AMR in Peano

A Software Framework for PDE Solvers on Spacetree Grids

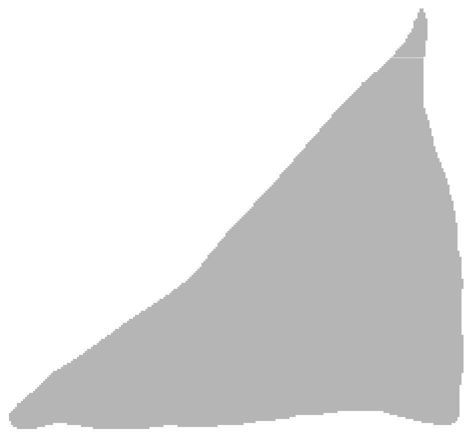
Tobias Weinzierl

[HPC]^3

February 5, 2012

The *k*-Spacetree – a Generalised Quadtree/Octree

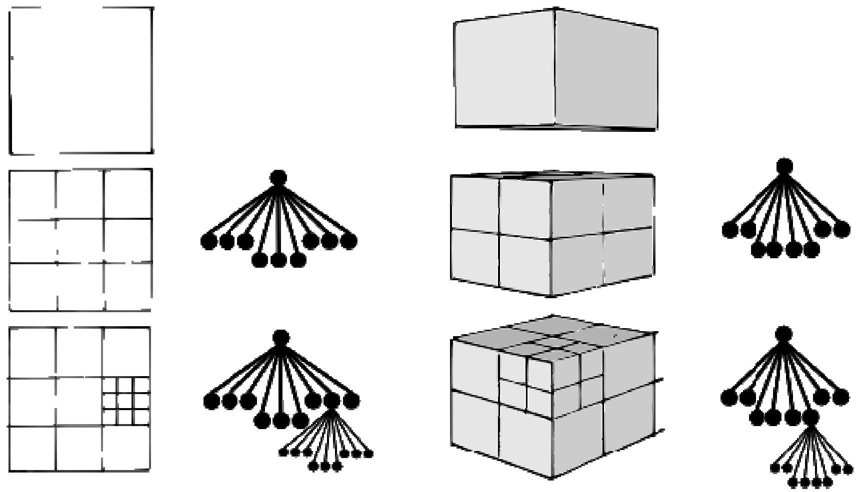
- *k*-Spacetree definition
- *k*-Spacetree properties
- *k*-Spacetree vertex management approaches



k-Spacetrees Have Many Nice Properties

- *k*-Spacetree definition

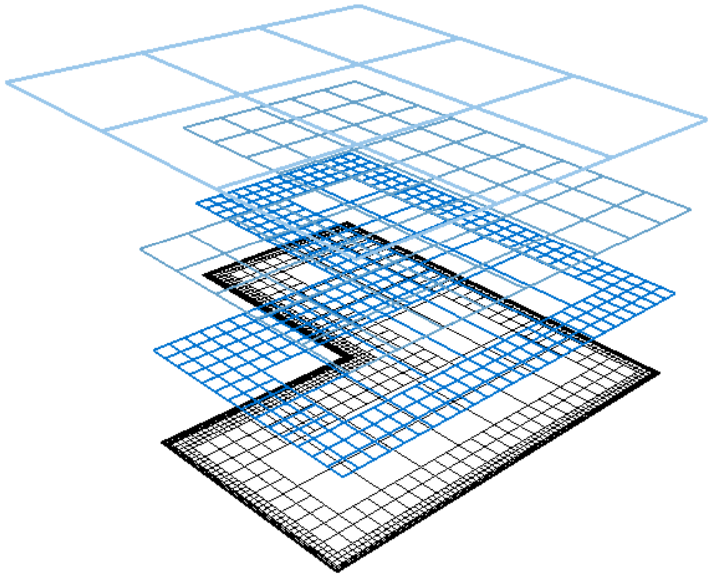
- *k*-Spacetree properties
  - Store tree instead of Cartesian grid
  - *d*-dimensional
  - In-situ mesh generation at low setup cost
  - Low memory requirements (tree data structure)
  - Dynamic adaptivity
  - Mathematical simplicity
  - Multiscale representation of domain
  - Spatial decomposition
  - …



- *k*-Spacetree vertex management approaches

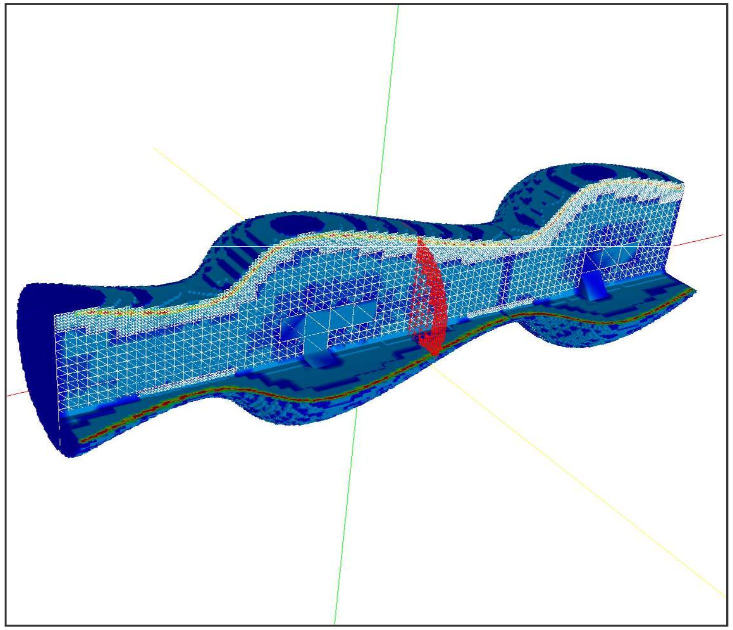
How do I Store Connectivity Information?

- *k*-Spacetree definition
- *k*-Spacetree properties
- *k*-Spacetree vertex management approaches
  - How does traversal look like?
  - Shall we store the finest grid only or the whole tree?
  - Store pointers to unstructured mesh? (memory footprint)
  - Map spacetree to vertex table? (uniquely decodable code)
  - Another sophisticated approach?



