Speedup (~41K dof per processor) IBM PowerI
Time(1) / Time(pe) * (N(1) * p) / N(pe)
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
mflops(pe) per sec / mflops(1) per sec * p
Restriction construction (Prometheus)
Fine grid creation (FEAP)
Solve setup (Epimetheus,PETSc)
Solve for "x" (PETSc)
Total
Solve Mflops/sec
10^1
10^2
Number of processors



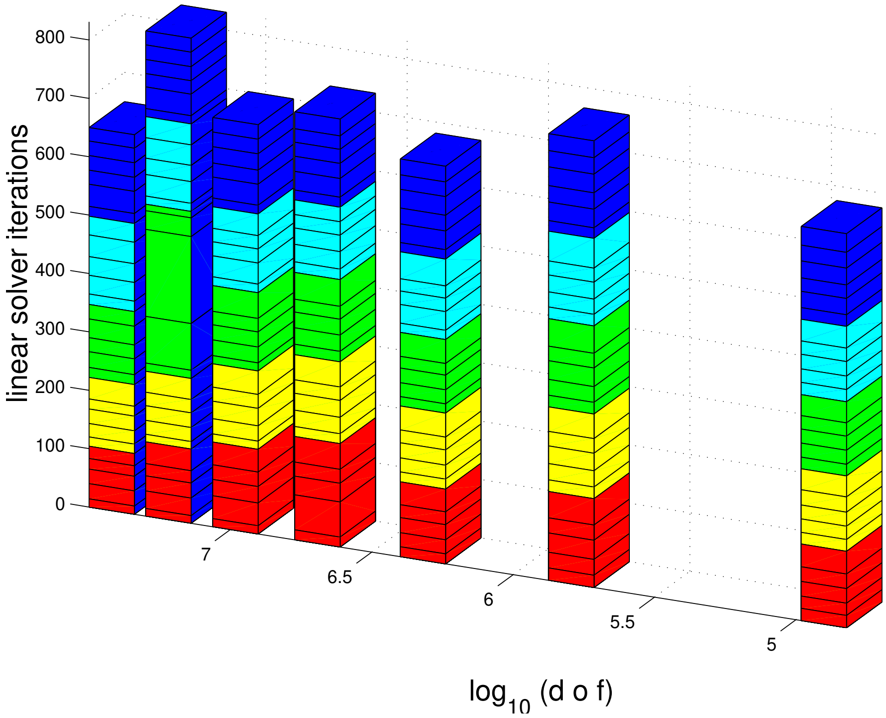
cc.spheres "end to end" Times (~41K dof per processor) IBM PowerPC
1500
1000
500
0
100
200
300
400
500
600
Time(pe)
Number of processors
Partitioning (Athena)
Restriction construction (Prometheus)
Fine grid creation (FEAP)
Solve setup (Epimetheus,PETSc)
Solve for "x"
Total

