Enabling hybrid tree-based Adaptive Mesh Refinement using Pyramids

David Knapp (German Aero- Spacecenter, Scientific Computing),
Johannes Holke (German Aero- Spacecenter, Scientific Computing),
Carsten Burstedde (University of Bonn)

DLR

Topics

Tree-based AMR

A very brief Introduction into `t8code`

SFC for pyramids

Results

Tree-based AMR

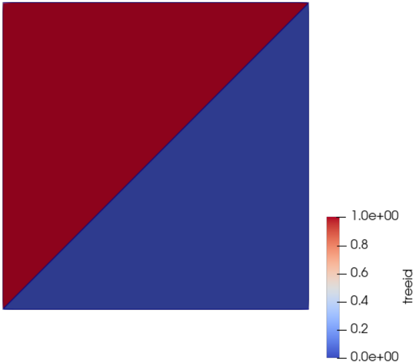


Figure: The coarse Mesh

Tree-based AMR

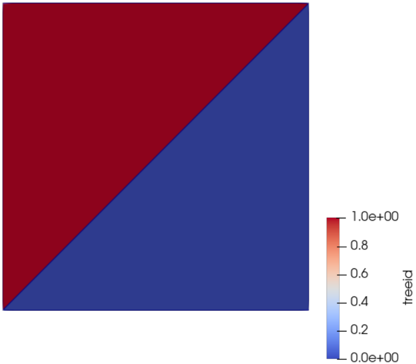


Figure: The coarse Mesh

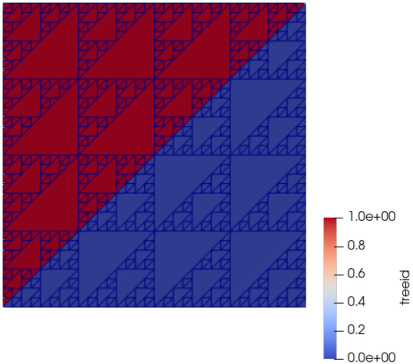
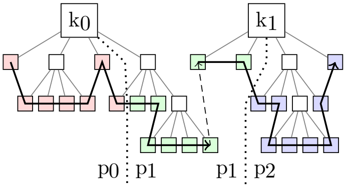
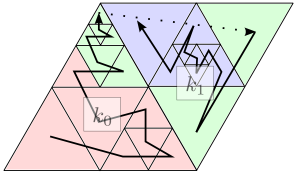


Figure: The fine Mesh

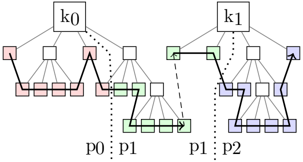
Tree-based AMR

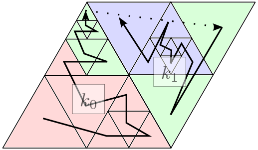




Organize each tree via a space-filling curve

Tree-based AMR

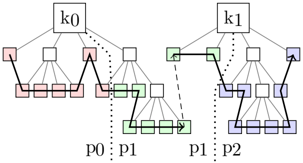


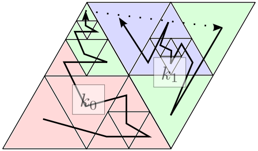


SFC

1. Restricted to a level, the index is unique

Tree-based AMR

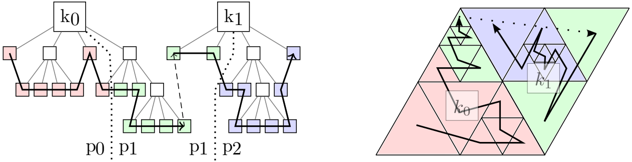




SFC

1. Restricted to a level, the index is unique
2. Refining does not decrease the index

Tree-based AMR



SFC

1. Restricted to a level, the index is unique
2. Refining does not decrease the index
3. Refining is local

`t8code`

High-level algorithms

- Adapt
- Partition
- Balance
- Ghost
- . . .

t8code

High-level algorithms

- Adapt
- Partition
- Balance
- Ghost
- . . .

Low-level algorithms

- Child
- Parent
- Neighbor
- Successor
- . . .

The High-level algorithms are independent of the implementation of the elements.

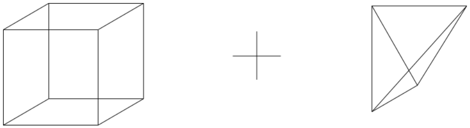
Why do we need Pyramids?

source: https://commons.wikimedia.org/wiki/File:Seattle_-_Smith_Tower_01.jpg

What about 3D?