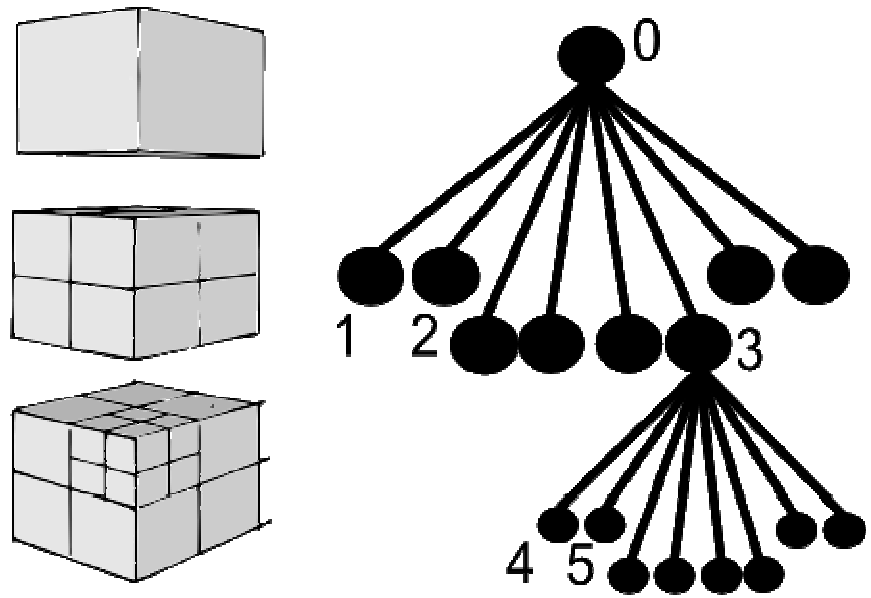
Outline

- Overview & motivation
- Peano – a tree traversal and grid management paradigm
  - Depth-first tree traversal
  - Deterministic traversal due to the Peano space-filling curve
  - A stack-based vertex management
  - Algorithmic properties
- ~~Hybrid parallelisation~~
- Peano – a PDE framework
  - Software architecture
  - Software engineering
  - Programming paradigm
- Wrap-up



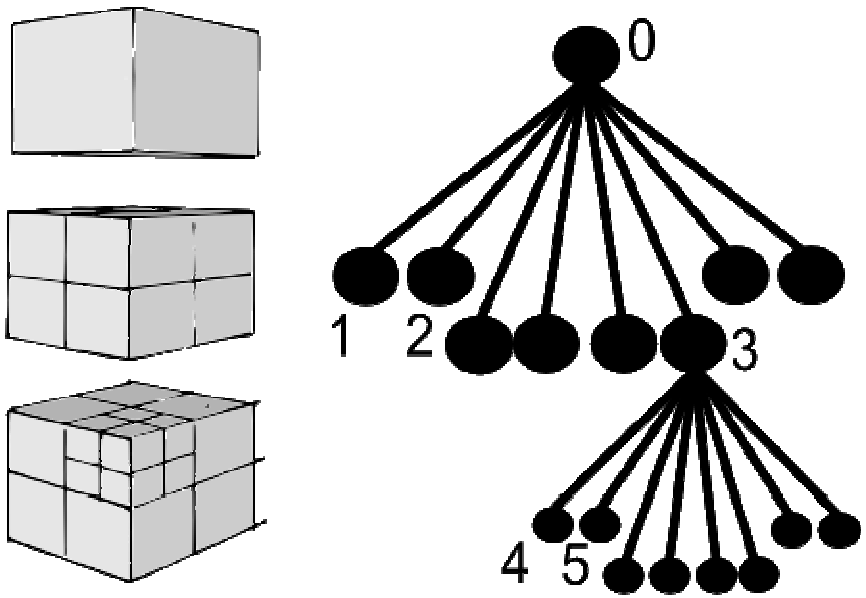
Tree Traversal Yields Element-wise Grid Traversal

- *k*-Spacetree traversals
  - Element-wise traversal is a standard concept for FEM
  - Tree travesal corresponds to element-wise grid traversal
  - We can embed operations of (matrix-free) solver(s) into traversal; traversal properties hold for solver, too
- Depth-first paradigm
- Deterministic order of children
- Temporary vertex containers



Depth-first Tree Traversal Yields Simple Encoding

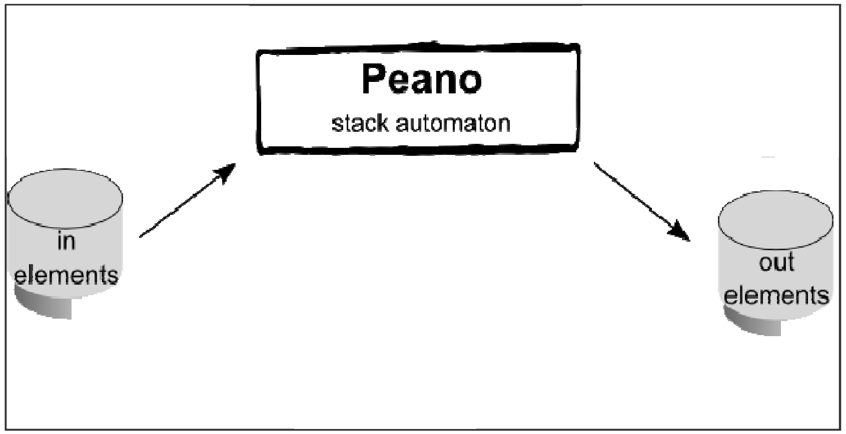
- *k*-Spacetree traversals
- Depth-first paradigm
  - Depth-first tree traversal facilitates recursive formulation with traversal automaton
  - Tree traversal allows simple, recursive implementation
  - Refinement information 1 bit
  - Swap traversal direction on each level: Cell containers are stacks
- Deterministic order of children
- Temporary vertex containers



