Multi-level μ-Finite Element Analysis of Human Bone Structures

Peter Arbenz

ETH Zürich
Institute of Computational Science
E-mail: arbenz@inf.ethz.ch

Talk at PMAA06, Rennes, September 7-9, 2006.

Collaborators

- Harry van Lenthe, Institute for Biomedical Engineering, ETH Zürich
- Uche Mennel, Institute of Computational Science, ETH Zürich
- Ralph Müller, Institute for Biomedical Engineering, ETH Zürich
- Marzio Sala Institute of Computational Science, ETH Zürich

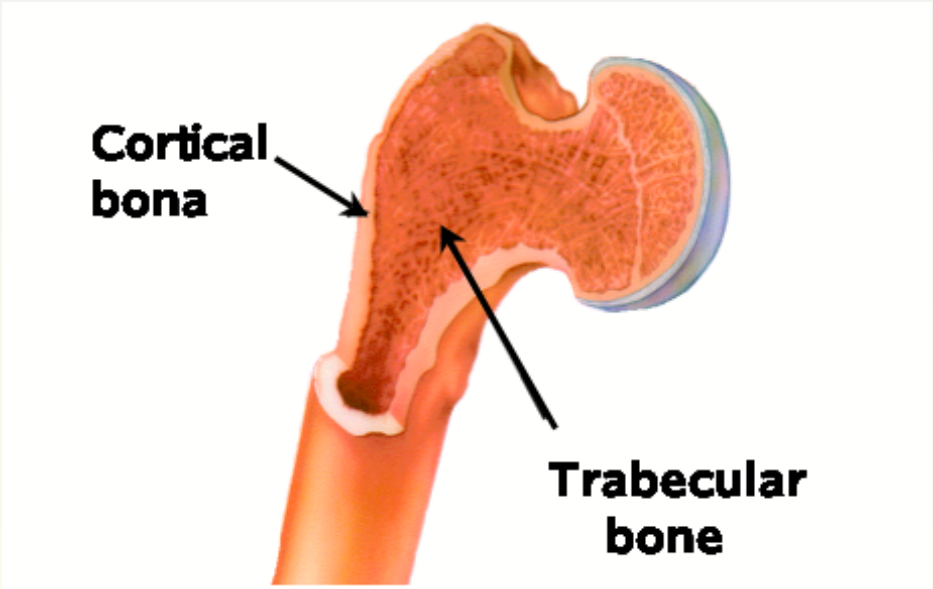
Outline of the talk

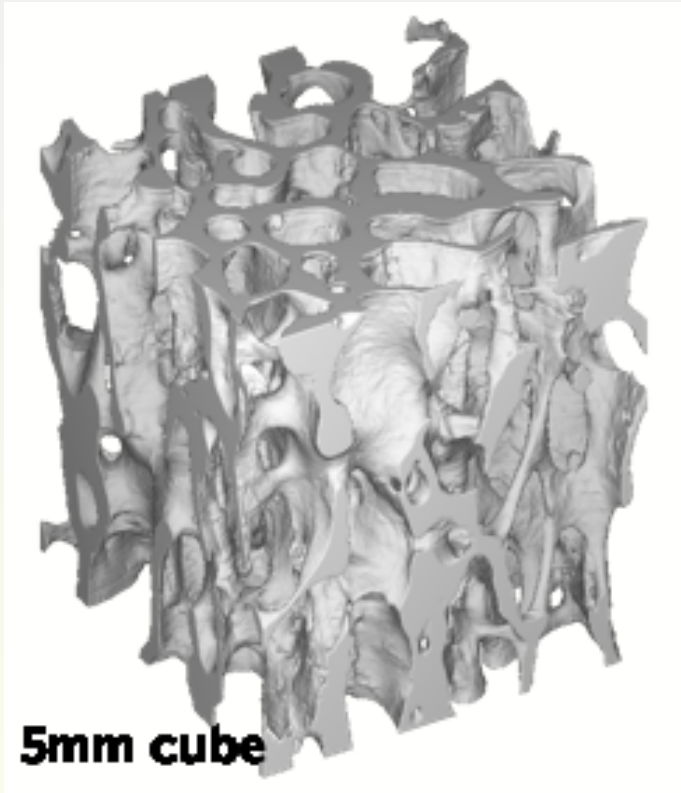
1. μFE bone modeling
2. The mathematical formulation
3. Solving the system of equations
4. The preconditioned conjugate gradient method
   - SA multilevel preconditioning
5. Numerical experiments
   - The Trilinos Software framework
   - Weak scalability test
   - Scalability test with real bone

The need for μFE analysis of bones

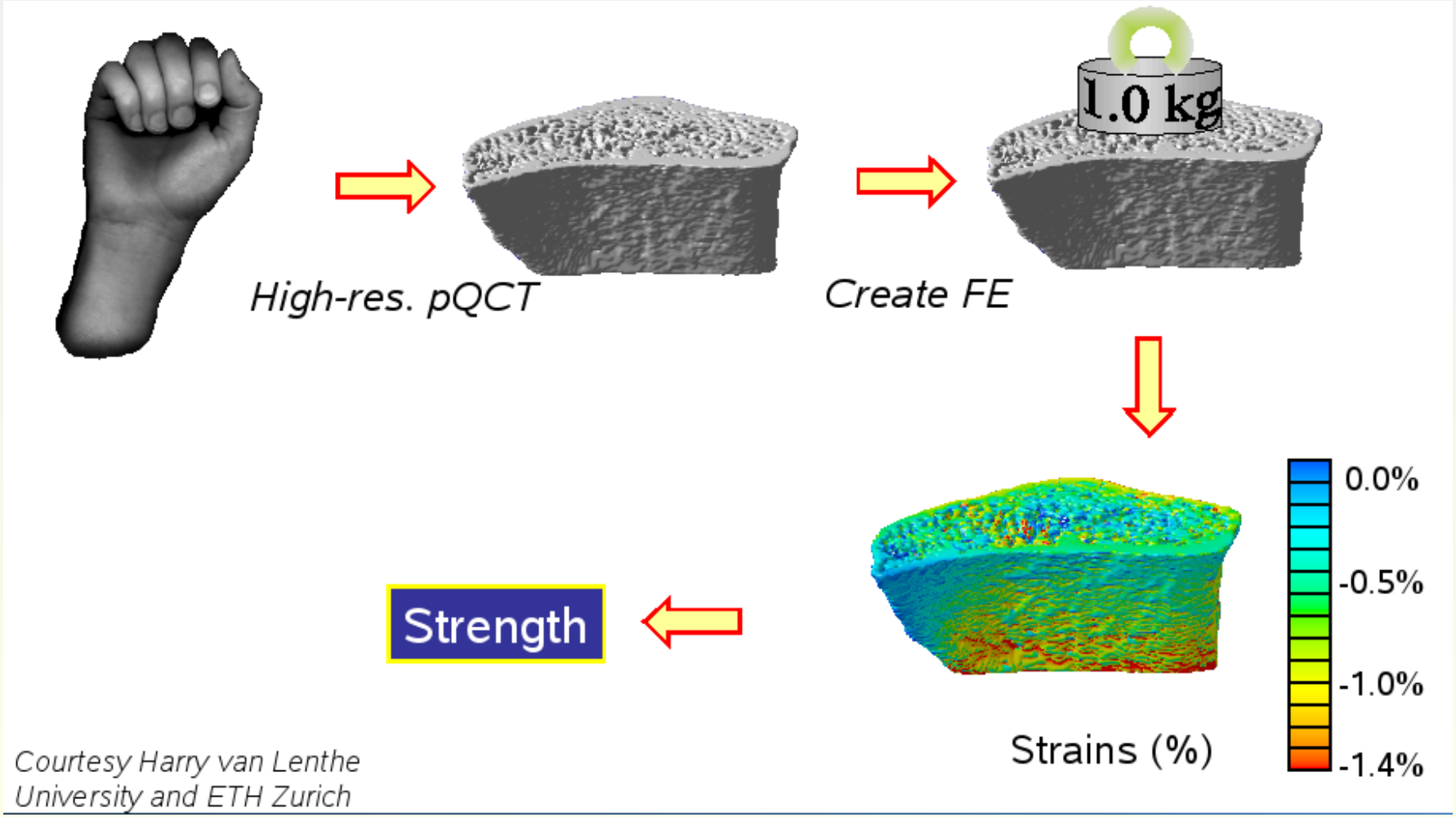
- *Osteoporosis* is disease characterized by low bone mass and deterioration of bone microarchitecture.
- Lifetime risk for osteoporotic fractures in women is estimated close to 40%; in men risk is 13%
- Enormous impact on individual, society and health care social systems (as health care problem second only to cardiovascular diseases)
- Since global parameters like bone density do not admit to predict the fracture risk, so patients have to be treated in a more individual way.
- Today's approach consists of combining 3D high-resolution CT scans of individual bones with a micro-finite element (μFE) analysis.