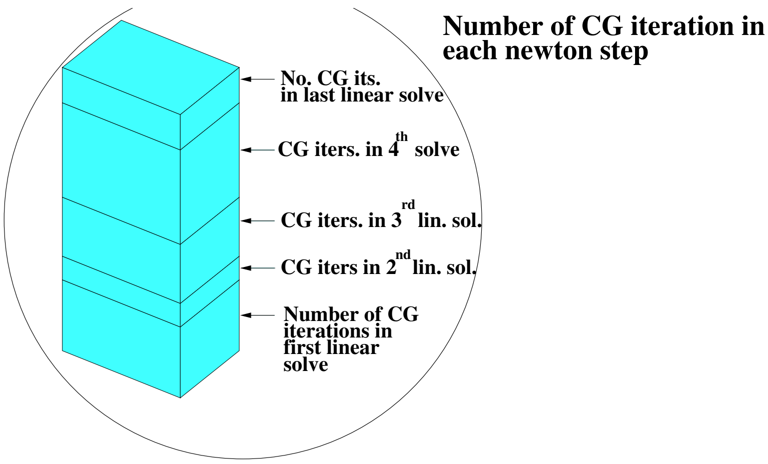


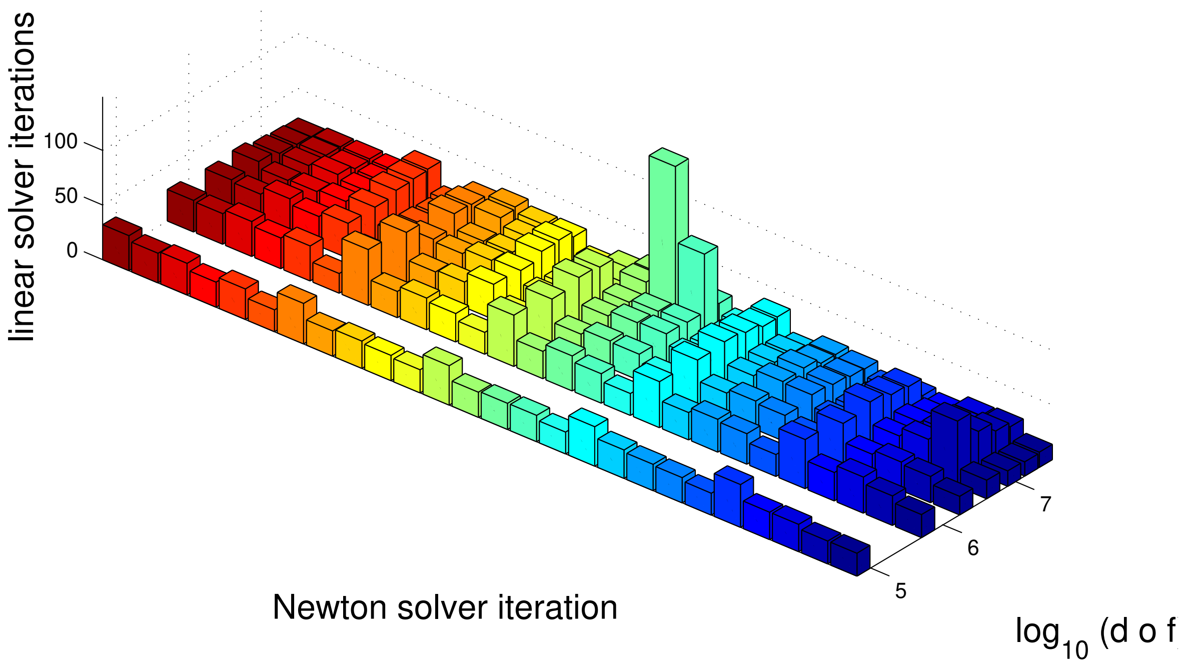
cc.spheres flop/iteration/proc Efficiency (~41K dof per processor) IBM PowerPC
f(1) / f(pe) * (N(1) * pe / N(pe))
Number of processors
load imbalance
communication
max flop/iteration
ave flop/iteration
time per iteration
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
10^1
10^2


cc.spheres flop/sec Efficiency (~41K dof per processor) IBM PowerPC
mflop/sec(pe) / mflop/sec(1)
Number of processors
communication
load imbalance
max flop/sec
ave flop/sec
0
0.2
0.4
0.6
0.8
1
10^1
10^2

Total number of multigrid iterations (5 Newton steps)






linear solver iterations
100
50
0
Newton solver iteration
5
6
7
log10 (d o f

Lagrange multiplier problems

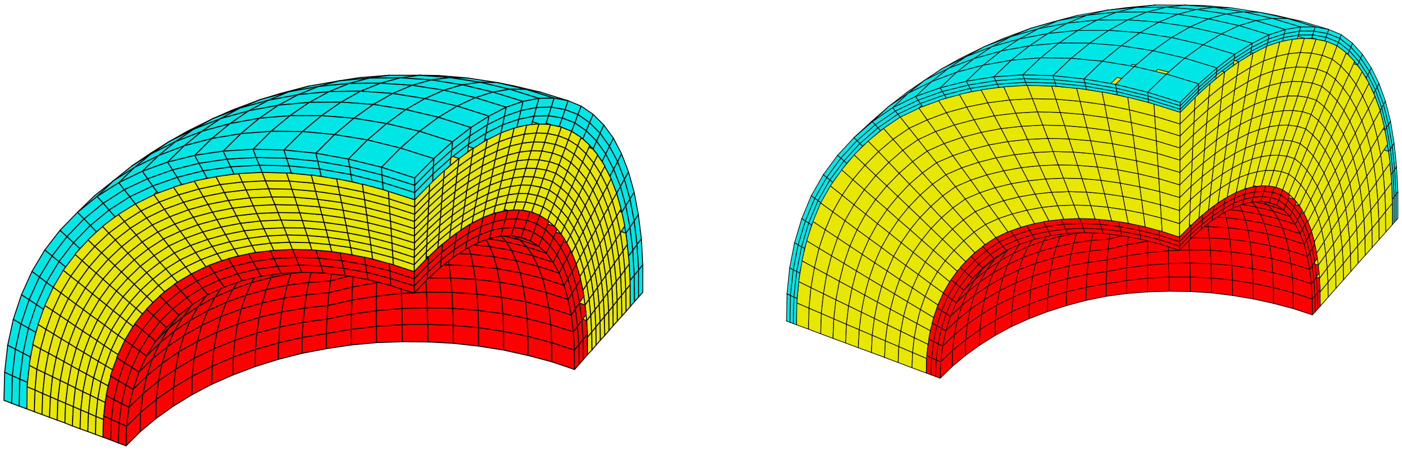


Figure 12: 22,092 dof concentric spheres with contact, undeformed and deformed shape