Automaton Processes Cell Stream

- *k*-Spacetree traversals

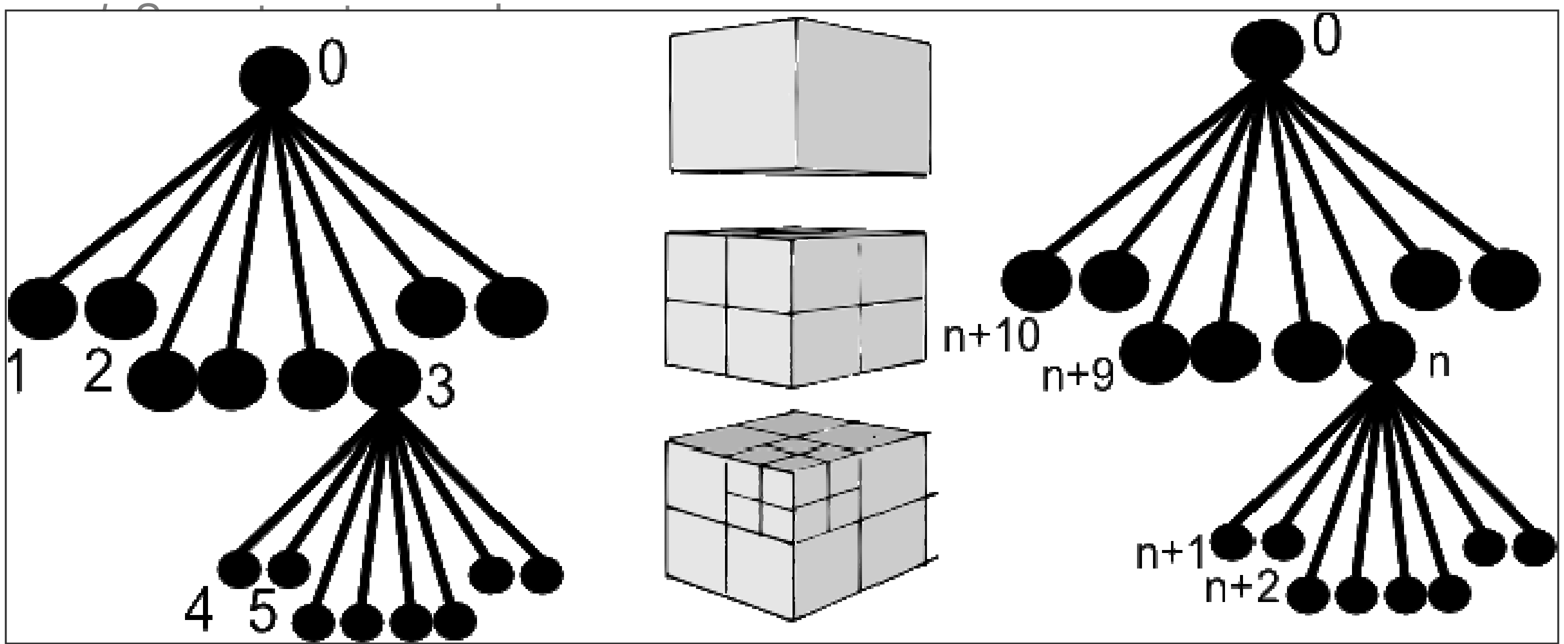
- Depth-first paradigm
  - Depth-first tree traversal facilitates recursive formulation with traversal automaton
  - Tree traversal allows simple, recursive implementation
  - Refinement information 1 bit
  - Swap traversal direction on each level: Cell containers are stacks



- Deterministic order of children

- Temporary vertex containers

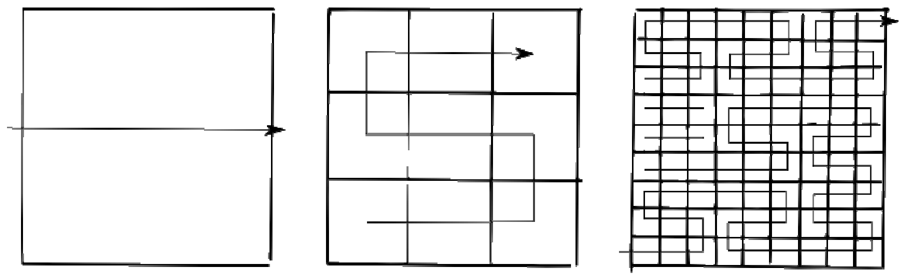
Input/Output Container for Cells are Stacks



- Deterministic order of children
- Temporary vertex containers

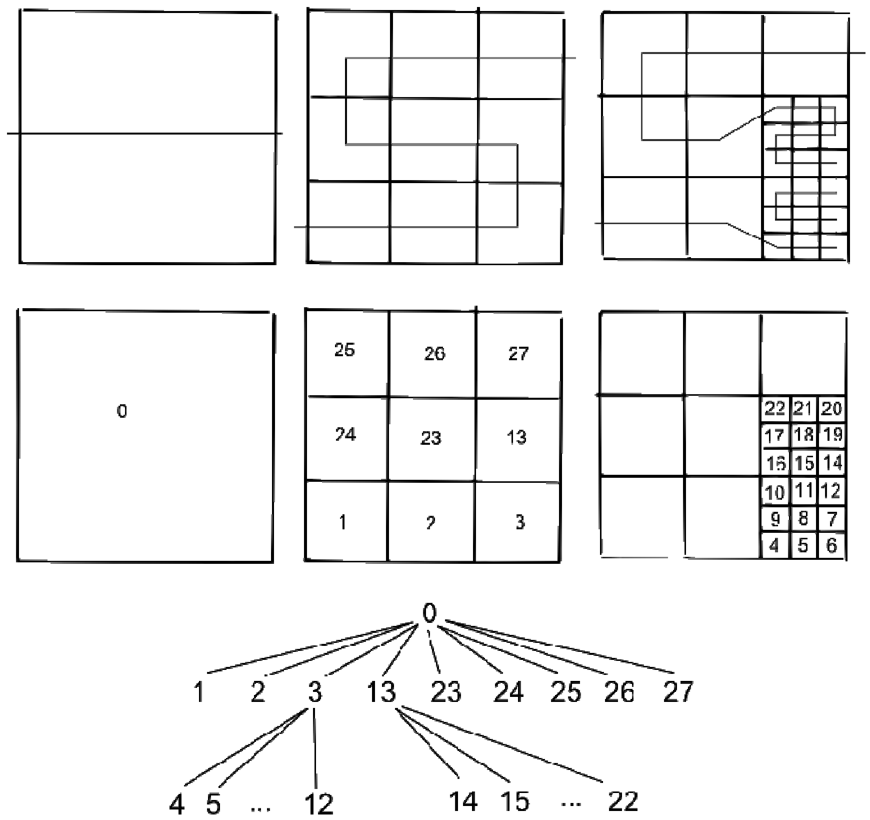
Peano Space-filling Curve Resembles *(k=3)*-spacetree

- *k*-Spacetree traversals
- Depth-first paradigm
- Deterministic order of children
  - Use iterate of Peano space-filling curve
  - Traversal Automaton
  - Swap traversal direction: in- and output stream are stacks
- Temporary vertex containers



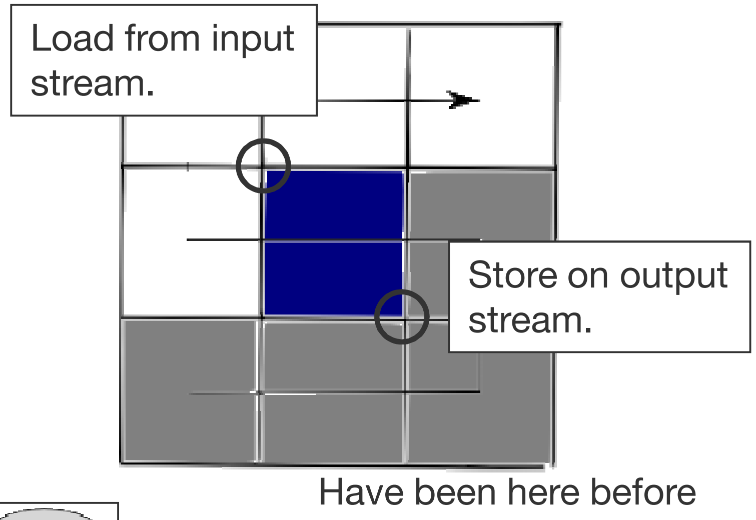
Space-filling Curve Defines Unique Node Order