Cortical vs. trabecular bone



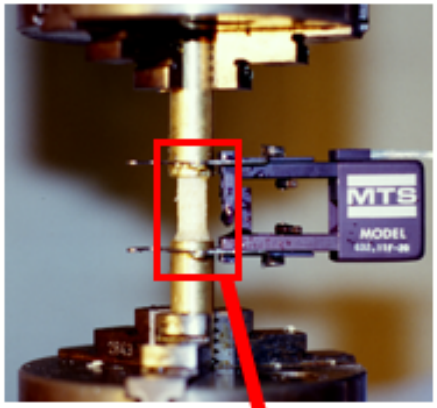


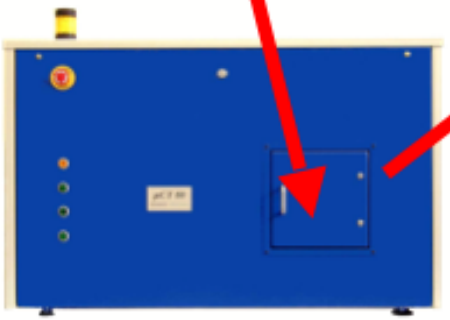
In vivo assessment of bone strength



Courtesy Harry van Lenthe
University and ETH Zurich

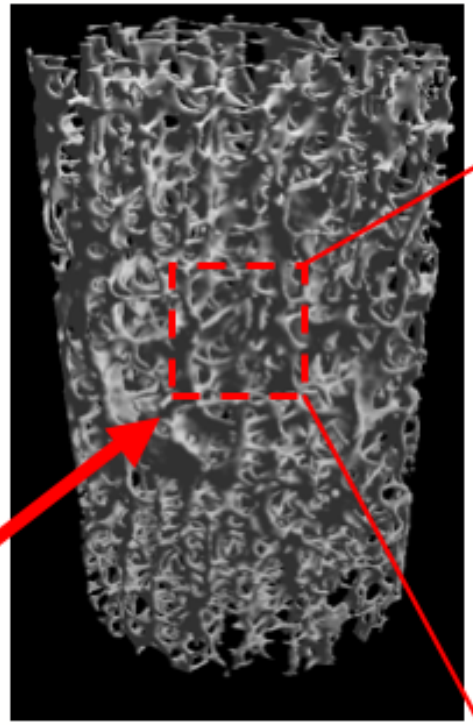
Mechanical Testing

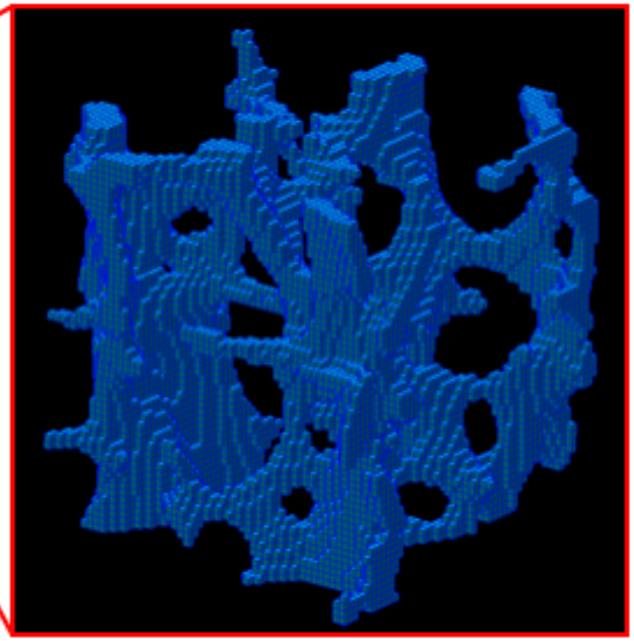




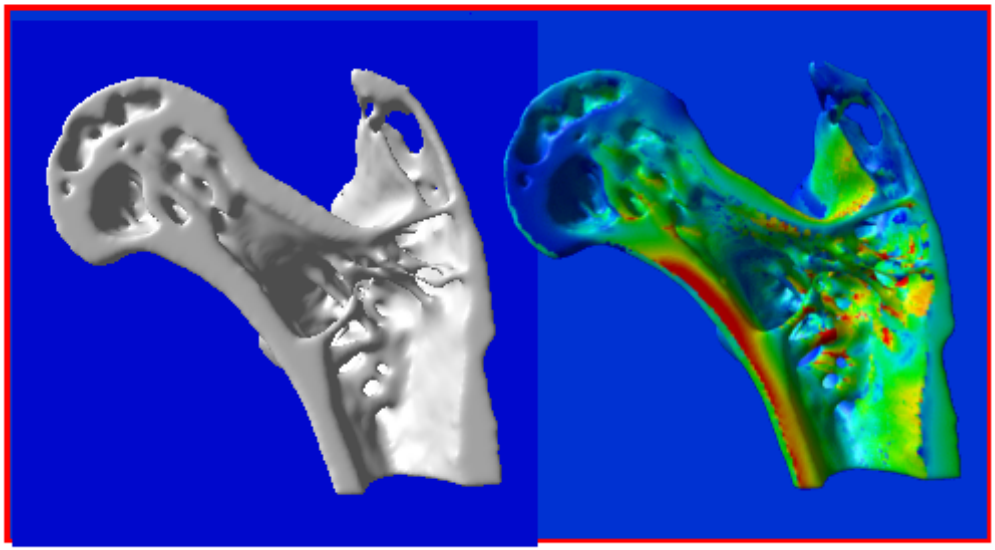
MicroCT @ 22 micrometer resolution

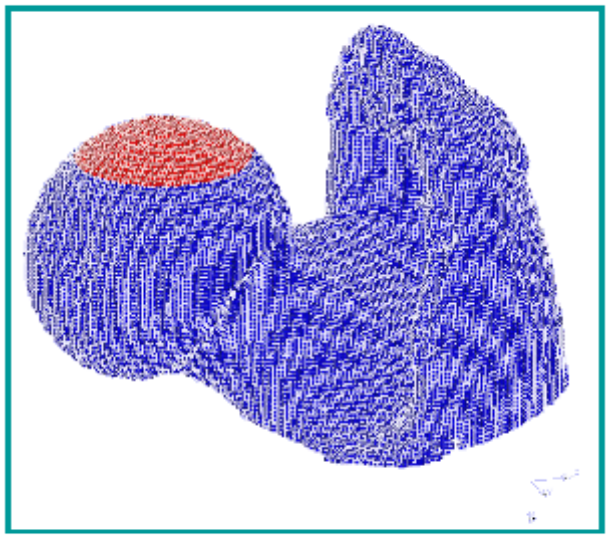
3D image





2.5mm^3
44 micrometer elements





Mathematical formulation

- Lamé equations of linear elasticity (weak formulation):
  Find $\mathbf{u} \in [H^1_E(\Omega)]^3 = \{v \in [H^1(\Omega)]^3 : \mathbf{v}_{|\Gamma_D} = \mathbf{u}_S\}$ s.t.

$$\int_\Omega [2\mu\varepsilon(\mathbf{u}) : \varepsilon(\mathbf{v}) + \lambda \operatorname{div}\mathbf{u} \operatorname{div}\mathbf{v}]\, d\Omega = \int_\Omega \mathbf{f}^t\mathbf{v} d\Omega + \int_{\Gamma_N} \mathbf{g}_S^t\mathbf{v} d\Gamma,$$

  for all $\mathbf{v} \in [H^1_0(\Omega)]^3 = \{v \in [H^1(\Omega)]^3 : \mathbf{v}_{|\Gamma_D} = 0\}$.
  with Lamé's constants λ, μ, volume forces $\mathbf{f}$, boundary tractions $\mathbf{g}$, symmetric strains

$$\varepsilon(\mathbf{u}) := \frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T).$$

- Domain Ω is extremely complicated: union of voxels.
- FE approximation: displacements $\mathbf{u}$ represented by piecewise trilinear polynomials.

Solving the system of equations

- System of equation

$$A\mathbf{x} = \mathbf{b}$$

  A is large (HUGE) sparse, symmetric positive definite.

- Approach by people of ETH Biomedical Engineering: preconditioned conjugate gradient (PCG) algorithm
  - element-by-element (EBE) matrix multiplication

$$A = \sum_{e=1}^{n_{\text{el}}} P_e A_e P_e^T, \tag{1}$$

  - diagonal preconditioning
  - very memory economic, slow convergence as problems get big

Solving the system of equations II

- New approach: smoothed aggregation AMG
- Requires assembling A
- Parallelization for distributed memory machines
- Employ software: Trilinos (Sandia NL),
  in particular use smoothed aggregation AMG preconditioner
  (Trilinos package ML), ParMETIS

The preconditioned conjugate gradient method

1: Given an initial vector $\mathbf{x}_0$ and a convergence tolerance ε this algorithm solves $A\mathbf{x} = \mathbf{b}$ approximately such that $\|A\mathbf{x} - \mathbf{b}\| < \varepsilon\|A\mathbf{x}_0 - \mathbf{b}\|$
2: Set $\mathbf{r}_0 := \mathbf{f} - A\mathbf{x}_0$;
3: Solve with the preconditioner: $M\mathbf{z}_0 = \mathbf{r}_0$;
4: Set $\rho_0 := \mathbf{z}_0^T\mathbf{r}_0$ and $\mathbf{p}_1 := \mathbf{z}_0$; $k := 0$;
5: **repeat**
6: $\quad k := k + 1$;
7: $\quad \mathbf{q}_k = A\mathbf{p}_k$;
8: $\quad \alpha_k := \rho_{k-1}/(\mathbf{p}_k{}^T\mathbf{q}_k)$;
9: $\quad \mathbf{x}_k := \mathbf{x}_{k-1} + \alpha_k\mathbf{p}_k$; $\mathbf{r}_k := \mathbf{r}_{k-1} - \alpha_k\mathbf{q}_k$;
10: $\quad$ Solve with the preconditioner: $M\mathbf{z}_k = \mathbf{r}_k$;
11: $\quad \rho_k := \mathbf{z}_k^T\mathbf{r}_k$; $\beta_k := \rho_k/\rho_{k-1}$;
12: $\quad \mathbf{p}_{k+1} := \mathbf{z}_k + \beta_k\mathbf{p}_k$;
13: **until** $\|\mathbf{r}_k\| < \varepsilon\|\mathbf{r}_0\|$

