A highly scalable matrix-free multigrid solver for μFE analysis of bone structures based on a pointer-less octree

P. Arbenz, C. Flaig
ETH Zurich

Talk at SuperCA++, Bansko BG, Apr 23, 2012

Outline

The need for μFE analysis of bones I

Osteoporosis

- A disease characterized by low bone mass and deterioration of bone microarchitecture (trabecular bone).
- High lifetime risk for a fracture caused by osteoporosis.
  In Switzerland, the risk for an osteoporotic fracture in women above 50 years is about 50%, for men the risk is about 20%.
- Since global parameters like bone density do not admit to predict the fracture risk, patients have to be treated in a more individual way.

The need for μFE analysis of bones I

Osteoporosis

- A disease characterized by low bone mass and deterioration of bone microarchitecture (trabecular bone).
- High lifetime risk for a fracture caused by osteoporosis.
  In Switzerland, the risk for an osteoporotic fracture in women above 50 years is about 50%, for men the risk is about 20%.
- Since global parameters like bone density do not admit to predict the fracture risk, patients have to be treated in a more individual way.

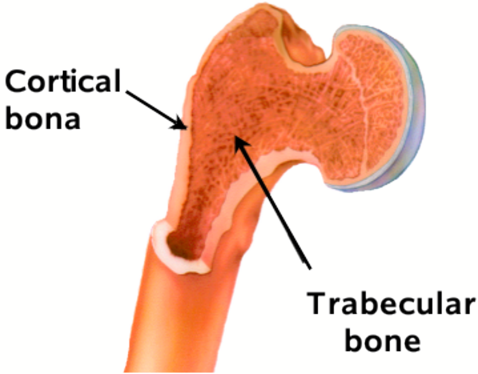
Research

μFE analysis improves the understanding of the importance of the structure of the trabecular bone.

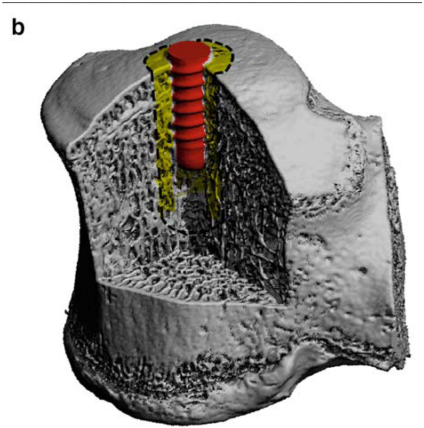
The need for μFE analysis of bones II

Progress

New approach consists of combining 3D high-resolution CT scans of individual bones with a micro-finite element (μFE) analysis.

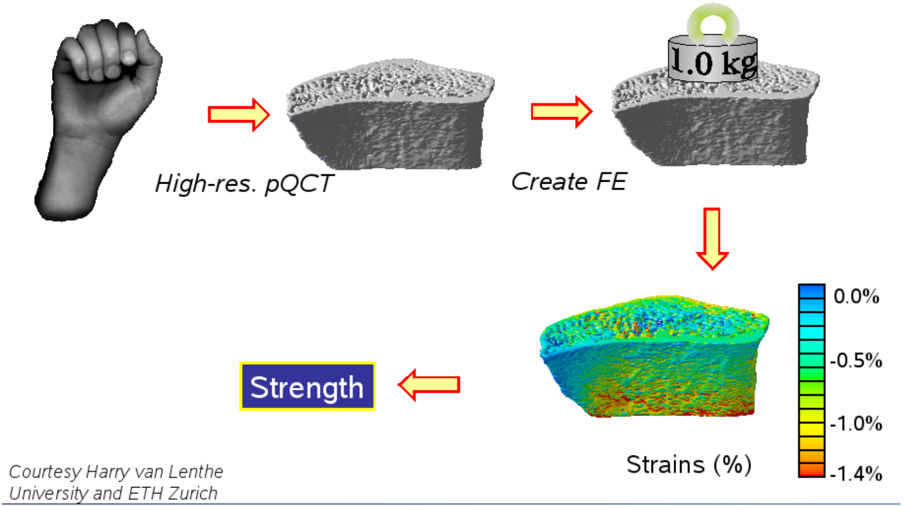


Another problem



A. Wirth *et al.*: *Mechanical competence of bone-implant systems can accurately be determined by image-based μFE analyses*. Arch. Appl. Mech. 80 (2010), 513–525.