Hybrid AMR

We want to use tetrahedra and hexahedra,

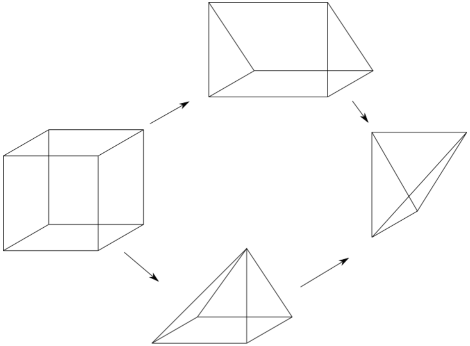


but we can not (directly) combine them.

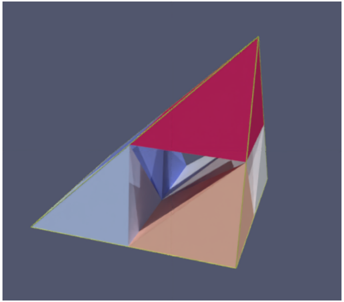
What about 3D?

Hybrid AMR

We use prisms and pyramids as binding elements.

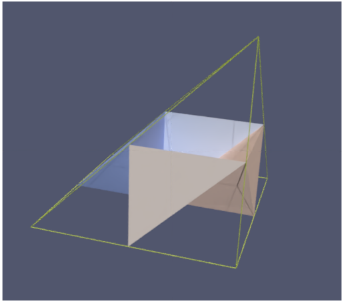


How to refine a pyramid



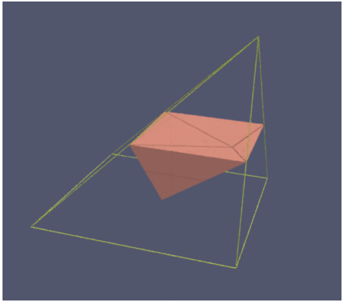
A pyramid refines into 5 pyramids (the corners), ...

How to refine a pyramid



A pyramid refines into 5 pyramids (the corners), 4 tetrahedra (gaps) ...

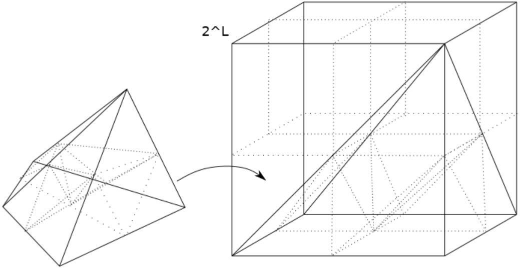
How to refine a pyramid



A pyramid refines into 5 pyramids (the corners), 4 tetrahedra (gaps) and another pyramid in the center.

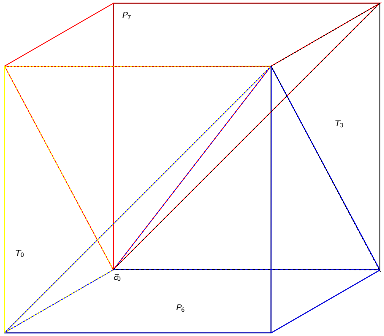
The reference pyramid

Map every pyramid of the coarse mesh onto a reference pyramid



- Refining the pyramid, we refine the cube implicitly
- Every child of the pyramid lays in a child of the cube

Types of pyramids



Alltogether, we have 8 types of pyramids

The pyramid-index

We can identify a pyramid via the anchor-node, the type and level of an element

The pyramid-index

Identification

We can identify a pyramid via the anchor-node, the type and level.

Pyramid-index

The **pyramid-index** of a pyramid $P \in \mathcal{P}$ is given as the interleaving of the $\mathcal{L}$-tuples, Z, Y, X and B:

$$m_P(P) := Z \mathbin{\dot{\perp}} Y \mathbin{\dot{\perp}} X \mathbin{\dot{\perp}} B^2 \mathbin{\dot{\perp}} B^1 \mathbin{\dot{\perp}} B^0 \tag{1}$$

where X, Y, and Z are the binary representation of the x-, y- and z-coordinate of the anchor coordinate. B^0, B^1 and B^2 encode B in binary.