The preconditioned conjugate gradient method

1: Given an initial vector $\mathbf{x}_0$ and a convergence tolerance ε this algorithm solves $A\mathbf{x} = \mathbf{b}$ approximately such that $\|A\mathbf{x} - \mathbf{b}\| < \varepsilon\|A\mathbf{x}_0 - \mathbf{b}\|$
2: Set $\mathbf{r}_0 := \mathbf{f} - A\mathbf{x}_0$;
3: Solve with the preconditioner: $M\mathbf{z}_0 = \mathbf{r}_0$;
4: Set $\rho_0 := \mathbf{z}_0^T\mathbf{r}_0$ and $\mathbf{p}_1 := \mathbf{z}_0$; $k := 0$;
5: **repeat**
6: $\quad k := k + 1$;
7: $\quad \mathbf{q}_k = A\mathbf{p}_k$;
8: $\quad \alpha_k := \rho_{k-1}/(\mathbf{p}_k{}^T\mathbf{q}_k)$;
9: $\quad \mathbf{x}_k := \mathbf{x}_{k-1} + \alpha_k\mathbf{p}_k$; $\mathbf{r}_k := \mathbf{r}_{k-1} - \alpha_k\mathbf{q}_k$;
10: $\quad$ Solve with the preconditioner: $M\mathbf{z}_k = \mathbf{r}_k$;
11: $\quad \rho_k := \mathbf{z}_k^T\mathbf{r}_k$; $\beta_k := \rho_k/\rho_{k-1}$;
12: $\quad \mathbf{p}_{k+1} := \mathbf{z}_k + \beta_k\mathbf{p}_k$;
13: **until** $\|\mathbf{r}_k\| < \varepsilon\|\mathbf{r}_0\|$

Multilevel: a simple multigrid V-cycle

1: Approximately solve $A_\ell \mathbf{u} = \mathbf{b}$ where ℓ is the current grid level.
2: procedure multilevel$(A_\ell, \mathbf{b}_\ell, \mathbf{u}_\ell, \ell)$
3: **if** $\ell < L$ **then**
4: $\quad \mathbf{u}_\ell = S_\ell(A_\ell, \mathbf{b}_\ell, \mathbf{u}_\ell)$; {Presmoothing}
5: $\quad \mathbf{r}_\ell = R_\ell(\mathbf{b}_\ell - A_\ell \mathbf{u}_\ell)$; {Coarse grid correction}
$\quad \mathbf{v}_{\ell+1} = \mathbf{0}$;
$\quad$ multilevel$(A_{\ell+1}, \mathbf{r}_{\ell+1}, \mathbf{v}_{\ell+1}, \ell+1)$;
$\quad \mathbf{u}_\ell = \mathbf{u}_\ell + P_\ell \mathbf{v}_{\ell+1}$;
6: $\quad \mathbf{u}_\ell = S_\ell(A_\ell, \mathbf{b}_\ell, \mathbf{u}_\ell)$; {Postsmoothing}
7: **else**
8: $\quad$ Solve $A_\ell \mathbf{u}_\ell = \mathbf{b}_\ell$;
9: **end if**

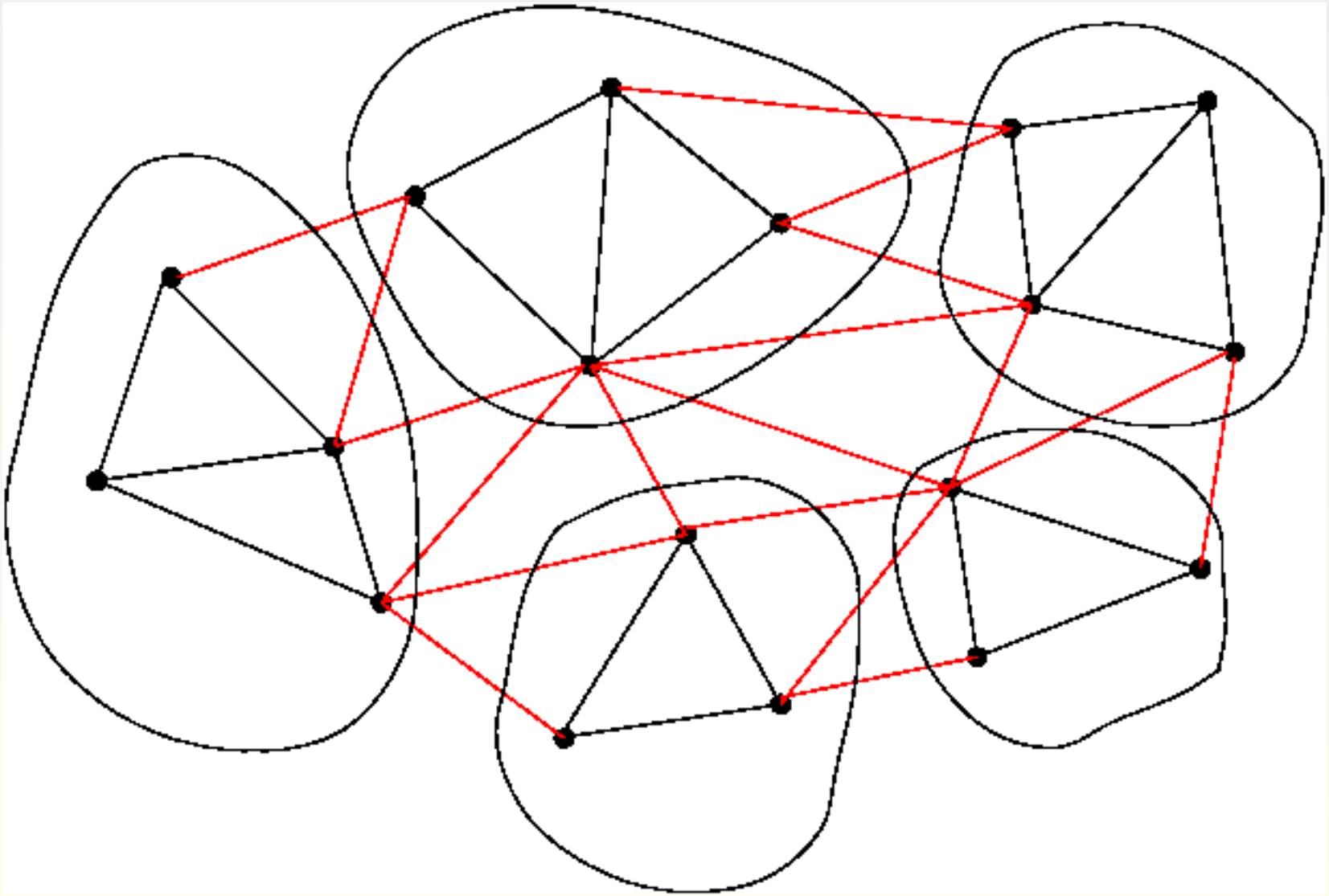
Preconditioner: Call procedure multilevel$(A_0 = A, \mathbf{b}, \mathbf{u} = \mathbf{0}, L)$

Smoothed aggregation

Key aspect in algebraic multigrid methods:
Definition of the auxiliary operators P_ℓ, R_ℓ, and A_ℓ.
Two variants:

- algebraically coarsen on each level by identifying a set of coarser-level nodes (C-nodes) and finer-level nodes (F-nodes) (Ruge, Stüben, 1987).
- Algebraically coarsen on each level by grouping the nodes into contiguous subsets, called aggregates, as done in smoothed aggregation (SA) (Vanek, Brezina, Mandel, 2001).

Aggregation



Outline of construction of SA preconditioner

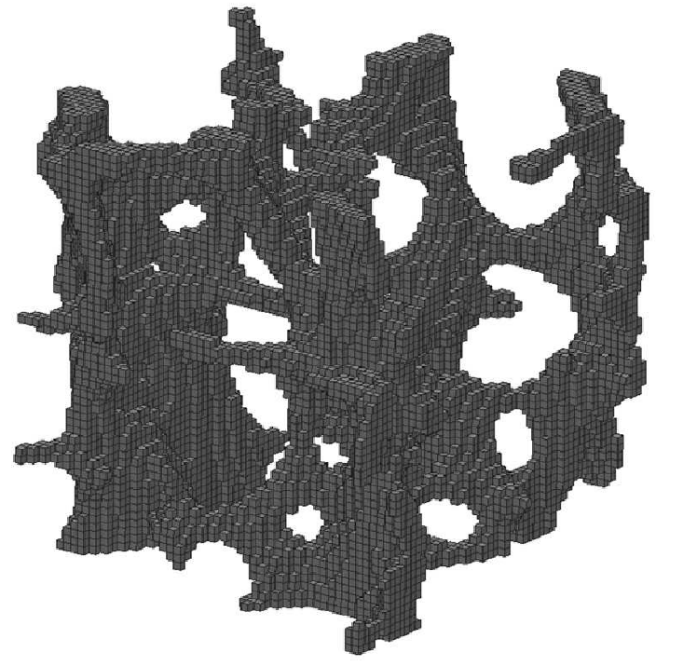
1. Build adjacency graph of A. (Take blocks into account.)
2. Group graph vertices into contiguous subsets, called *aggregates* ((Par)METIS). Each aggregate represents a coarser grid vertex.
3. Define a grid transfer operator:
   Use (e.g.) low-energy modes (in our case, the rigid body modes as obtained when no boundary conditions are applied), that are 'chopped' and inserted within the $(i,j)^{th}$ block if the i^{th} fine grid point has been assigned to the j^{th} aggregate.
   This gives the tentative prolongator $P_{0,\ell}$.
4. 'Smooth' tentative prolongator

$$P_\ell = (I_\ell - \omega_\ell D_\ell^{-1} A_\ell) P_{0,\ell} \qquad \omega_\ell = \frac{4/3}{\lambda_{\max}(D_\ell^{-1} A_\ell)},$$

Smoother S_ℓ

- Gauss-Seidel
  Looses its quality as processor number increases if restricted to processors' local portions of A.
- Polynomial smoother.
  Choose a Chebyshev polynomial that is small on the upper part of the spectrum of A_ℓ (Adams, Brezina, Hu, Tuminaro, 2003).
  Parallelizes perfectly, quality independent of processor number.