Work flow



pQCT: Peripheral Quantitative Computed Tomography

The mathematical model

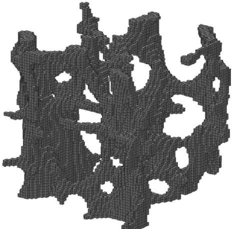
- Equations of linearized 3D elasticity (pure displacement formulation): Find displacement field $\mathbf{u}$ that minimizes total potential energy

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

with Lamé's constants $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$, $\mu = \frac{E}{2(1+\nu)}$, volume forces $\mathbf{f}$, boundary tractions $\mathbf{g}$, symmetric strain tensor

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

- Free boundary except top/bottom
- The computational domain Ω consist of identical voxels



The mathematical model

- Equations of linearized 3D elasticity (pure displacement formulation): Find displacement field **u** that minimizes total potential energy

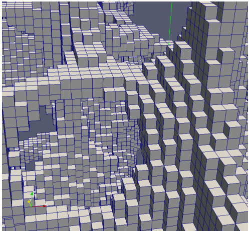
$$\int_\Omega \left[\mu\varepsilon(\mathbf{u}) : \varepsilon(\mathbf{u}) + \frac{\lambda}{2}(div\mathbf{u})^2 - \mathbf{f}^t\mathbf{u}\right] d\Omega - \int_{\Gamma_N} \mathbf{g}_S^t \mathbf{u} d\Gamma,$$

with Lamé's constants $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$, $\mu = \frac{E}{2(1+\nu)}$, volume forces **f**, boundary tractions **g**, symmetric strain tensor

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

- Free boundary except top/bottom

- The computational domain Ω consist of identical voxels



Discretization using voxel elements

- Finite element approximation: displacements $\mathbf{u}$ represented by piecewise trilinear polynomials
- Each voxel has 8 nodes
- In each node we have 3 degrees of freedom: displacements in x-, y-, z-direction
- In total 24 degrees of freedom per voxel
- strains / stresses computable by means of nodal displacements

Solving the system of equations

- The discretization results in a linear system of equation:

$$\mathbf{A}\mathbf{u} = \mathbf{f}$$

- **A** is sparse and symmetric positive definite.
- The fine resolution of the CT scan entails that **A** is HUGE.
- Approach to solve this linear system: preconditioned conjugate gradient (PCG) algorithm
  - Diagonal (Jacobi) preconditioner (Rietbergen et al., 1996)
  - Avoid the assembling of the stiffness matrix. Compute the matrix-vector multiplication in an element-by-element (EBE) fashion

$$A = \sum_{e=1}^{n_{\mathrm{el}}} T_e A_e T_e^T, \qquad (1)$$

  - Multigrid preconditioning (SA-AMG)

ParFE: Smoothed aggregation I (Trilinos ML package)

Parts of SA-AMG

- Prolongator
  - First level: Tentative (unsmoothed) prolongator formed from the aggregation of the matrix graph.
  - Second level and beneath: Smoothed aggregation
- Coarser level matrix: $K_{i+1} = P_i^T K_i P_i$
- Smoother: Chebyshev polynomial that is small on the upper part of the spectrum of K_ℓ

Memory savings about factor 3.5 (Arbenz et al., 2008)

2009: Solved a problem with $1.9 \cdot 10^9$ dofs at CSCS.

- *ParMETIS → Recursive Coordinate Bisection*
- Limiting number of levels (no direct solving on coarsest level)

ParFE: Smoothed aggregation I (Trilinos ML package)

Parts of SA-AMG

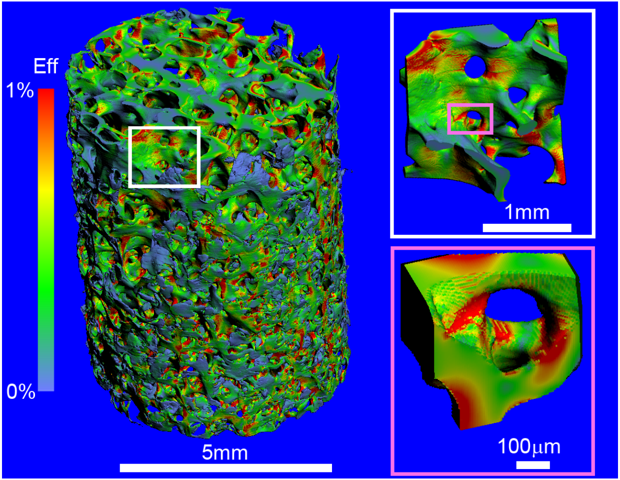
- Prolongator
  - First level: Tentative (unsmoothed) prolongator formed from the aggregation of the matrix graph.
  - Second level and beneath: Smoothed aggregation
- Coarser level matrix: $K_{i+1} = P_i^T K_i P_i$
- Smoother: Chebyshev polynomial that is small on the upper part of the spectrum of K_ℓ

Memory savings about factor 3.5 (Arbenz et al., 2008)

2009: Solved a problem with $1.9 \cdot 10^9$ dofs at CSCS.

