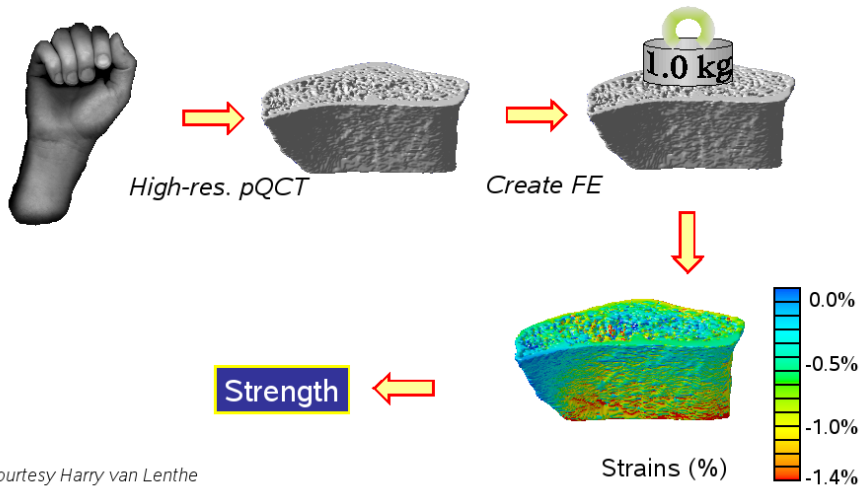


Usage of HDF5 in Finite Element Analysis of Bone Structures

Cyril Flaig
Inst. Comput. Science
ETH Zurich

HDF5 Tutorial 9. / 12. September

Overview



Courtesy Harry van Lenthe
University and ETH Zurich

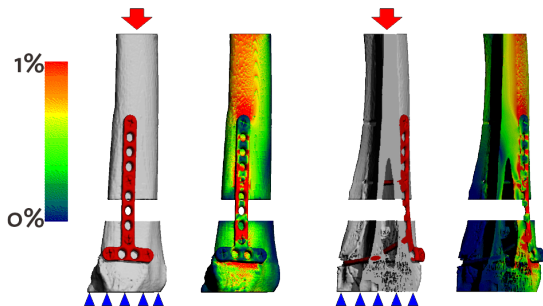
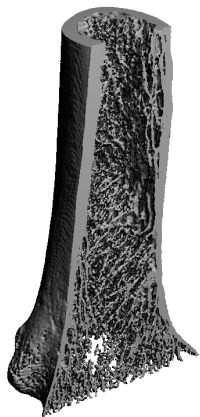
pQCT: Peripheral Quantitative Computed Tomography

ParFE: Parallel Finite Element Solver

- ParFE is written in C++
- Based on high-performance libraries for scientific computing (BLAS, LAPACK, MPI, Trilinos, ParMETIS and HDF5)
- Portable code and files (BlueGene, Cray, MacOS, Linux PC)
- Highly scalable (we solved models with up to 1.6 billion dof)
- Supporting of different materials in a model (enables analysis of fixed fracture)
- Models can consist of either hexahedra or tetrahedra (3D linear / trilinear elements)
- Support of different boundary conditions (loaded nodes, fixed displacement on nodes)

<http://parfe.sourceforge.net/index.php>

Example Models



Requirements of the Data File Format

- Transferability (different architectures)
- Parallel I/O
- 1 file per simulation:
 - Different types
 - Grouping (e.g.: Configuration, Mesh, Solution)
 - Flexible size of a group
- File size > 2 GByte (8MB - 20GB)
- well-documented API

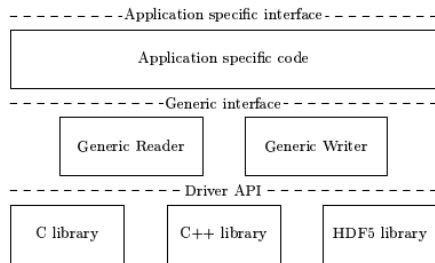
Structure

ParFE uses the following hierarchical Structures:

- Parameters: mostly scalar values
 - Properties of the mesh (e.g.: dimension, material types..)
 - Basic parameters of the simulation (e.g.: tolerance, iteration limit, amount of integration points)
- Mesh
 - Coordinates: Type H5T_IEEE_F64LE, Dataspace SIMPLE
 - Elements: Type H5T_STD_I32LE, Dataspace SIMPLE
 - Material IDs: Type H5T_STD_I32LE, Dataspace SIMPLE
- Boundary Conditions
 - This group stores several vectors used to describe the boundary conditions
- Solution
 - Solution of the simulation consisting of displacements, forces, stresses and strains.

Implementation

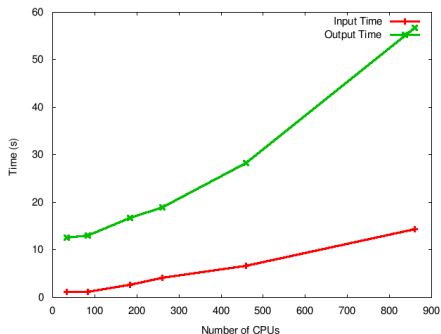
- ParFE uses a layered I/O model, due to the support of different file formats
- Generic Reader/Writer interfaces to the HDF5 library
- Templates are used in the Generic Reader/Writer to generate the read/write function for different types



- Implemented by U. Mennel and M. Sala

Performance

- Real measurement on the Cray XT3 at the CSCS
- File size is about 60 MB per CPU (about 1 mio dof per CPU) including the solution (about 66% Solution)



- Scales very well
- Bounded by the number of I/O nodes

Conclusion

HDF5 enables us in our project:

- Portability and transferability
- Fast and parallel I/O
- Easy parsing through the hierarchical structure
- Compact format enabled by binary storage of the data with small overhead
- Need 60% of storage of “equal” ASCII file (w/o HDF5 compression)