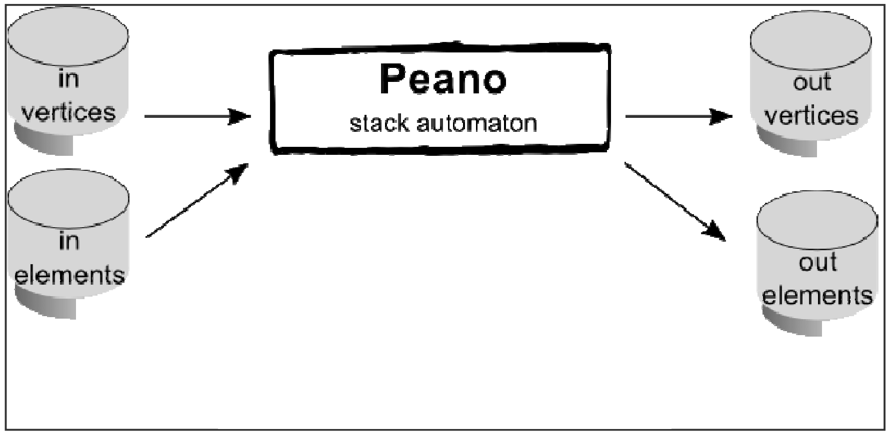
- *k*-Spacetree traversals
- Depth-first paradigm
- Deterministic order of children
  - Use iterate of Peano space-filling curve
  - Traversal Automaton
  - Swap traversal direction: in- and output stream are stacks
- Temporary vertex containers



Input/Output Containers for Vertices are Stacks

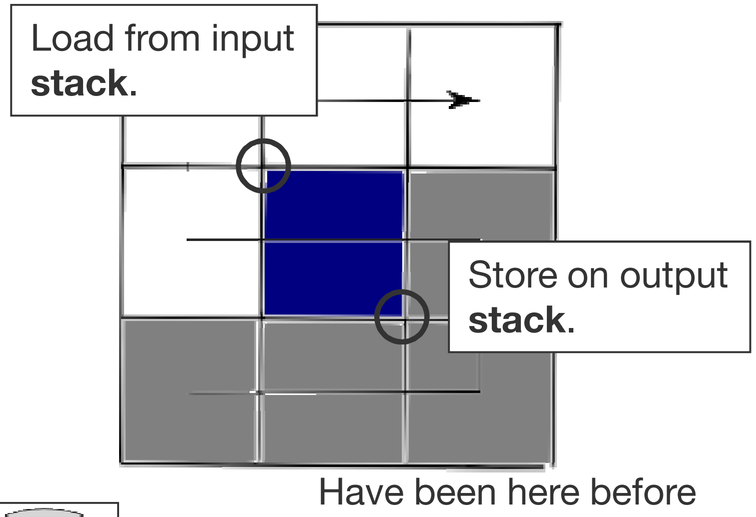
- *k*-Spacetree traversals
- Depth-first paradigm
- Deterministic order of children
  - Use iterate of Peano space-filling curve
  - Traversal Automaton
  - Swap traversal direction: in- and output stream are stacks

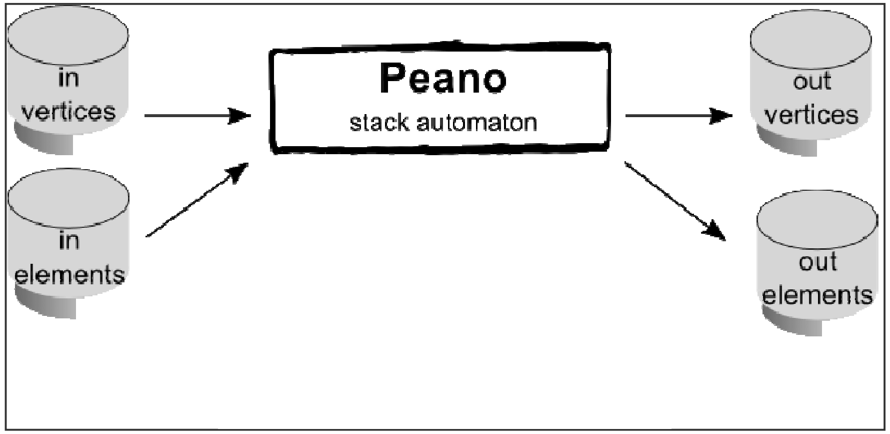




Input/Output Containers for Vertices are Stacks

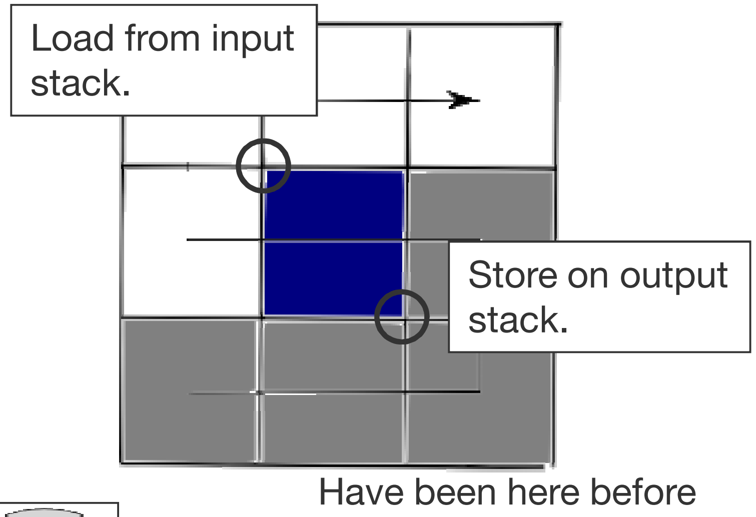
- *k*-Spacetree traversals
- Depth-first paradigm
- Deterministic order of children
  - Use iterate of Peano space-filling curve
  - Traversal Automaton
  - Swap traversal direction: in- and output stream are stacks