- *ParMETIS* $\rightarrow$ *Recursive Coordinate Bisection*
- Limiting number of levels (no direct solving on coarsest level)

Very large real bone



Effective strains with zooms.
(Image by Jean Favre, CSCS)

"Full space" approach

Motivation

- We did not exploit regular structures except on finest level.
- To reduce overhead for information on unstructured grids:
  - Use regular grids on all levels.
  - Apply geometric multigrid
  - Stay matrix-free on all levels
- First approach in this direction:
  - Embed the bone structure in a cuboid with original regular grid extended.
  - The 'empty space' is assumed to be filled with very soft material (Margenov, 2006).

- Goal is to have an algorithm with high memory efficiency.

"Full space" approach

Motivation

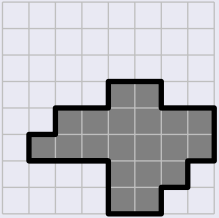
- We did not exploit regular structures except on finest level.
- To reduce overhead for information on unstructured grids:
  - Use regular grids on all levels.
  - Apply geometric multigrid
  - Stay matrix-free on all levels
- First approach in this direction:
  - Embed the bone structure in a cuboid with original regular grid extended.
  - The 'empty space' is assumed to be filled with very soft material (Margenov, 2006).

- Goal is to have an algorithm with high memory efficiency.

"Full space" approach (con't)

Parts of "full space" approach

- Empty region also meshed
  $\Longrightarrow$ increased number of dof's



- Poisson's ratio (ν) constant, Young's modulus (E) can vary
- Geometric multigrid preconditioner
  - Prolongator: trilinear interpolation
  - Smoother: Chebyshev polynomial
  - Coarser level problem: average of the Young's modulus of the encased voxels
  - Iterative solver with limited precision on coarsest level

"Full space" approach (con't)

Parts of "full space" approach II

- All matrix-vector multiplications are implemented in a element-by-element manner

$$K_\ell = \sum_{e=1}^{n_{\mathrm{el}}^\ell} E_e^\ell \, T_e^\ell \, K_e^\ell \, (T_e^\ell)^T,$$

$$E_{x,y,z}^{\ell+1} = \frac{1}{8} \sum_{i,j,k=0}^{1} E_{2x+i,2y+j,2z+k}^\ell, \qquad K_e^{\ell+1} = \frac{1}{8} K_e^\ell.$$

- Perfect weak scalability up to 8000 cores on a Cray XT5
- Largest problem had $16 \cdot 10^9$ dofs (Flaig, Arbenz, LSSC'11)

"Full space" approach (con't)

Parts of "full space" approach II

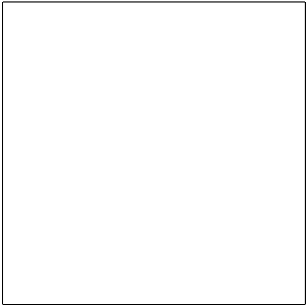
- All matrix-vector multiplications are implemented in a element-by-element manner

$$K_\ell = \sum_{e=1}^{n_{\text{el}}^\ell} E_e^\ell \, T_e^\ell \, K_e^\ell \, (T_e^\ell)^T,$$

$$E_{x,y,z}^{\ell+1} = \frac{1}{8} \sum_{i,j,k=0}^{1} E_{2x+i,2y+j,2z+k}^\ell, \qquad K_e^{\ell+1} = \frac{1}{8} K_e^\ell.$$

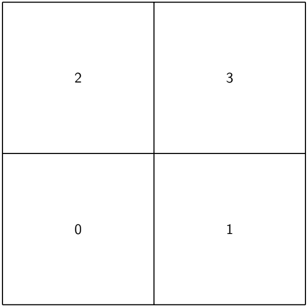
- Perfect weak scalability up to 8000 cores on a Cray XT5
- Largest problem had $16 \cdot 10^9$ dofs (Flaig, Arbenz, LSSC'11)

Octree approach



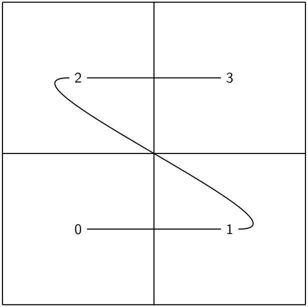
- 1D: binary tree
  - intervals are divided into 2 subintervals
- 2D: quadtree
  - squares are divided into 4 subsquares
- 3D: octree
  - cubes are divided into 8 subcubes