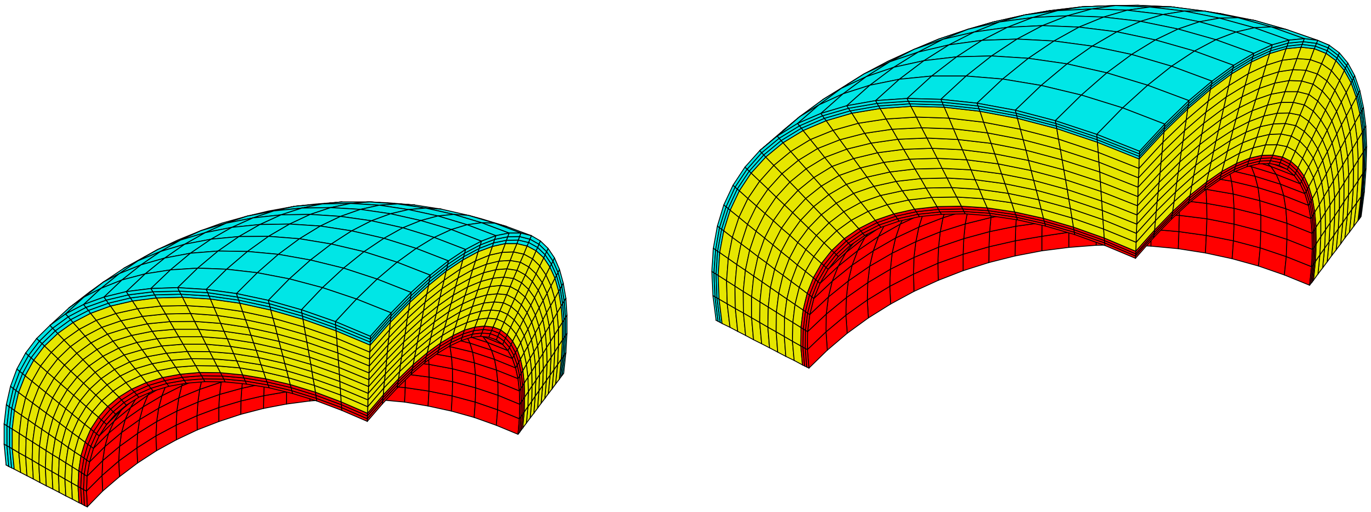


Figure 13: 15,810 dof concentric spheres without contact, undeformed and deformed shape

Constrained problems with Lagrange multipliers:

$$\begin{pmatrix} A & C^T \\ C & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}$$

Uzawa solver:

while $\left|Ax + C^T\lambda_{i-1} - f\right| > rtol \cdot |f|$ or $|Cx_i| > atol$
 solve for x_i: $\bar{A}x_i = f - C^T\lambda_{i-1}$
 $\lambda_i \leftarrow \lambda_{i-1} + \rho C x_i$
 $i \leftarrow i + 1$

Figure 14: Uzawa algorithm

- Outer iteration relative tolerance ($rtol = 10^{-6}$ or 10^{-12})
- Inner iteration relative tolerance = $|Cx|$

	inner iterations (total)		outer iterations	
rtol	10^{-12}	10^{-6}	10^{-12}	10^{-6}
no contact	27	12	1	1
contact	**313**	**85**	14	7

Conclusion

- Only fully scalable (algorithm and implementation) automatic unstructured FE solver, requiring only the FE mesh and fine grid stiffness matrix
- Highly scalable, fully non-linear, state-of-the-art FE application
- Promising results for nonlinear problems
  - Large def. elasticity and plasticity
  - Simple (serial) contact solutions
- Library: Prometheus-1.0
  (http://www.cs.berkeley.edu/~madams/Prometheus-1.0)
  - Prometheus alpha release -
    1^{st} customer: Chris Johnson, Utah
  - Good support for Cray T3E and IBMs.
  - Currently porting to SGI Origin2000
    - Berkeley NOW (Sun Ultasparc)
    - Berkeley Millenium (PC clusters)

Future work

- Harden algorithms and implementation by testing on more applications
- Extend algorithms and implementation for non-continuum elements
- Indefinite systems from large deformation elasticity and plasticity
- Indefinite systems from constraints w/ Lagrange multipliers

Future scalable solver library work …

1. Extend algorithm and features
    - More sophisticated face identification algorithms
    - Parallel Delaunay $\Rightarrow$ more robust, and complete algorithms
    - Extend implementation for shells, beams, trusses, etc.
    - Higher order elements and multi-physics problems

2. Current bottleneck: PETSc ...
   - Need RAR^T (sparse matrix triple product) in PETSc
   - Off processor matrix assembly routines need work
   - (restriction) Matrices need "left" and "right" communicators
   - Additive Schwarz has problem in setup phase
   - Some non-scalable constructs in matrix class
   - ... on PETSc's "todo" list
3. Rebuild and cleanup prototype's infrastructure for distributed multilevel unstructured grid classes ...
   - Redesign parallel distributed multi-level classes
   - Redesign data structures, clean code up

4. Add other promising "algebraic" multigrid algorithms ...
   - Agglomeration w/ rigid body modes: Bulgakov, Kuhn (1995)
   - Smoothed agglomeration: Vanek and Mandel (1995) ...
5. Build, maintain, document, and support a Library interface ...
   - Encapsulate Prometheus into PETSc "PC" object (low level)
   - Perhaps layer - high level - interface (e.g. "FEI")