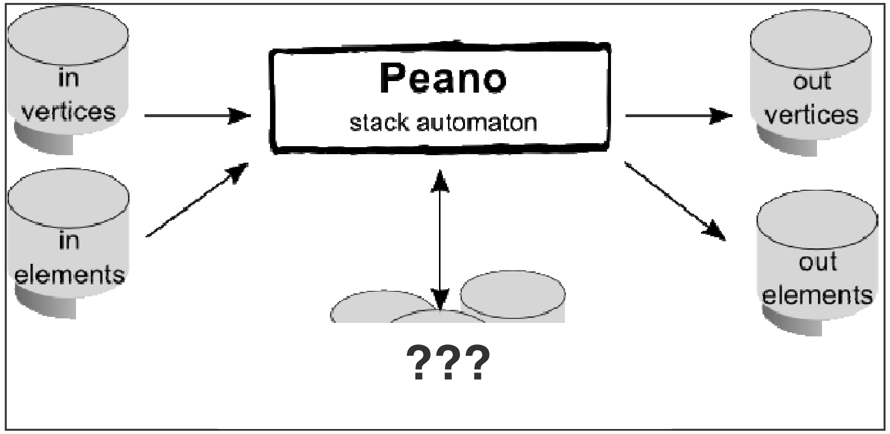




Input/Output Containers for Vertices are Stacks

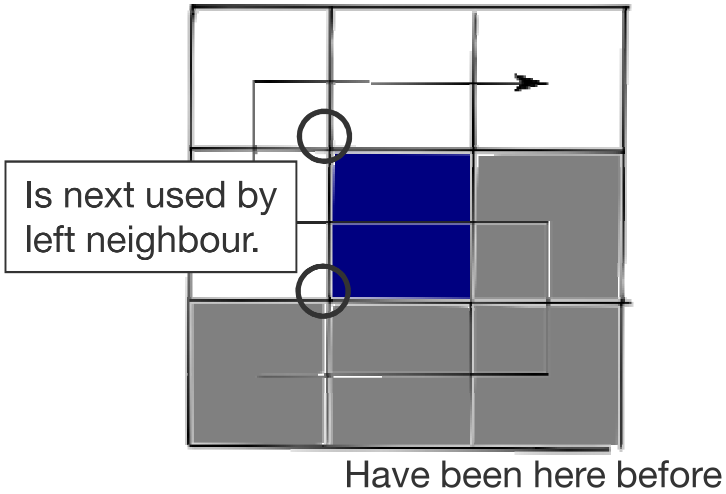
- *k*-Spacetree traversals
- Depth-first paradigm
- Deterministic order of children
  - Use iterate of Peano space-filling curve
  - Traversal Automaton
  - Swap traversal direction: in- and output stream are stacks





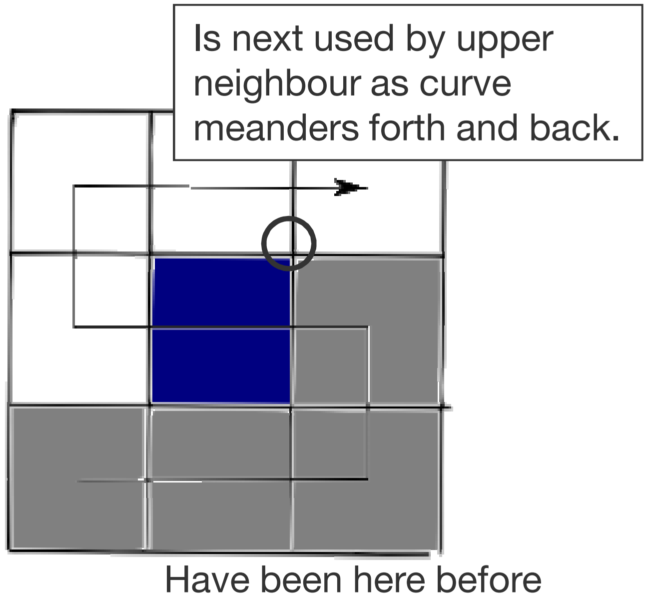
Automaton Knows Upcoming Access Scheme

- *k*-Spacetree traversals
- Depth-first paradigm
- Deterministic order of children
- Temporary vertex containers
  - Curve is continuous
  - Palindrome property
  - Projection property
  - Temporary vertex containers are stacks

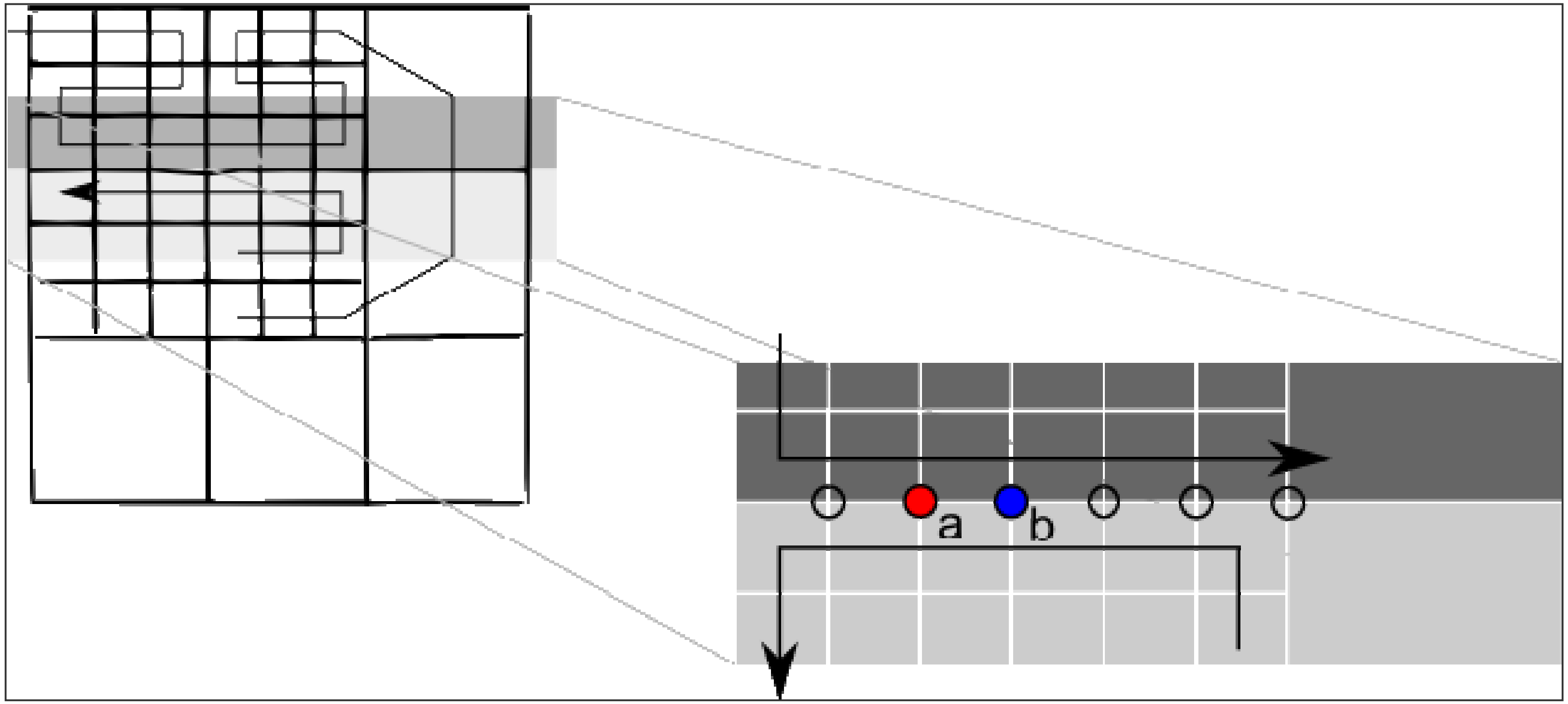


Automaton Knows Upcoming Access Scheme

- *k*-Spacetree traversals
- Depth-first paradigm
- Deterministic order of children
- Temporary vertex containers
  - Curve is continuous
  - Palindrome property
  - Projection property
  - Temporary vertex containers are stacks