Octree approach



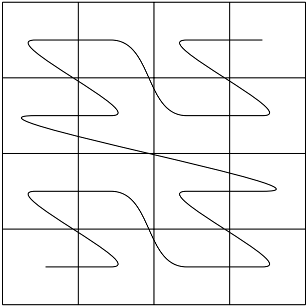
- 1D: binary tree
  - intervals are divided into 2 subintervals
- 2D: quadtree
  - squares are divided into 4 subsquares
- 3D: octree
  - cubes are divided into 8 subcubes

Octree approach



- 1D: binary tree
  - intervals are divided into 2 subintervals
- 2D: quadtree
  - squares are divided into 4 subsquares
- 3D: octree
  - cubes are divided into 8 subcubes

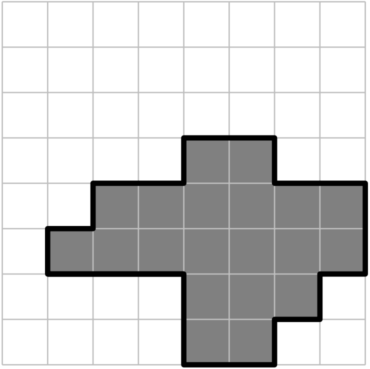
Octree approach



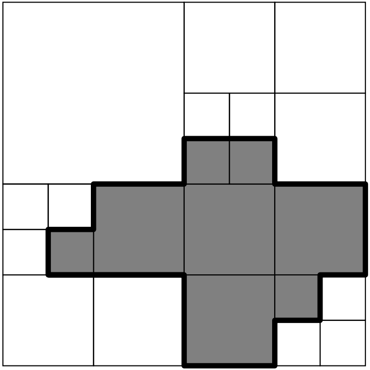
- 1D: binary tree
  - intervals are divided into 2 subintervals
- 2D: quadtree
  - squares are divided into 4 subsquares
- 3D: octree
  - cubes are divided into 8 subcubes

$\rightarrow$ Depth first traversal results in the *Morton ordering (Z curve)*

Octree approach (con't)

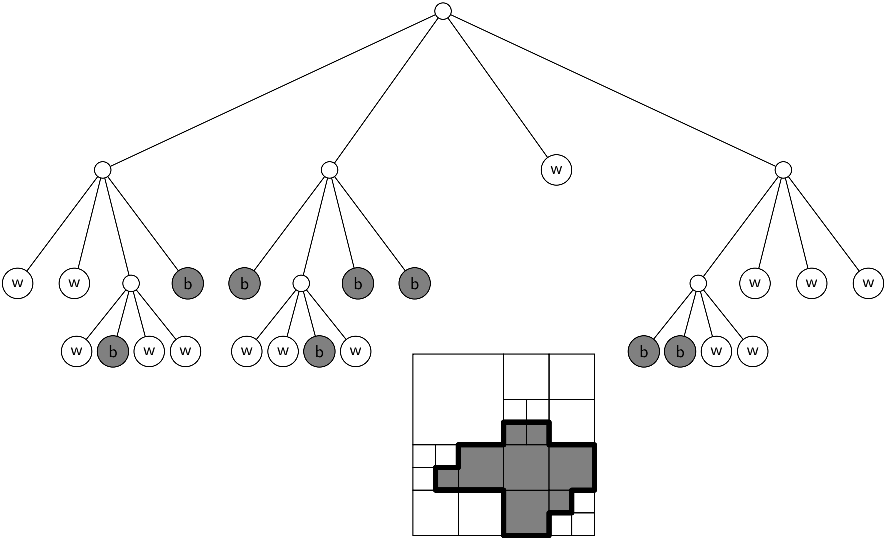


Left: finest level



right: various levels of a quadtree

Octree approach (con't)

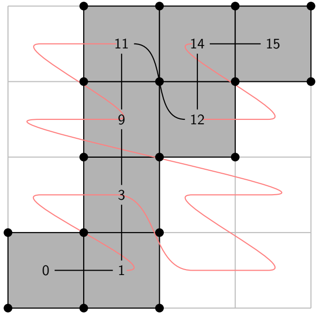


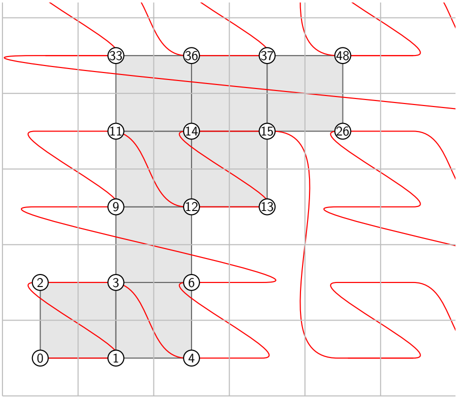
Octree approach (geometric multigrid)