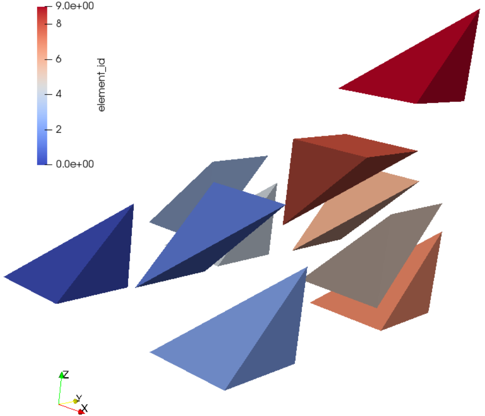
Interleaving?

Interleaving?

Interleaving!

$$x = (x_2, x_1, x_0)$$
$$y = (y_2, y_1, y_0)$$
$$z = (z_2, z_1, z_0)$$
$$b = (b_2, b_1, b_0)$$
$$z \dot{\perp} y \dot{\perp} x \dot{\perp} b = (z_2, y_2, x_2, b_2, z_1, y_1, x_1, b_1, z_0, y_0, x_0, b_0)$$

The pyramidal SFC



Shape of an element

Problem

For High-Level algorithms, all elements in pyramidal refinement are pyramids

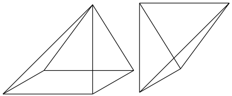
Shape of an element

Problem

For High-Level algorithms, all elements in pyramidal refinement are pyramids

Solution

The shape of an element



Two elements of the class pyramid, one in the shape of a pyramid, the other in the shape of a tetrahedron.

Example: The parent

Algorithm: `t8_dpyramid_parent`

if *Shape(P)=Pyramid* **then**
    Shift coordinates and compute type of parent;
else
    if *type(P) neither 0 nor 3* **then**
        `t8_dtet_parent(P)`
    else
        if *P inside Tet* **then**
            `t8_dtet_parent(P)`
        else
            Shift coordinates and compute type of parent;
        end
    end
end

Changes in High-level Algorithms

Old version `New`

- Compute first element i
- Compute tree of i via $\left\lfloor \frac{i}{8^l} \right\rfloor$
- Compute successor of i until last element is computed

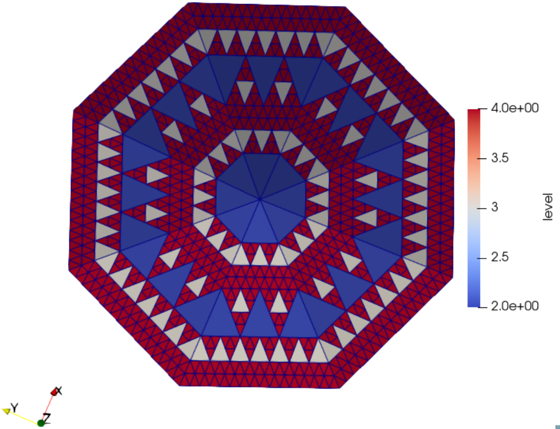
Adaptation of `New`

- Iterate over levels:
  - Refine one level
  - Partition

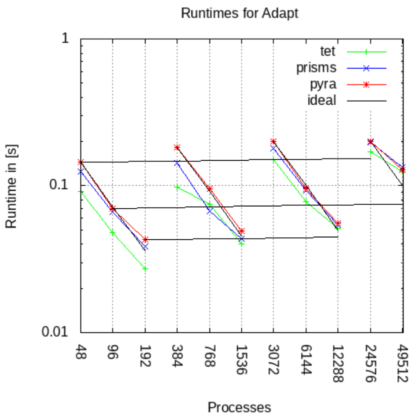
Outlook

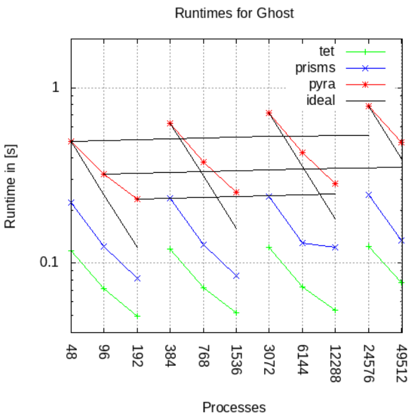
- Direct computation of the ranges of each process
- Computation independent of the level