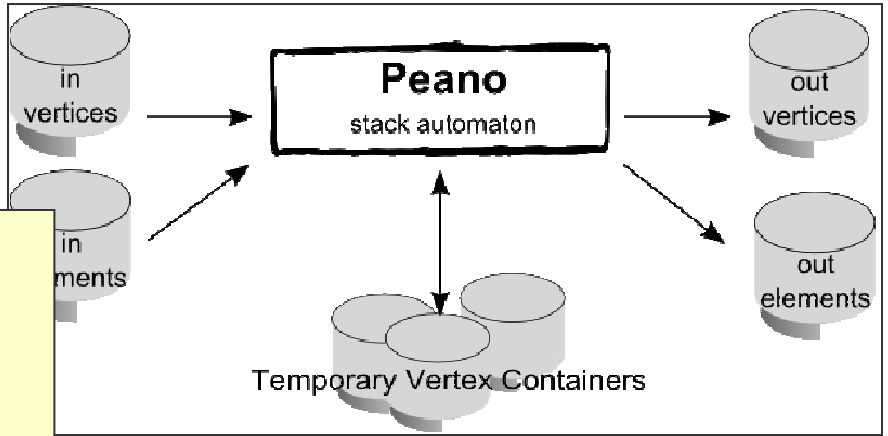


Continuous Curve Meanders Forth and Back



Temporary Vertex Containers are Stacks, too

- *k*-Spacetree traversals
- Depth-first paradigm
- Deterministic order of children



Realisation Properties

- Element containers: 2 stacks
- Vertex in/out containers: 2 stacks
- Temporary vertex containers: 2d stacks (former 3^d-1)
- 1 bit per vertex/cell for refinement information
- No pointers, no maps, no memory overhead

Space-filling curve idea due to Christoph Zenger

Algorithmic Properties: Memory Footprint

Memory footprint timestepping (matrix-free) for diffusion equation in bytes			
Dimension	Explicit Euler	Jacobi	BoxMG (A,P,R)
1	18	26	52
2	18	26	144
3	18	26	580

T. Weinzierl and T. Köppl: A Geometric Space-time Multigrid Algorithm for the Heat Equation.
Numerical Mathematics: Theory, Methods and Applications, Volume 5(1), p. 110–130.
Global Science Press, February 2012.

I. Yavneh and M. Weinzierl: Nonsymmetric Black Box multigrid with coarsening by three.
In Numerical Linear Algebra with Applications, Special Issue Copper Mountain 2011, 2012.
accepted.

Algorithmic Properties: Size of Vertex Stacks

Maximum elements per stack for different resolutions ($d = 3$). The upper experiments study a regular grid, the lower block shows results for an adaptive grid based upon a geometric refinement criterion.

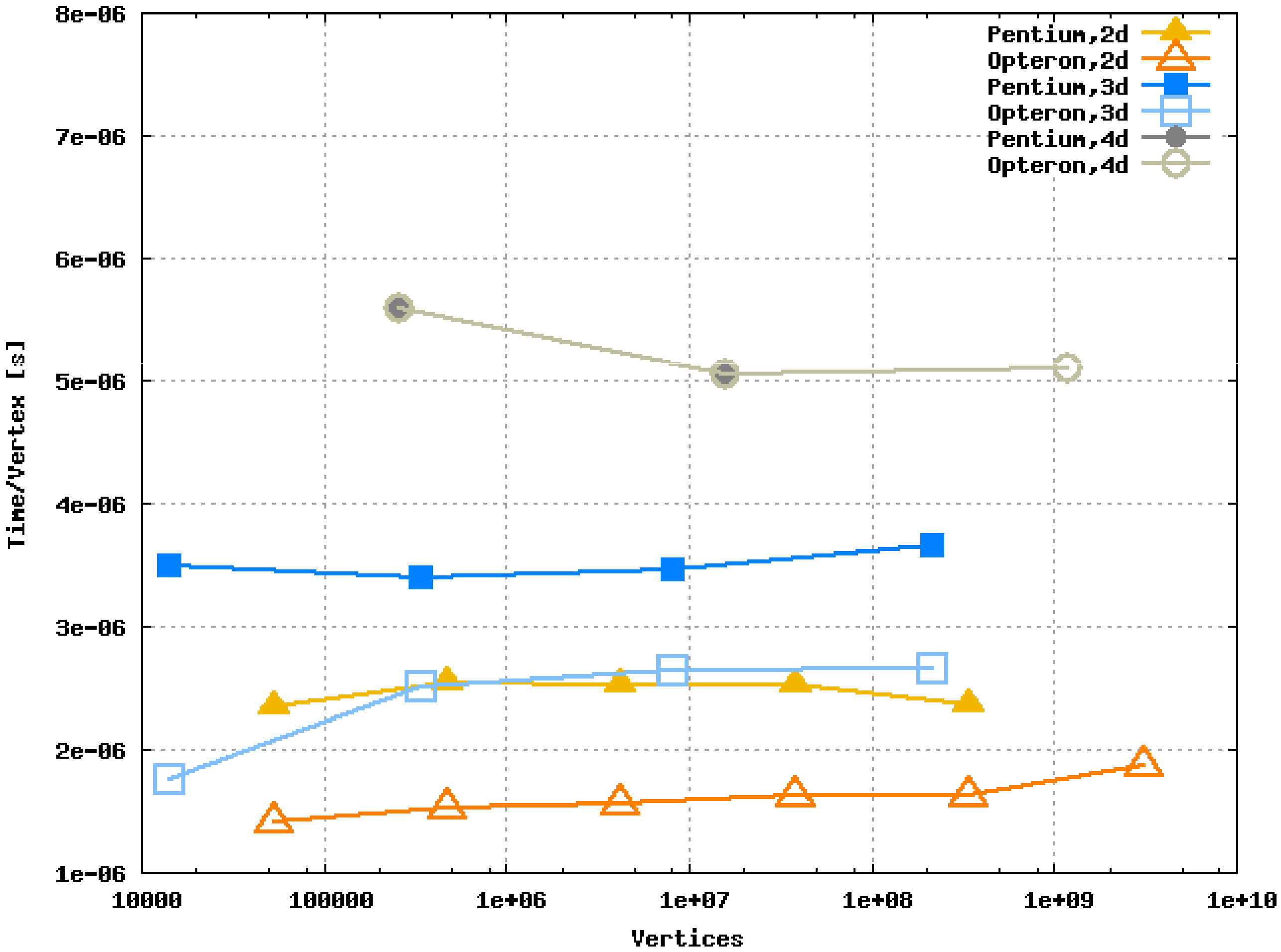
	in/out	temp 1	temp 2	temp 3		in/out	temp 1	temp 2	temp 3
Cube	8	0	0	0	Sphere	8	0	0	0
	520	15	31	63		520	15	31	63
	3,264	115	171	262		1,032	15	39	85
	36,032	899	1,249	1,361		14,288	115	299	671
	672,088	7,623	10,351	8,994		319,360	899	2,464	5,874
Cube	8	0	0	0	Sphere	8	0	0	0
	520	15	31	63		520	15	31	63
	3,200	115	171	262		1,032	15	39	85
	25,320	899	1,091	1,361		13,000	91	216	562
	222,400	7,623	8,293	8,994		150,464	392	922	2,365
	1,994,120	67,159	69,279	71,221		1,494,720	1,336	3,081	7,858