Parts of octree approach

- Same setup as the "full space" approach
  - Geometric multigrid preconditioner:
    - Prolongator: Trilinear interpolation
    - Smoother: Chebyshev polynomial
    - Coarser level problem: Average of the Young's modulus of the encased voxels
    - Iterative solver with limited precision on coarsest level
  - Poisson's ratio (ν) constant, Young's modulus (E) can vary
- Computational region is stored in a hierarchical data structure Octree (H. Sundar et al., 2007), Space filling curve (M. Mehl et al., 2006), `p4est` (Burstedde et al., 2011)
  - Linearized octree
  - Nodes are stored according to their Morton ordering (Z curve)
  - Optimized search strategy to access the nodes of the elements

Numbering

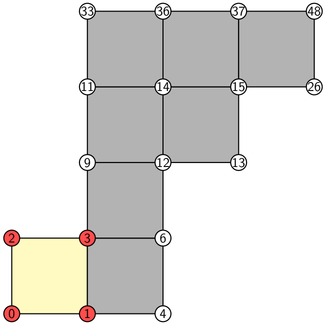




0	1	2	3	4	6	9	11	12	13	14	15	26	33	36	37	48

- Coordinate to key transformation: $key = \cdots y_3 x_3 y_2 x_2 y_1 x_1 y_0 x_0$

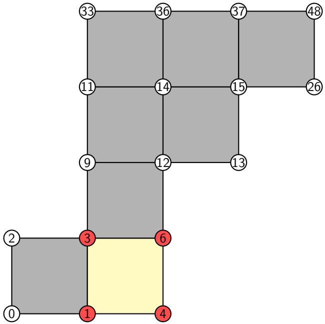
Accessing the nodes



0	1	2	3	4	6	9	11	12	13	14	15	26	33	36	37	48

- Left lower node has always the smallest key
- Space filling curve leads to locality of the nodes in the array
- Interval increasing by factor of 4 corresponds to ascending one level in the quadtree

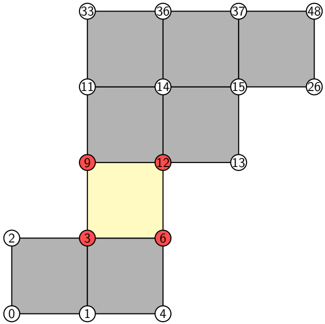
Accessing the nodes



0	1	2	3	4	6	9	11	12	13	14	15	26	33	36	37	48

- Left lower node has always the smallest key
- Space filling curve leads to locality of the nodes in the array
- Interval increasing by factor of 4 corresponds to ascending one level in the quadtree

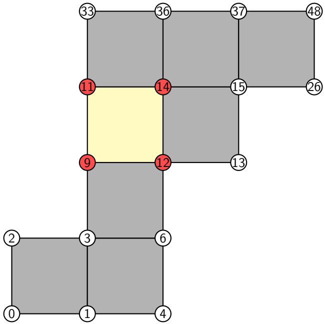
Accessing the nodes



0	1	2	3	4	6	9	11	12	13	14	15	26	33	36	37	48

- Left lower node has always the smallest key
- Space filling curve leads to locality of the nodes in the array
- Interval increasing by factor of 4 corresponds to ascending one level in the quadtree

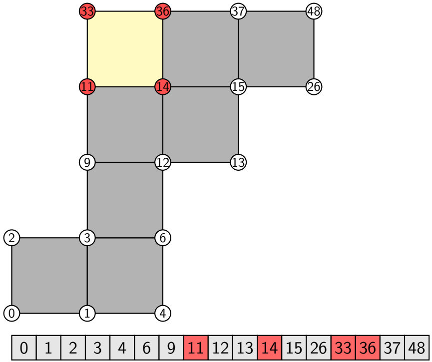
Accessing the nodes



0	1	2	3	4	6	9	11	12	13	14	15	26	33	36	37	48

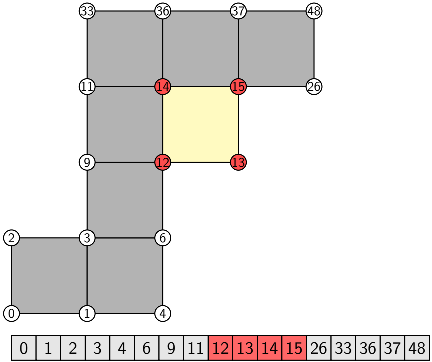
- Left lower node has always the smallest key
- Space filling curve leads to locality of the nodes in the array
- Interval increasing by factor of 4 corresponds to ascending one level in the quadtree