Parallel mesh reading



Mesh file content

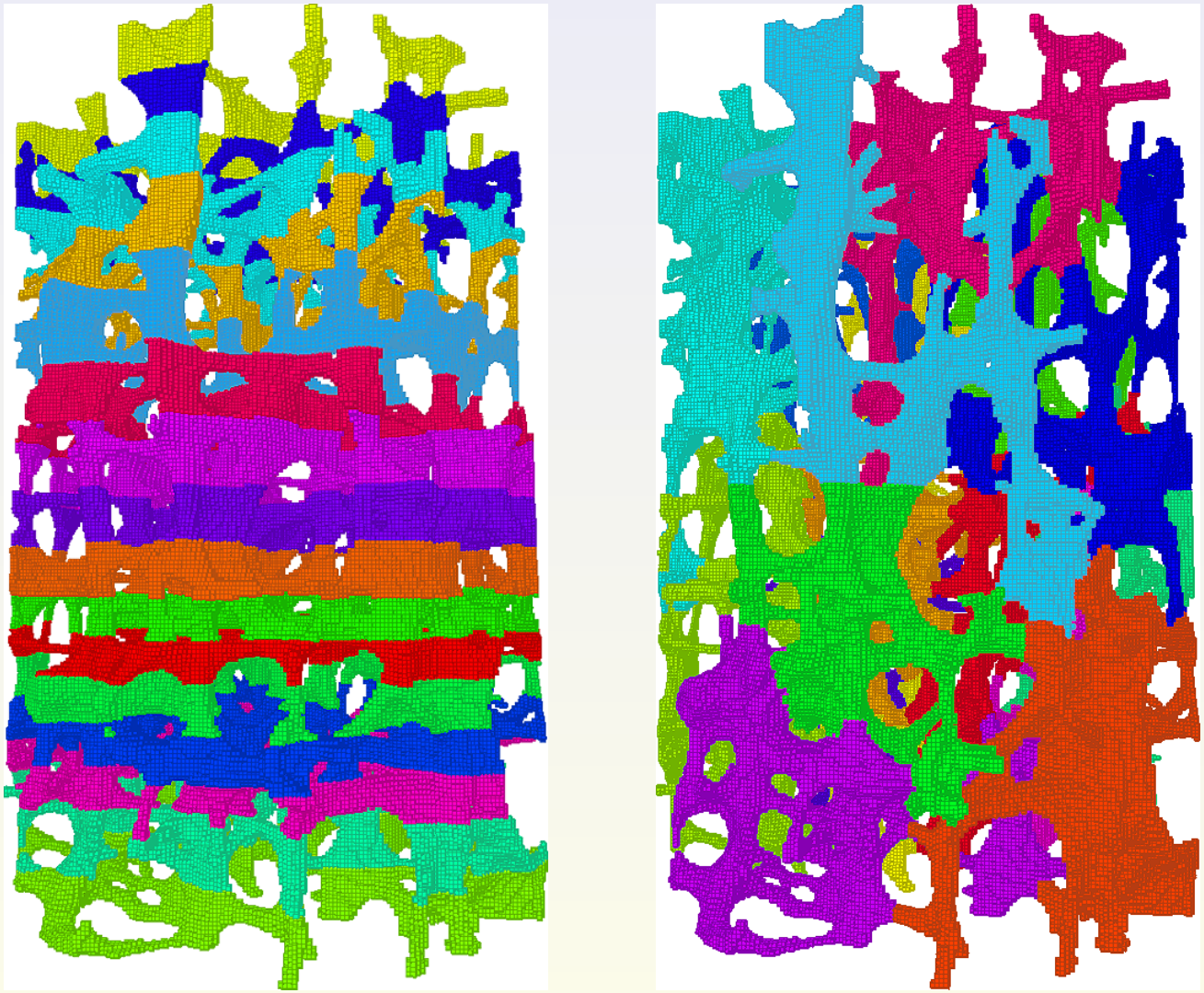
- A list of node coordinates (x, y, z)
- A list of hexahedra (8 nodes)
- A list of boundary conditions

Implementation: HDF5 format/library

- Binary file format allows for efficient I/O
- Allows for parallel I/O
- Mesh reading scales with number of processors

Mesh partitioning

- Purpose
  - Load balance: Each processor gets the same number of nodes
  - Minimize solver communication: Minimize the surface area of the interprocessor interfaces
  - Crucial for efficient parallel execution
- Implementation
  - ParMETIS: Parallel library for graph partitioning.
  - Heuristic multilevel algorithm



Initial partition (left) based on node coordinates
ParMETIS repartition (right)

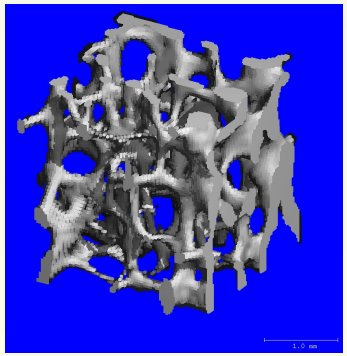
The Trilinos Software framework

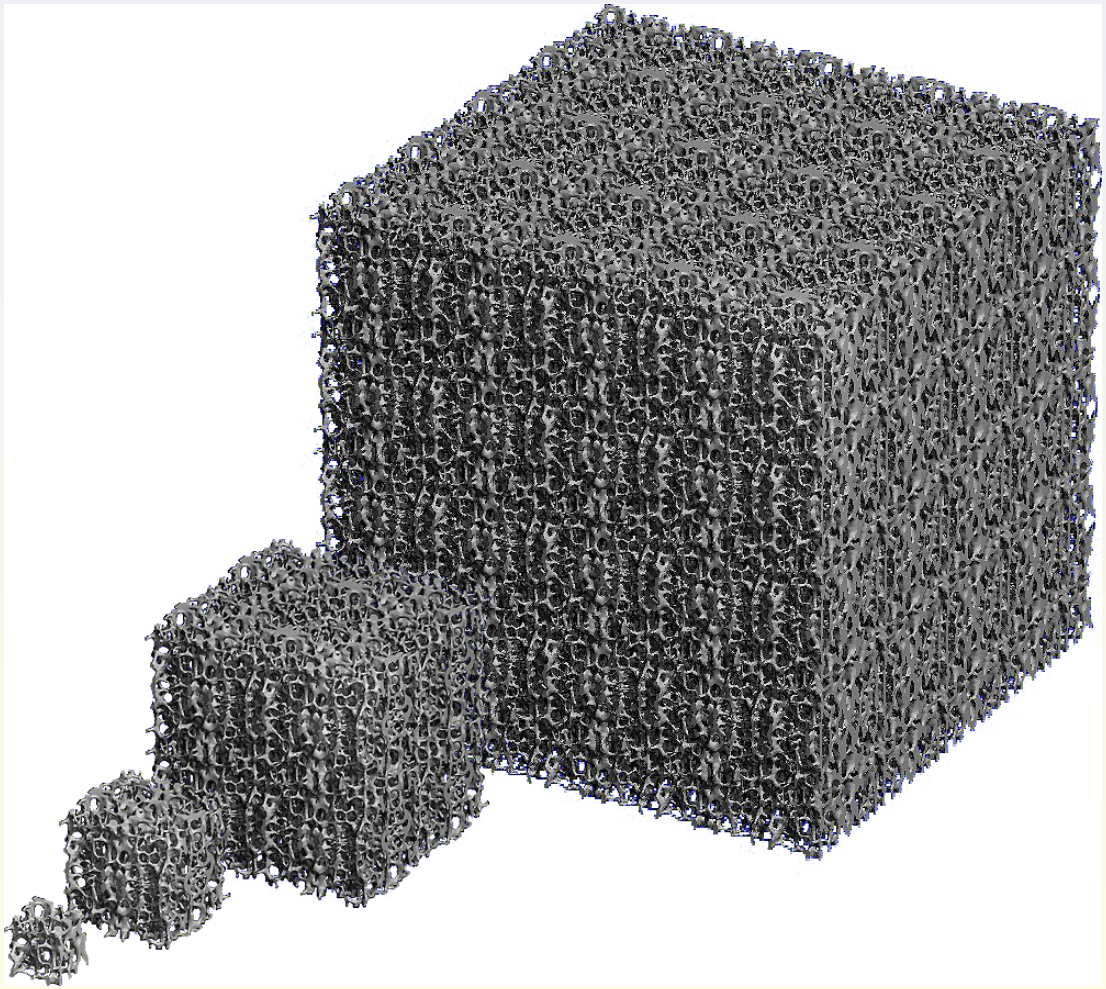
- The Trilinos Project is an effort to develop parallel solver algorithms and libraries within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific applications.
- See `http://software.sandia.gov/trilinos/`
- Provides means to distribute (multi)vectors and (sparse) matrices (Epetra package).
- Provides solvers that work on these distributed data.
- Iterative solvers and preconditioners (AztecOO/IFPACK).
- Smoothed aggregation multilevel preconditioner (ML package).
- Data distribution for parallelization (ParMETIS).
- Direct solver on coarsest level (AMESOS)

Computational environment

- Gonzales: Linux cluster
  - 160 nodes
  - node = two 64-bit AMD Opteron 2.4 GHz processors, 8 GB RAM
  - Compute nodes inter-connected via two-layer Quadrics QsNet II network. Bandwidth 900 MB/s. Latency 2$< \mu$sec
- Cray XT3 (at Swiss Supercomputer Center CSCS)
  - 1100 2.6 GHz AMD Opteron processors, 2 GB RAM
  - Cray SeaStar high speed network, bandwidth 7.6 GB/s (4 GB/s sustained)
  - Peak performance is 5.9 Tflop/s.
- Software
  - UNICOS/lc, MPI-2, Trilinos 4.0