Algorithmic Properties: Cache Hit Rate

Memory access characteristics for different geometries. The data results from the Itanium's hardware counters.

		d	$\frac{\text{L2 Misses}}{\text{L2 References}}$	$\frac{\text{L3 References}}{\text{L2 Misses}}$	$\frac{\text{L3 Misses}}{\text{L3 References}}$	Bus Load
Square	regular	2	0.03722%	1.610	43.08%	$\approx 0\%$
	adaptive		0.05370%	1.776	39.47%	$\approx 0\%$
Circle	regular		0.03243%	1.567	40.35%	$\approx 0\%$
	adaptive		0.04029%	1.378	42.75%	$\approx 0\%$
Cube	regular	3	0.06213%	1.770	30.93%	$\approx 0\%$
	adaptive		0.06778%	1.364	43.77%	$\approx 0\%$
Sphere	regular		0.06908%	1.486	22.63%	$\approx 0\%$
	adaptive		0.05258%	1.346	38.36%	$\approx 0\%$
Hypercube	regular	4	0.07454%	1.493	14.13%	$\approx 0\%$
	adaptive		0.09130%	1.238	30.31%	14%
Hypersphere	regular		0.05671%	1.391	13.83%	$\approx 0\%$
	adaptive		0.12047%	0.880	31.03%	13%

Algorithmic Properties: Cost per Vertex



Outline

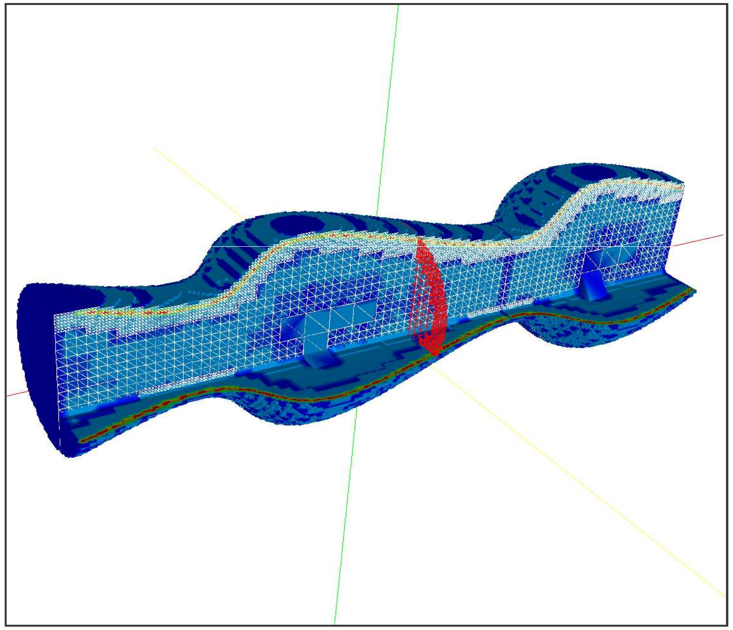
- Overview & motivation

- Peano – a tree traversal and grid management paradigm
  - Depth-first tree traversal
  - Deterministic traversal due to the Peano space-filling curve
  - A stack-based vertex management
  - Algorithmic properties

- ~~Hybrid parallelisation~~

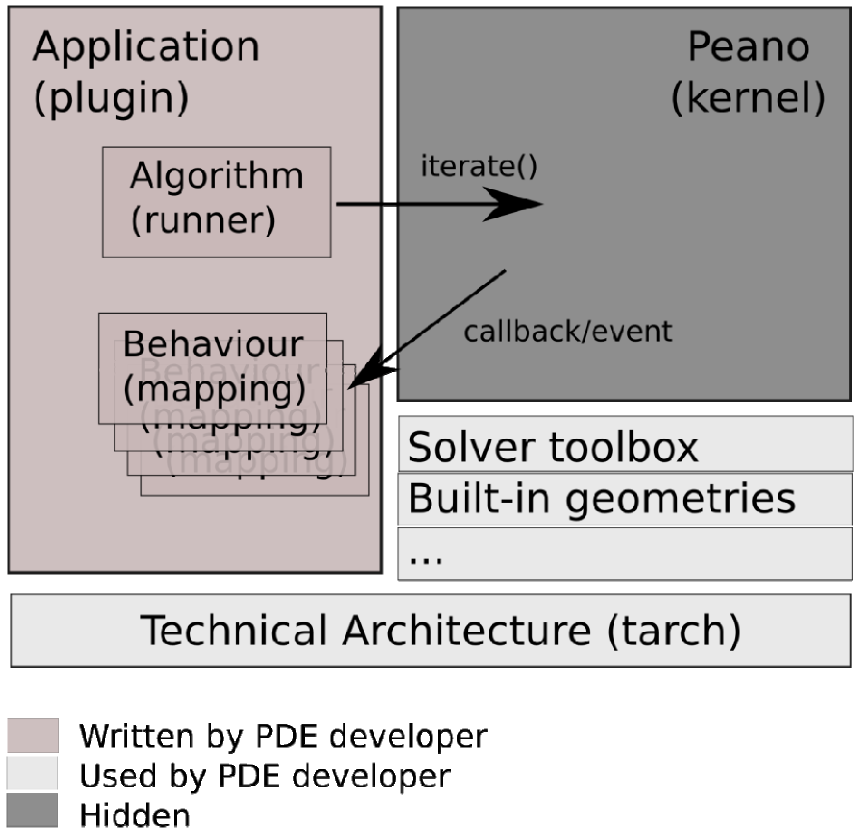
- Peano – a PDE framework
  - Software architecture
  - Software engineering
  - Programming paradigm

- Wrap-up



Peano‘s Four Building Blocks

- Architecture
  - Component overview
  - Three interaction paradigms
- Software engineering
- Development workflow
- Peano in parallel



Interaction Paradigm: Black-Box Grid Traversal

- Architecture
  - Component overview
  - Three interaction paradigms
- Software engineering
  - Agile development (no branches, no releases)
  - Unit test concept
  - Nightly build & continuous integration
- Development workflow
- Peano in parallel

I

We decide where.