Accessing the nodes



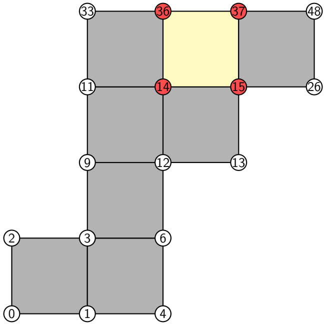
- Left lower node has always the smallest key
- Space filling curve leads to locality of the nodes in the array
- Interval increasing by factor of 4 corresponds to ascending one level in the quadtree

Accessing the nodes



- Left lower node has always the smallest key
- Space filling curve leads to locality of the nodes in the array
- Interval increasing by factor of 4 corresponds to ascending one level in the quadtree

Accessing the nodes

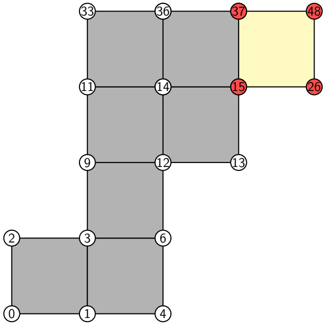


0	1	2	3	4	6	9	11	12	13	14	15	26	33	36	37	48

- Left lower node has always the smallest key
- Space filling curve leads to locality of the nodes in the array
- Interval increasing by factor of 4 corresponds to ascending one level in the quadtree

function int **SearchIndex**(int *start*, t_octree_key *key*, t_tree *tree*)
  int $count = 1$;
  while key $> tree[start + count].key$ **do**
    $count = count \cdot 4$;
  end while
  return binarySearch($start + count/4$, $start + count$, key, $tree$);

Accessing the nodes

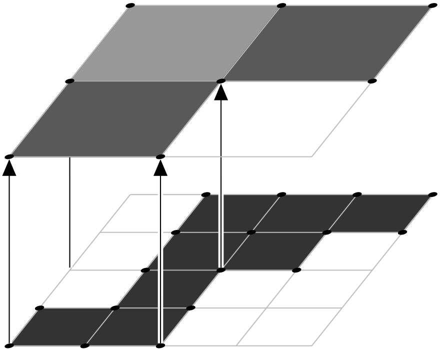


0	1	2	3	4	6	9	11	12	13	14	15	26	33	36	37	48

- Left lower node has always the smallest key
- Space filling curve leads to locality of the nodes in the array
- Interval increasing by factor of 4 corresponds to ascending one level in the quadtree

function int **SearchIndex**(int *start*, t_octree_key *key*, t_tree *tree*)
  int $count = 1$;
  while key $> tree[start + count].key$ **do**
    $count = count \cdot 4$;
  end while
  return binarySearch($start + count/4$, $start + count$, key, $tree$);

Coarsening



$$key = \cdots y_3 x_3 y_2 x_2 y_1 x_1 y_0 x_0$$

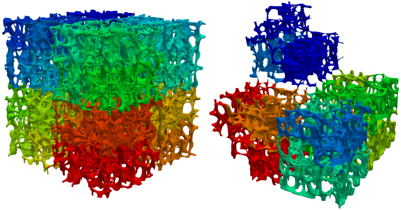
$$coarsekey = \cdots y_3 x_3 y_2 x_2 y_1 x_1$$

$$coarsekey = \frac{key}{4}$$

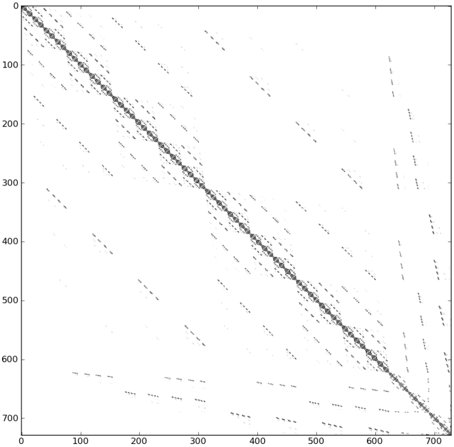
On the coarser level: Young's modulus is obtained by averaging the Young's moduli of the encased voxels.