Weak scalability test





Problem size scales with the number of processors

Weak scalability test [cont'd]

name	nodes	elements	equations
cube 1	98'381	60'482	295'143
cube 2	774'717	483'856	2'324'151
cube 3	2'609'611	1'633'014	7'828'833
cube 4	6'164'270	3'870'848	18'492'810
cube 5	12'038'629	7'560'250	36'115'887
cube 6	20'766'855	13'064'112	62'300'565
cube 7	32'983'631	20'745'326	98'950'893
cube 8	49'180'668	30'966'784	147'542'004
cube 9	70'042'813	44'091'378	210'128'439

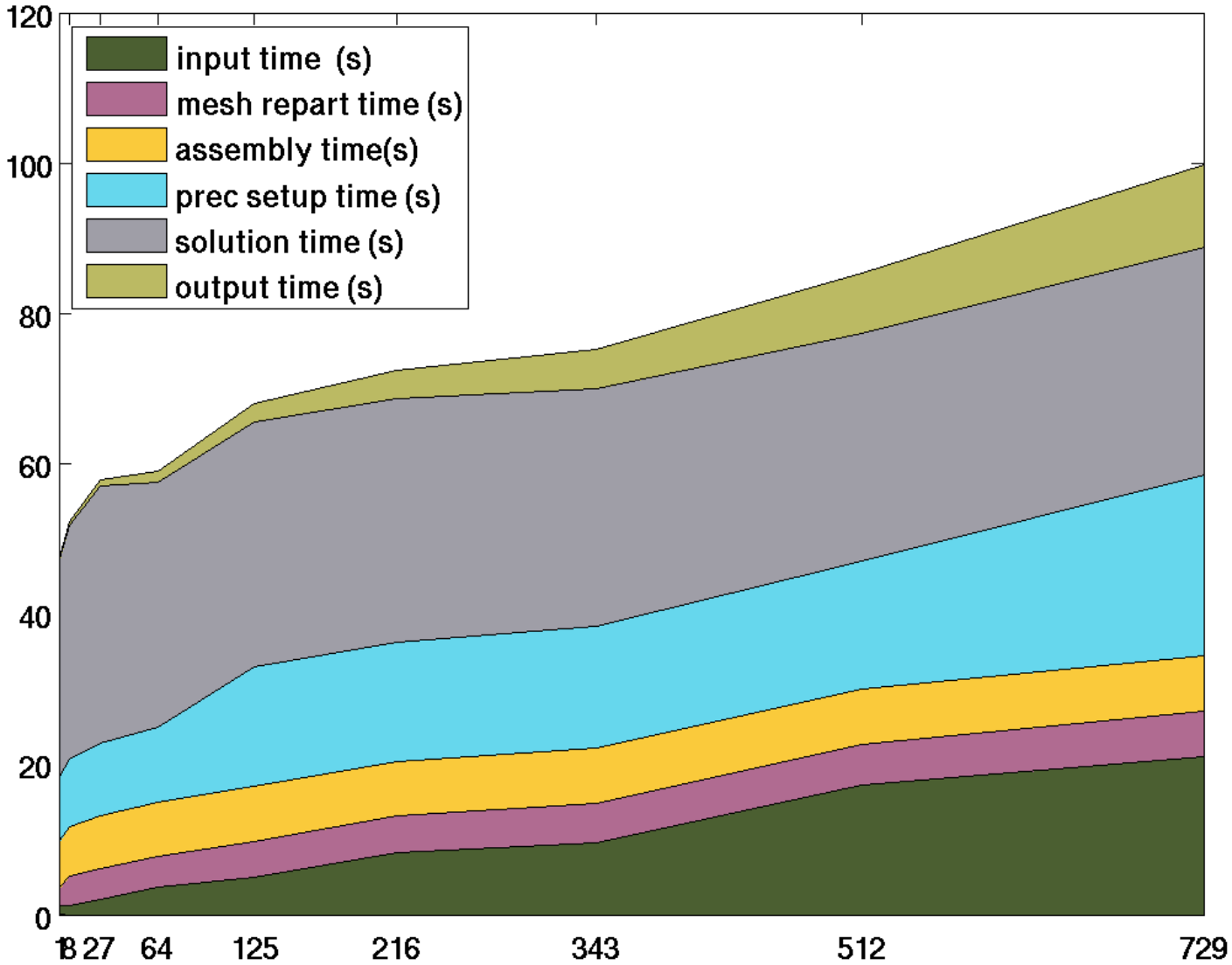
Reduction of residual error by a factor 10^5, i.e.,

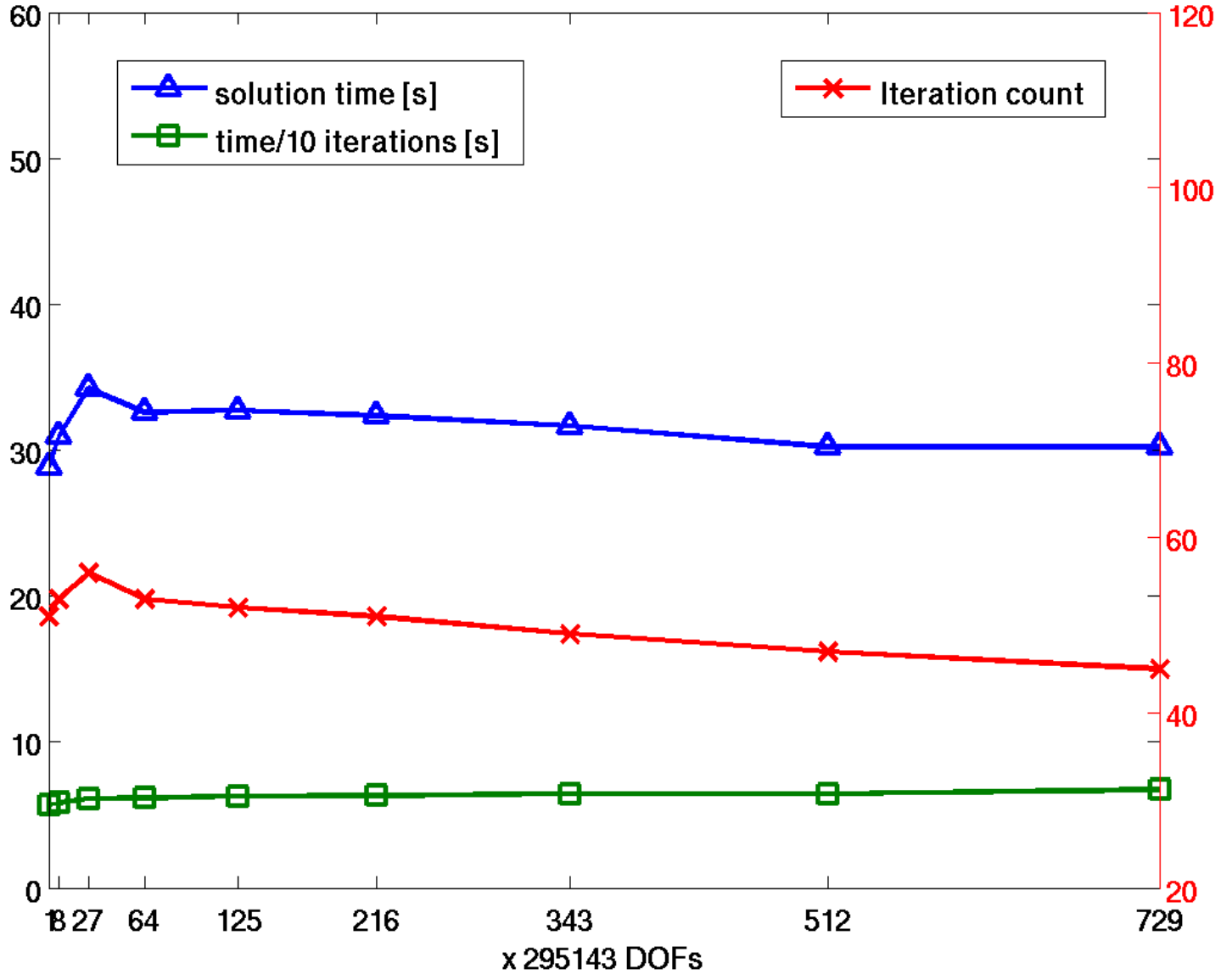
$$\|\mathbf{r}_k\|_2 = \|\mathbf{b} - A\mathbf{x}_k\|_2 < 10^{-5}\|\mathbf{b}\|_2.$$

Execution times (in seconds) and number of PCG interations of the weak scalability test on the Cray XT3.