Experiment to compare performance of elements in `t8code`



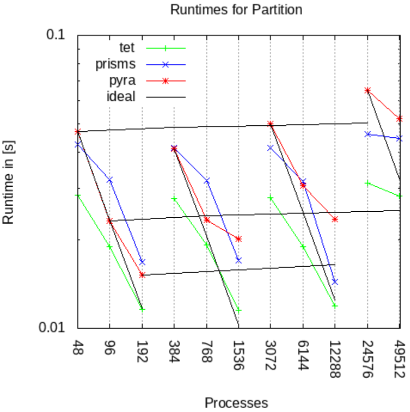
Runtimes of `Adapt` and `Ghost`

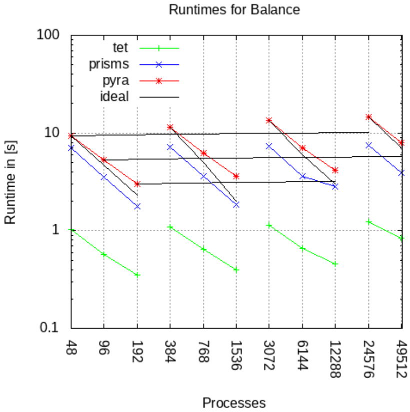




Up to $2e6$ elements per process and up to $5.1e10$ in total. Computation were done on the Jewels Supercomputer.

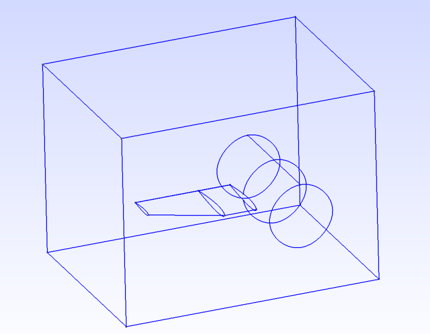
Runtimes of `Partition` and `Balance`

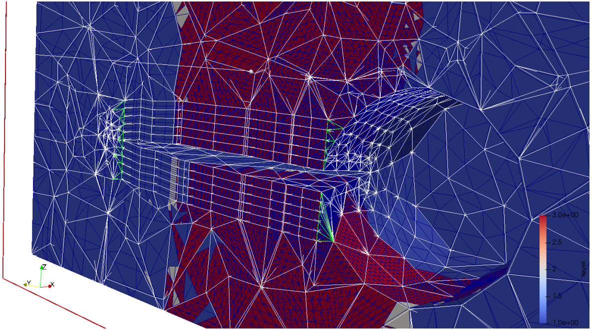




Up to 2*e*6 elements per process and up to 5.1*e*10 in total. Computation were done on the Jewels Supercomputer.

Hybrid Mesh

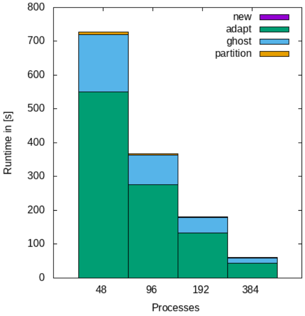




A "plane" that is approximated by the recommendations in "Mesh Generation for the NASA High lift Common Research Model" by C.D. Woeber et al. There are 69,431 tetrahedra, 3,800 hexahedra, 29,520 prisms and 3,120 pyramids in the coarse mesh.

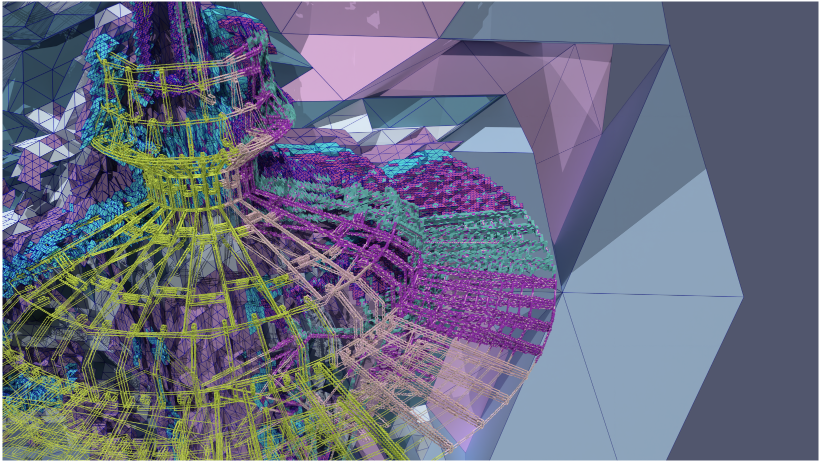
Hybrid Mesh

Example: Moving Wall



Summed over 14 iterations. Up to 1.1e10 elements arise.