Partitioning and load balancing

- Key objective: equal memory usage per core
- Distribution to the cores is done according to space filling curve
- Equally sized sets of contiguous nodes
- Coarser levels are not redistributed



Communication scheme:



Memory consumption

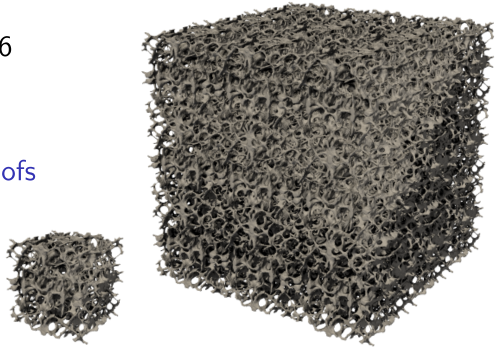
Memory consumption [Bytes per dof](measured values)

	SA-AMG	“full space”	“Octree”
Mesh	366	2.5	5.5
Matrix, PCG	199	32	32
Preconditioner	523	39.5	43
temporary memory	$\sim$400	$\sim$7	$\sim$12
Total	$\sim$1488	$\sim$81	$\sim$92.5

- About 15 mio. dofs per compute core with 1.33 GB memory.
- SA-AMG needs more than 16$\times$ more memory.
- For dense bone, “full space” approach should be chosen.

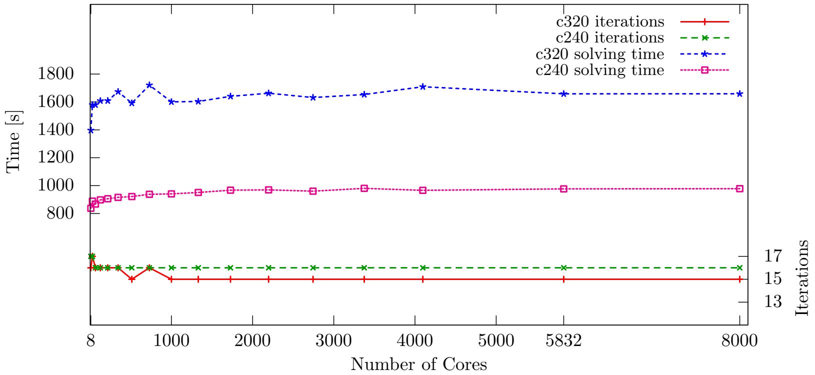
Weak scalability

- Weak scalability test with two meshes:
  - c240 is encased in a 240^3 cube with $6.98 \cdot 10^6$ dofs
  - c320 is encased in a 320^3 cube with $11.8 \cdot 10^6$ dofs
- Bigger grids are generated by "3D-mirroring"
- Chebyshev smoother deg 6
- Stopping criterion: $||\mathbf{r}_k||_{M^{-1}} \leq 10^{-6}||\mathbf{r}_0||_{M^{-1}}$
- Biggest mesh: $388 \cdot 10^9$ dofs



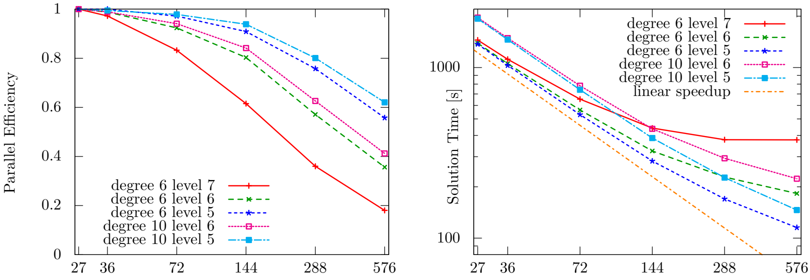
Weak scalability ($94 \cdot 10^9$ dofs)

Smoother: Cheb. deg. 6, tol. 10^{-6}



- Timings are measured on a Cray XT5 at CSCS
- The iteration counts varies between 15 and 17
- Very good scaling up to 8000 cores.