CPUs	input	repart.	assembly	precond.	solution	output	total	iters
1	1.25	2.28	6.25	8.58	28.86	0.10	47.32	51
8	1.27	3.84	6.64	9.03	30.98	0.52	52.28	53
27	2.00	4.18	7.03	9.67	34.23	0.78	57.88	56
64	3.65	4.20	7.12	10.05	32.60	1.33	58.94	53
125	5.03	4.78	7.26	15.86	32.71	2.33	67.97	52
216	8.23	4.92	7.26	15.91	32.34	3.81	72.47	51
343	9.58	5.27	7.38	16.09	31.64	5.25	75.21	49
512	17.34	5.39	7.29	17.04	30.24	8.03	85.33	47
729	20.98	6.18	7.36	23.98	30.24	11.05	99.78	45

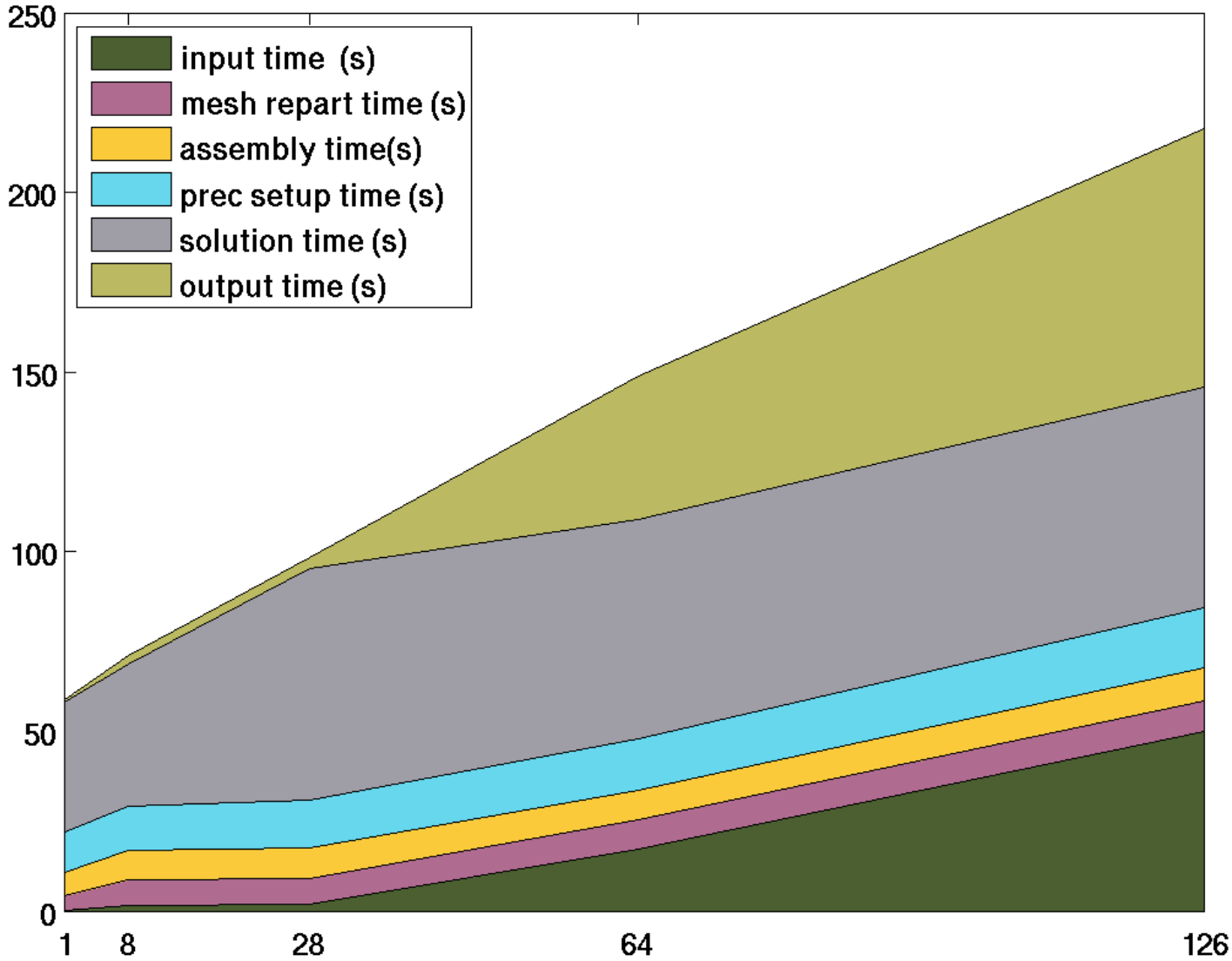
Problem size $n = \#$ CPUs $\times$ 295143

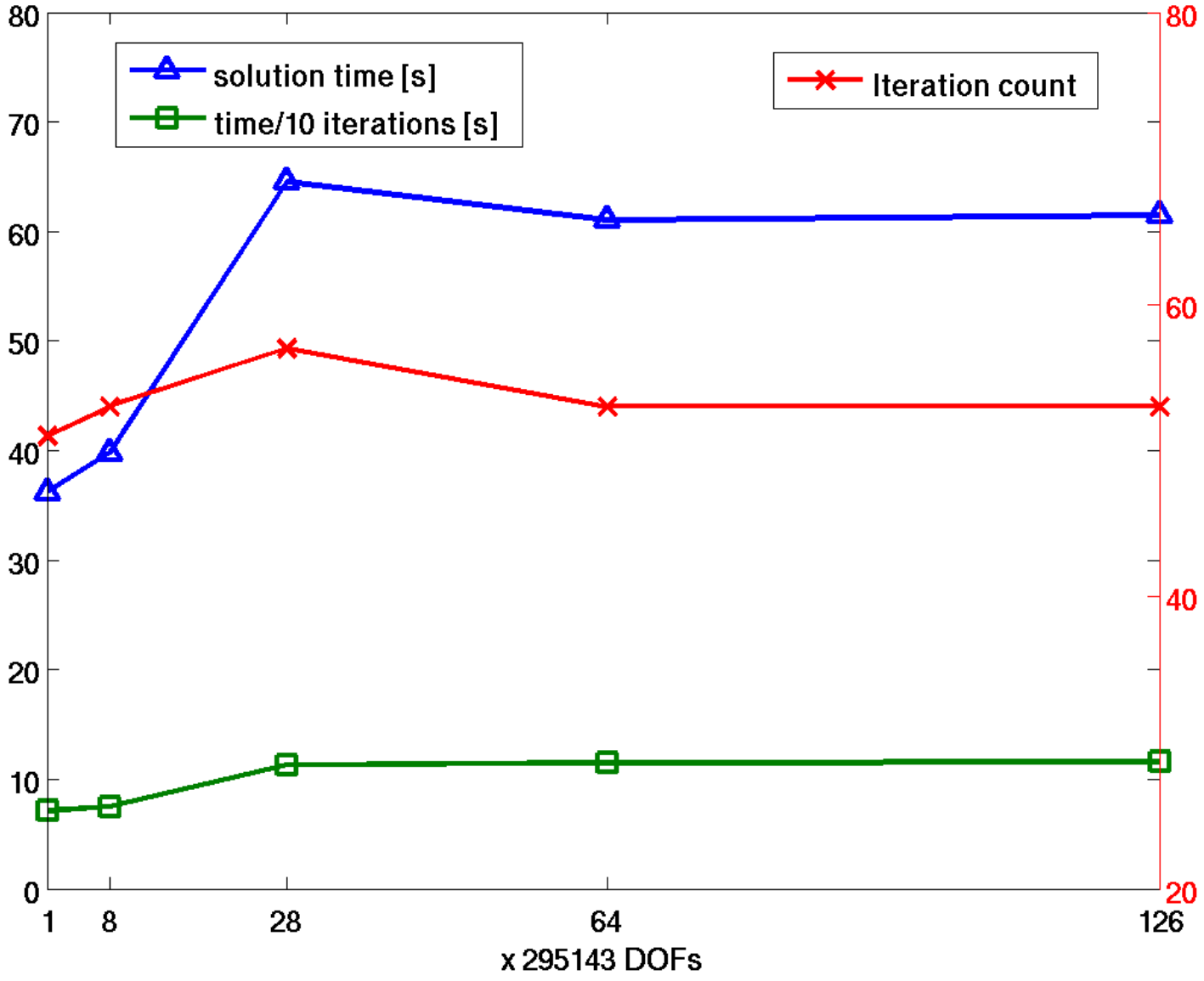

input time (s)
mesh repart time (s)
assembly time(s)
prec setup time (s)
solution time (s)
output time (s)
120
100
80
60
40
20
0
1
8
27
64
125
216
343
512
729


solution time [s]
time/10 iterations [s]
Iteration count
60
50
40
30
20
10
0
120
100
80
60
40
20
1
8
27
64
125
216
343
512
729
x 295143 DOFs

Execution times (in seconds) and number of PCG interations of the weak scalability test on Gonzales.

CPUs	input	repart.	assembly	precond.	solution	output	total	iters
1	0.28	4.05	6.34	11.29	36.19	0.46	58.61	51
8	1.64	7.17	7.97	12.25	39.81	2.18	71.03	53
27	1.89	7.28	8.41	13.25	64.54	3.16	98.53	57
64	17.3	8.09	8.13	14.54	61.03	39.67	148.78	53
125	49.8	8.61	9.33	16.64	61.49	71.98	217.88	53

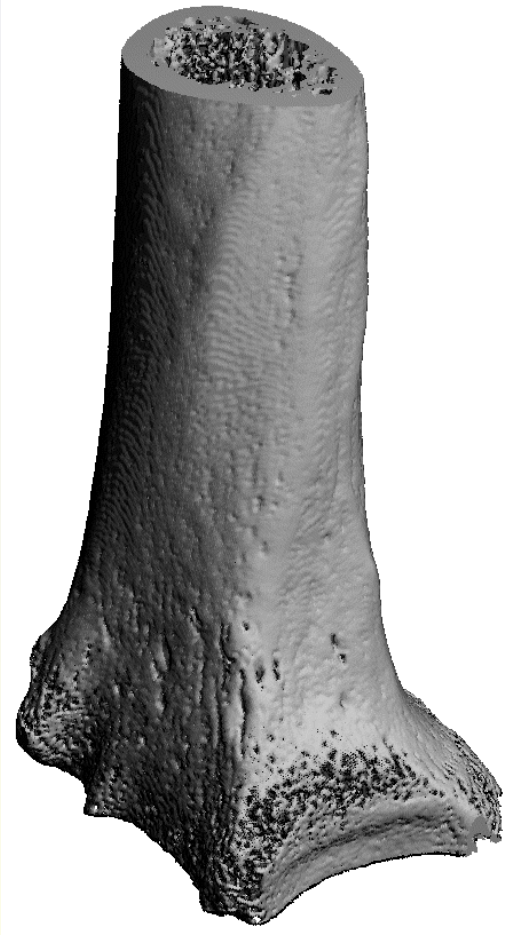

250
200
150
100
50
0
1
8
28
64
126
input time (s)
mesh repart time (s)
assembly time(s)
prec setup time (s)
solution time (s)
output time (s)