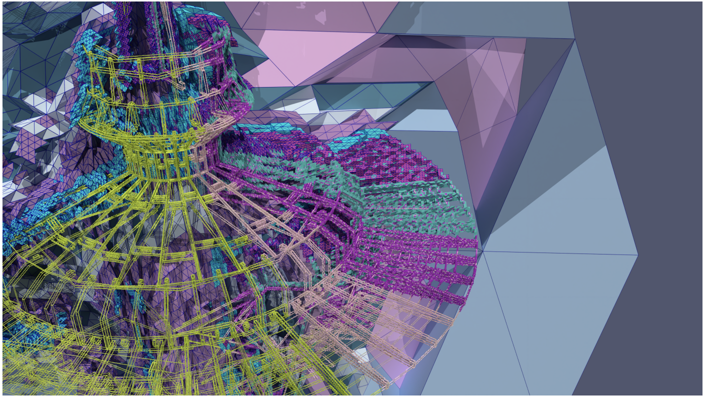
Experiment



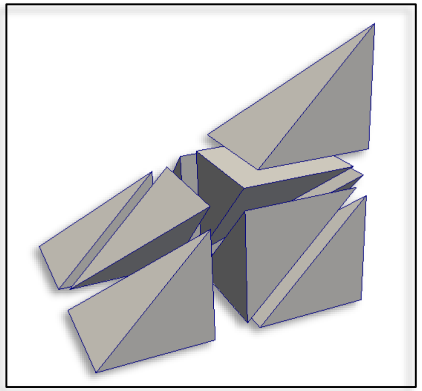
Experiment

Level 19

- 111.965.464.003 elements
- 5.480.470 pyramids
- 6144 Processes
- 1.8 million elements per Process
- 14.3 seconds in total

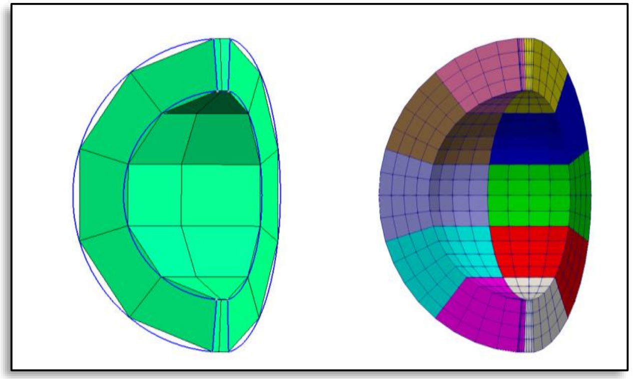


t8code @ SIAMPP22



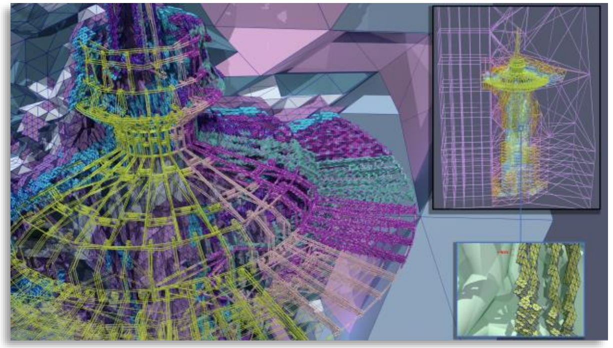
MS28 Thursday, 24th, 11:45 UTC-8 / 19:45 GMT

A space-filling curve for pyramids
David Knapp, J. Holke, C. Burstedde



IMR22, Thursday, 24th, 17:10 GMT / 09:10 UTC-8

Constructing a Volume Geometry Map For Hexahedra With Curved Boundary Geometries
Johannes Holke, S. Elsweijer, J. Kleinert, D. Reith



Meshing contest @IMR22

More about `t8code`, AMR and SFC

Code: https://github.com/holke/t8code

Article: An Optimized, Parallel Computation of the Ghost Layer for Adaptive Hybrid Forest Meshes, Submitted to SIAM Journal on Scientific Computing, Johannes Holke and David Knapp and Carsten Burstedde

Thesis: A space-filling curve for pyramidal adaptive mesh refinement, Master thesis at University of Bonn, David Knapp

Article: A Tetrahedral Space-Filling Curve for Nonconforming Adaptive Meshes, SIAM Journal on Scientific Computing, Carsten Burstedde and Johannes Holke

PhD Scalable algorithms for parallel tree-based adaptive mesh refinement with general element types, PhD thesis at University of Bonn, Johannes Holke

Thesis: The local discontinuous galerkin method for the advection-diffusion equation on adaptive meshes, Master thesis at University of Bonn, Lukas Dreyer

Code: https://github.com/lukasdreyer/t8dg

and more