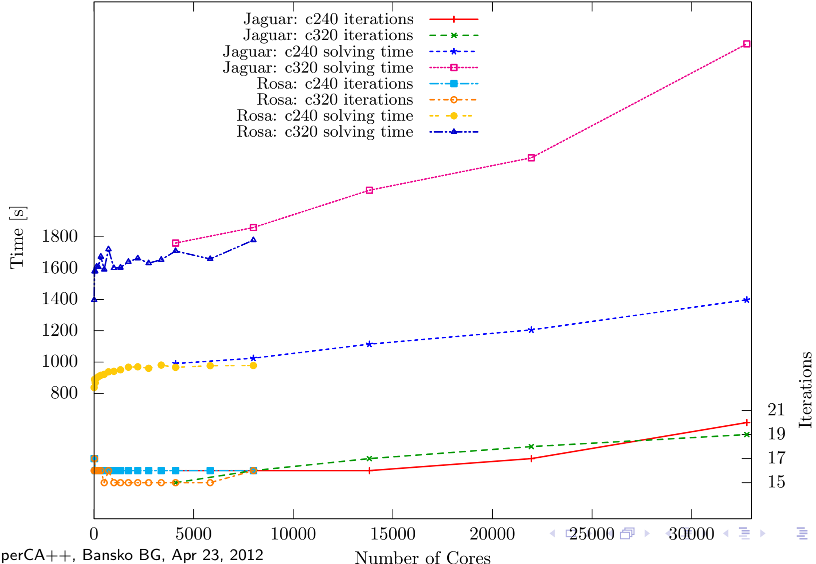
Strong scalability



- Execution times measured on a Cray XT5 at CSCS
- Stopping criterion: $||\mathbf{r}_k||_{M^{-1}} \leq 10^{-6}||\mathbf{r}_0||_{M^{-1}}$
- Mesh: c320 3× "3D-mirrored"

- Good scaling from 27 cores to 576 cores
- Higher smoother degree results in a better efficiency
- It is worth to limit the number of levels

Extended weak scalability on Cray XT5 Jaguar at Oak Ridge NL (388 billion dofs)



Conclusion, future work

Conclusions

- Solved huge problem with $388 \cdot 10^9$ dofs.
  Real bones up to $3.1 \cdot 10^9$ dofs.
- New approach has $16\times$ smaller memory footprint than ParFE.
- Perfect weak scalability and a good strong scalability.
- Space filling curve is a good tool for partitioning.
- Code is cache efficient.

Future work

- Improve the access of the node by using low collision hashing.
- Individual redistribution of the coarser meshes.
- Find a better homogenization procedure.
- GPU implementation (full space approach).

References

[1] P. Arbenz, G. H. van Lenthe, U. Mennel, R. Müller, and M. Sala. *A scalable multi-level preconditioner for matrix-free μ-finite element analysis of human bone structures*. Internat. J. Numer. Methods Eng. 73(7), 927–947, 2008.

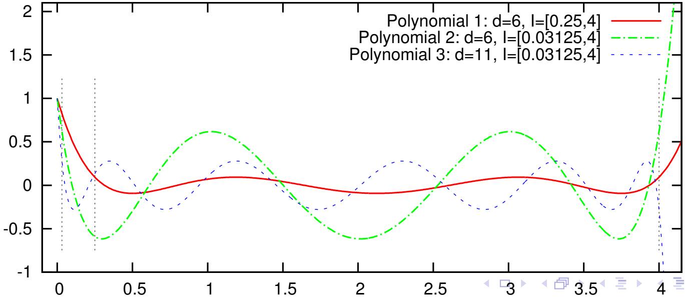
[2] C. Flaig and P. Arbenz. *A scalable memory efficient multigrid solver for micro-finite element analyses based on CT images*, Parallel Comput. 37(12) 846–854, 2011.

[3] C. Flaig and P. Arbenz. *A highly scalable matrix-free multigrid solver for μFE analysis based on a pointer-less octree*. Proceedings LSSC 2011. Springer Lecture Notes in Computer Science 7116, 498–506.

Chebyshev smoother

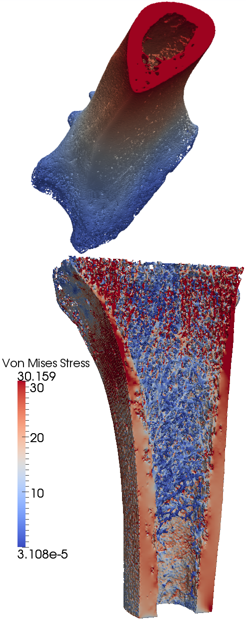
Smoother

- Degree 10 polynomial of the diagonally scaled matrix
- Polynomial is chosen to be minimal on $[\lambda_{min}, \lambda_{max}]$
- The spectrum (biggest, smallest eigenvalue) is estimated with 10 steps of Lanczos algorithm.
- Important to set λ_{min} relative to λ_{max}: $\lambda_{min} = \lambda_{max}/16$



Real Bone

Smoother: Cheb. deg. 6, tol. 10^{-6}



- Part of the human radius
- Global size: $303 \times 459 \times 553$
- $9 \cdot 10^6$ elements, $36.5 \cdot 10^6$ dofs

Prec	#iter	time [s]
GMG	54	1336
Jac.	6827	5891