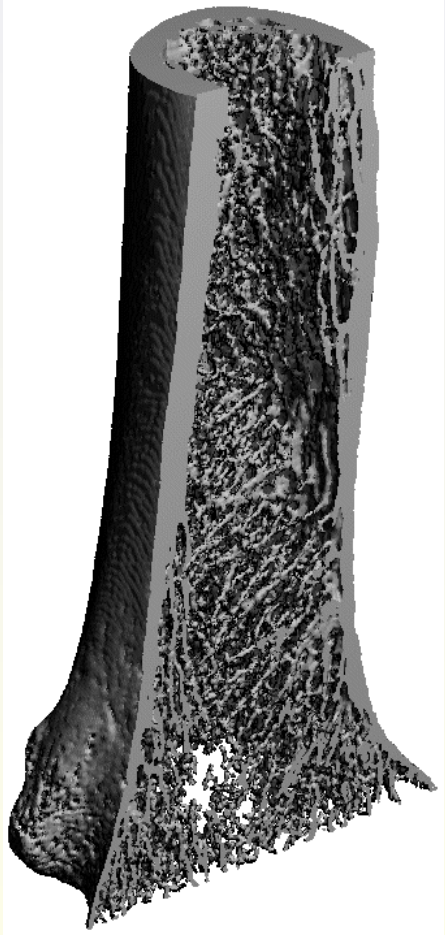

solution time [s]
time/10 iterations [s]
Iteration count
80
70
60
50
40
30
20
10
0
80
60
40
20
1
8
28
64
126
x 295143 DOFs

Weak scalability test conclusions

- All phases but the I/O scale very well.
- Poor scaling of I/O is mostly due to the limited number of available I/O nodes.
- The 200M degrees of freedom test is solved in less than 100 seconds on the Cray XT3.

Scalability test with real bone



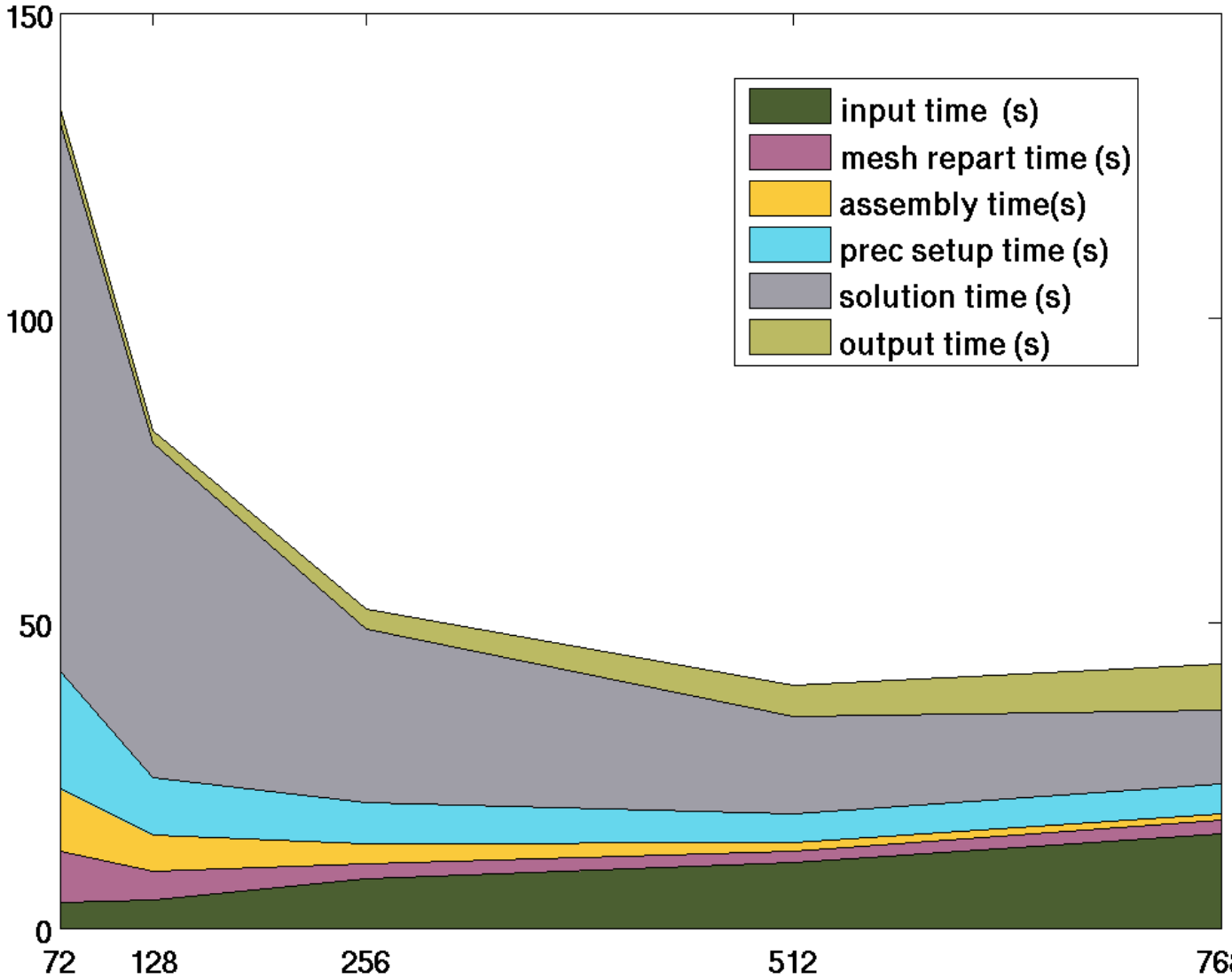


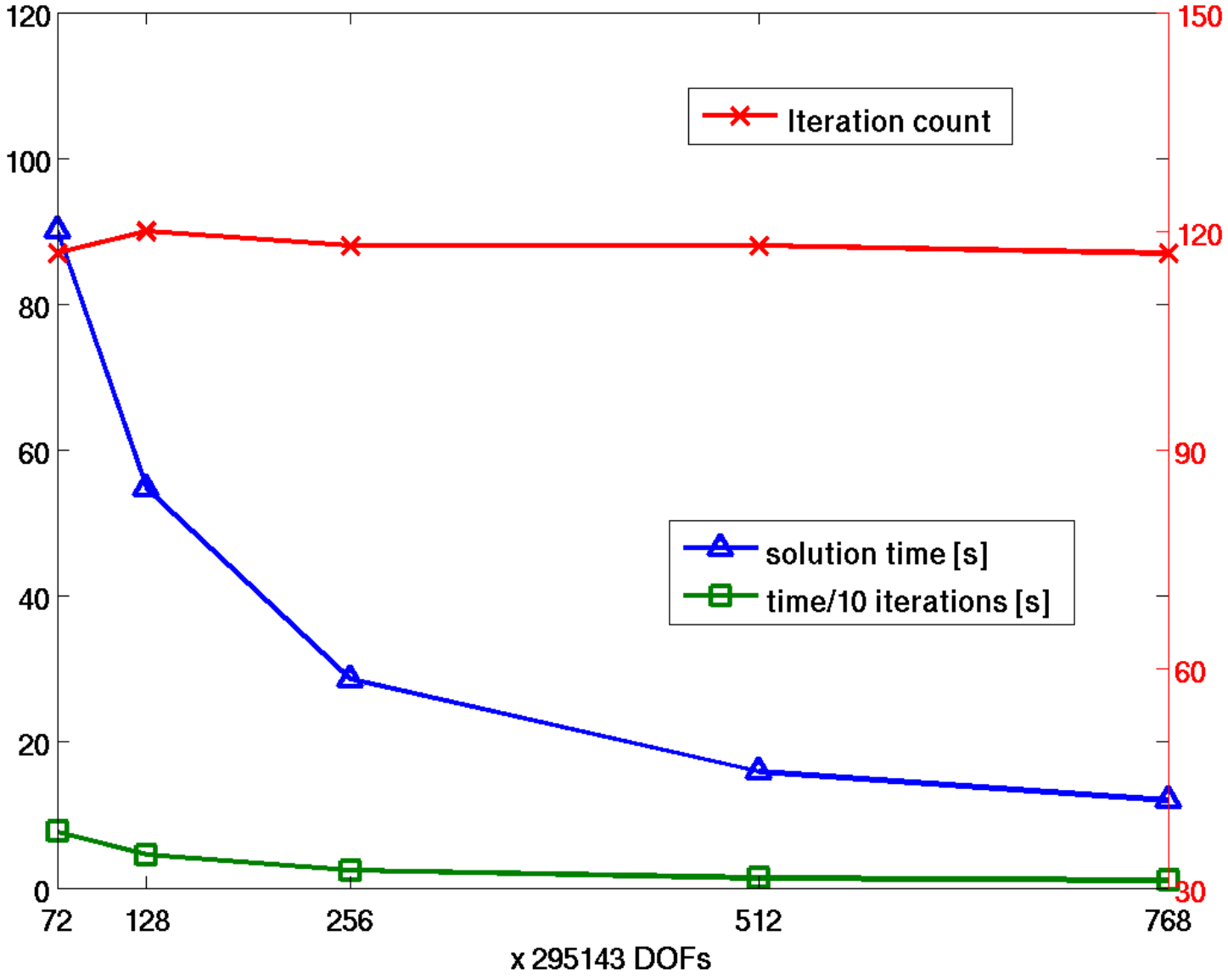
Distal part (20% of the length) of the radius in a human forearm.

Execution times (in seconds) and PCG interation count on the Cray XT3.

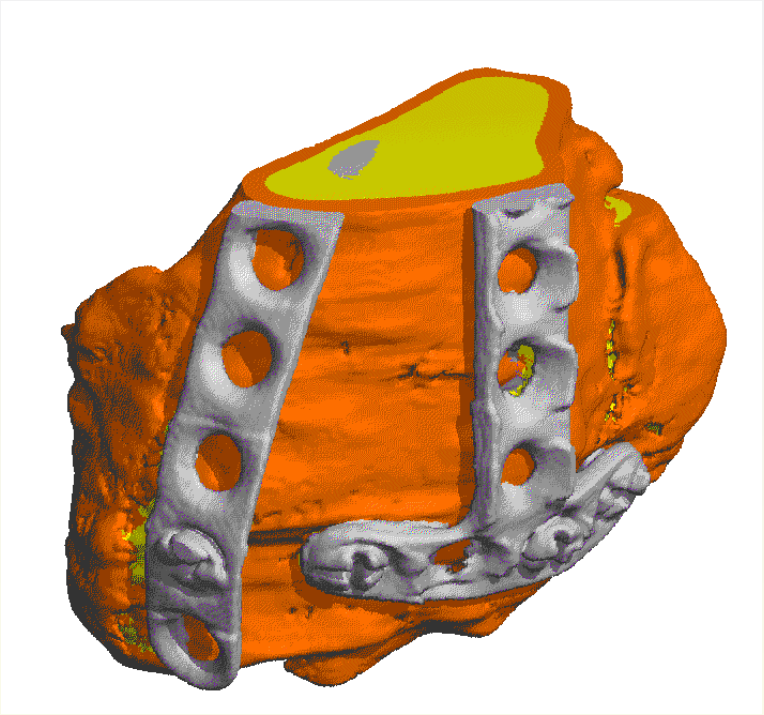
CPUs	input	repart.	assembly	precond.	solution	output	total	iters
72	4.10	8.47	10.2	19.3	90.1	2.22	134.	117
128	4.62	4.73	5.97	9.36	54.8	2.08	81.6	120
256	8.16	2.47	3.09	6.75	28.6	3.11	52.2	118
512	10.8	1.74	1.56	4.61	16.0	5.18	39.8	118
768	15.5	2.13	1.04	5.02	12.0	7.58	43.3	117

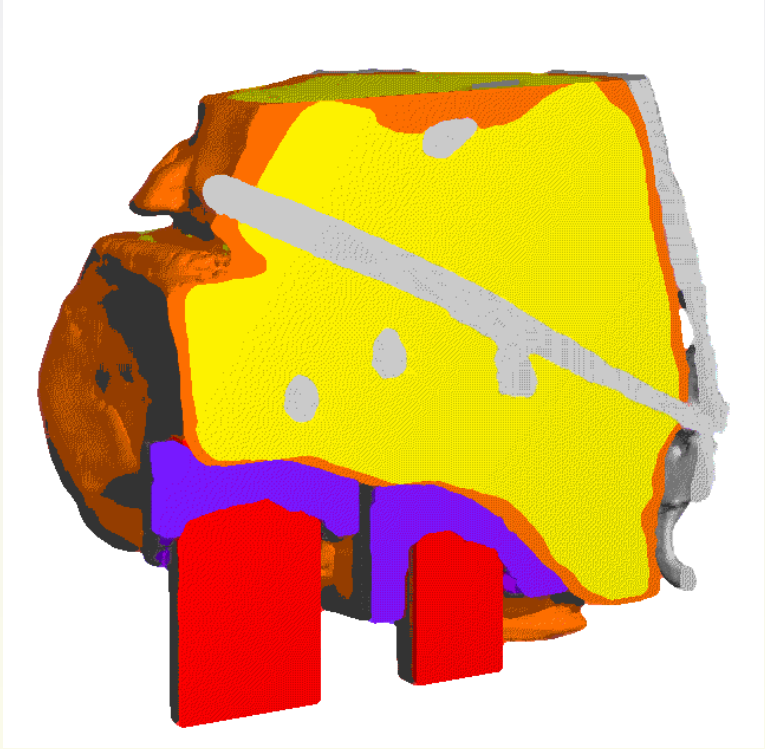
Fixed problem size $n = 23'284'416$.


150
100
50
0
72
128
256
512
768
input time (s)
mesh repart time (s)
assembly time(s)
prec setup time (s)
solution time (s)
output time (s)


120
100
80
60
40
20
0
150
120
90
60
30
Iteration count
solution time [s]
time/10 iterations [s]
72
128
256
512
768
x 295143 DOFs

Scalability test with real anisotropic 'bone'



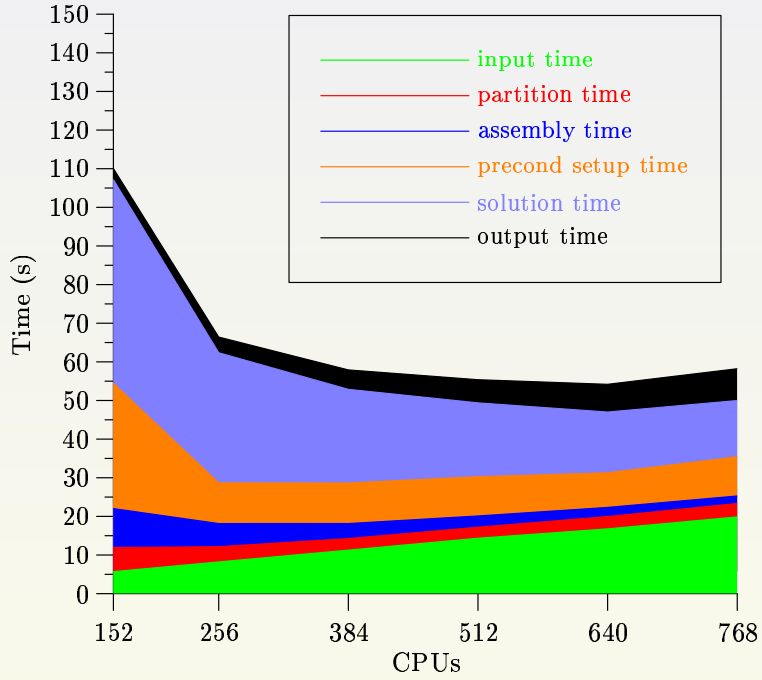


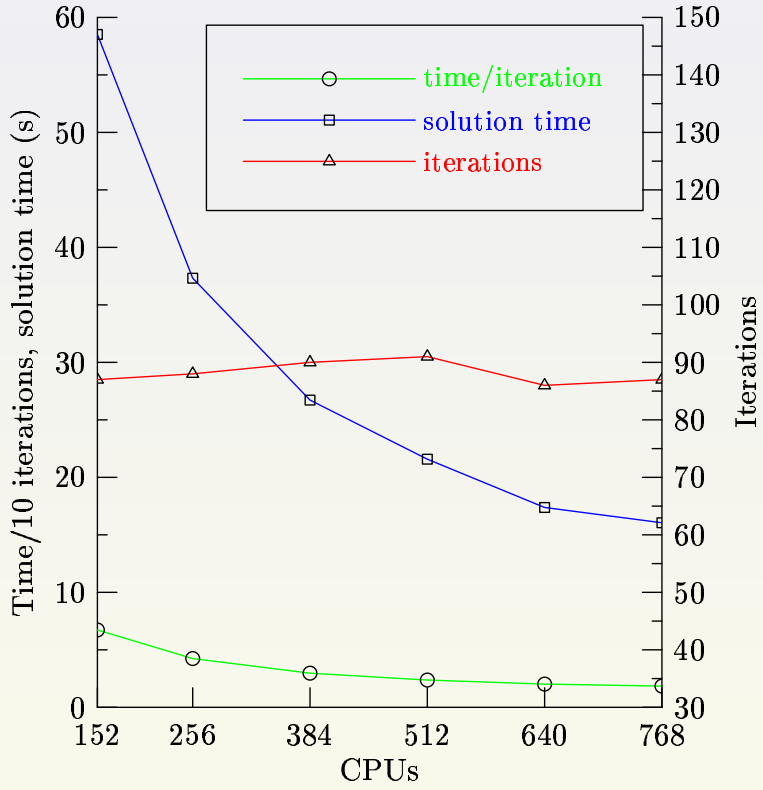
Operated distal radius fracture with fixation (varying elasticity modulus).

Execution times (in seconds) and PCG interation count on the Cray XT3.

CPUs	input	repart.	assembly	precond.	solution	output	total	iters
152	8.74	6.31	10.1	32.4	58.5	2.77	119.	87
256	13.9	3.92	5.99	10.5	37.3	3.78	75.4	88
384	12.2	2.99	3.93	10.5	26.7	4.91	61.3	90
512	13.4	2.83	2.91	10.2	21.6	6.02	56.9	91
640	16.7	3.18	2.36	8.95	17.4	7.54	56.1	86
768	19.4	3.45	1.95	10.1	16.0	8.37	59.3	87

Fixed problem size $n = 38'335'350$.


input time
partition time
assembly time
precond setup time
solution time
output time
Time (s)
CPUs
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
152
256
384
512
640
768


time/iteration
solution time
iterations
Time/10 iterations, solution time (s)
Iterations
CPUs
0
10
20
30
40
50
60
30
40
50
60
70
80
90
100
110
120
130
140
150
152
256
384
512
640
768

Conclusions

- Have devised an parallel highly scalable FE solver for bone structure analysis based PCG with SA multilevel preconditioner.
- Public domain software (Trilinos family, ParMETIS)
- Made the medical doctors happy.

Future Work

- Nonlinear (geometrically) elasticity necessary to model bone failure.
- Smooth surfaces / interfaces for better representation of pressures.

6th International Congress on Industrial and Applied Mathematics

iciam07

Zurich, Switzerland
16 - 20 July 2007

www.iciam07.ch