II

Don’t call us – we call you!

III

[...] you do not know the day nor the hour.

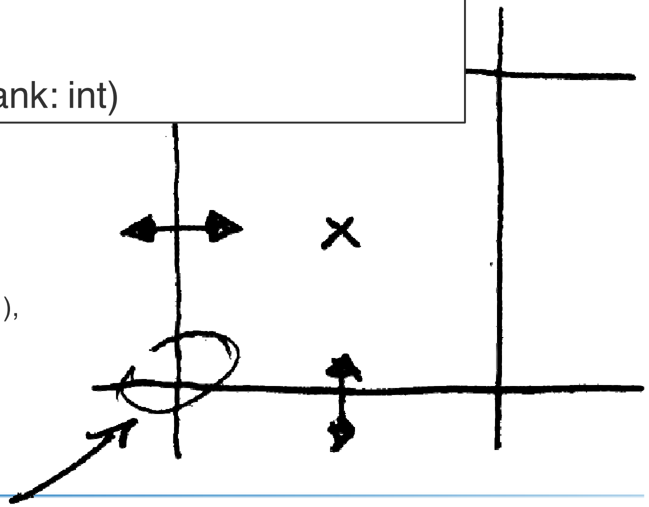
DSL: Data is Modelled With DaStGen

- Architecture
- Software engineering
- Development workflow
  - Data modelling due to DaStGen (C++ extension with data compression and MPI support)
  - Use of (rapid) prototyping tool: script language (AOP)
  - Simple event concept per mapping (phase)
  - Grid manipulation
- Peano in parallel

MyVertex

- Enum State = {
  Inner, Dirichlet, Neumann
  }
- **packed** state: State
- value: double [2]
- **packed parallelise** isInitialised: bool
- **discard** vertexNumber: long int
- **parallelise** residual: double

+ get…()
+ set…()
+ send(toRank: int)



H.-J. Bungartz, W. Eckhardt, T. Weinzierl and C. Zenger: A Precompiler to Reduce the Memory Footprint of Multiscale PDE Solvers in C++. Future Generation Computer Systems, Volume 26(1), p. 175–182. Elsevier, January 2010.

H.-J. Bungartz, M. Mehl, T. Weinzierl and W. Eckhardt: DaStGen - A Data Structure Generator for Parallel C++ HPC Software. Volume 5103 of Lecture Notes in Computer Science, p. 213–222. Springer-Verlag, Heidelberg, Berlin, June 2008.

Algorithm is Modelled as Sequence of Phases

- Architecture
- Software engineering
- Development workflow
  - Data modelling due to DaStGen (C++ extension with data compression and MPI support)
  - Use of (rapid) prototyping tool: script language (AOP)
  - Simple event concept per mapping (phase)
  - Grid manipulation
- Peano in parallel

Algorithm

Compute F and G from $\mathbf{u}_{old}$

While (res>ϵ_{res})

update p

While (du>ϵ_{u})

update $\mathbf{u}_{new}$

Switch to next time step:

$\mathbf{u}_{old} \leftarrow \mathbf{u}_{new}$

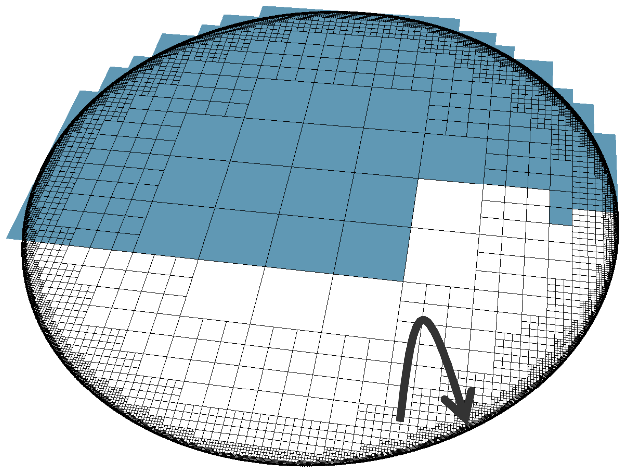
If (do plot)

plot $\mathbf{u}_{old}$

Automaton Transitions Trigger Events

- Architecture
- Software engineering

g



An event comprises

• automaton transition

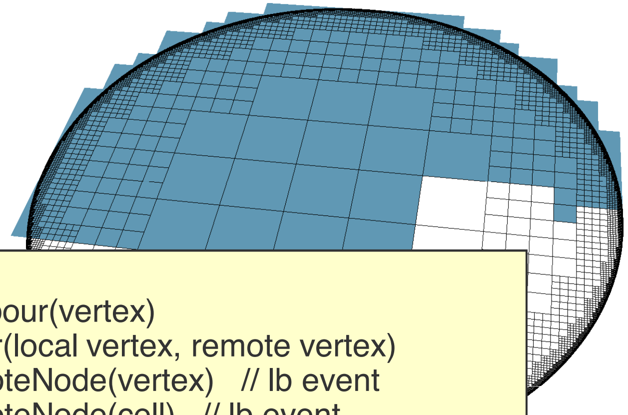
• data assigned to entity (cell/vertex)

1. There is a set of events defined by one interface.
2. There is one interface implementation per algorithm phase.
3. There is a runner (on rank 0) selecting one phase; the grid then calls respective event.

Multiple events (mappings) can be combined.
grid transitions (refinement/coarsening) are triggered by PDE software.

Joint work with Kristof Unterweger

Event/Mapping Signature



- Architecture

```
createInnerVertex(vertex, x, h,...)
createBoundaryVertex(vertex, x, h,...)
createHangingVertex(vertex, parent vertices, ....)
destroyVertex(...)
destroyHangingVertex(...)
createCell(cell, x, h, adjacent vertices, ...)
destroyCell(...)
```

(C++ extension with data compression and MPI sup

- Use of (rapid) prototyping tool: script language (AOP)
- Simple event concept per (phase)
- Grid manipulation

```
#ifdef Parallel
preareSendToNeighbour(vertex)
mergeWithNeighbour(local vertex, remote vertex)
prepareCopyToRemoteNode(vertex)   // lb event
prepareCopyToRemoteNode(cell)   // lb event
mergeWithRemoteDataDueToForkOrJoin(...)
prepareSendToMaster(...)
mergeWithMaster(...)
mergeWithWorker(...)
#endif
```

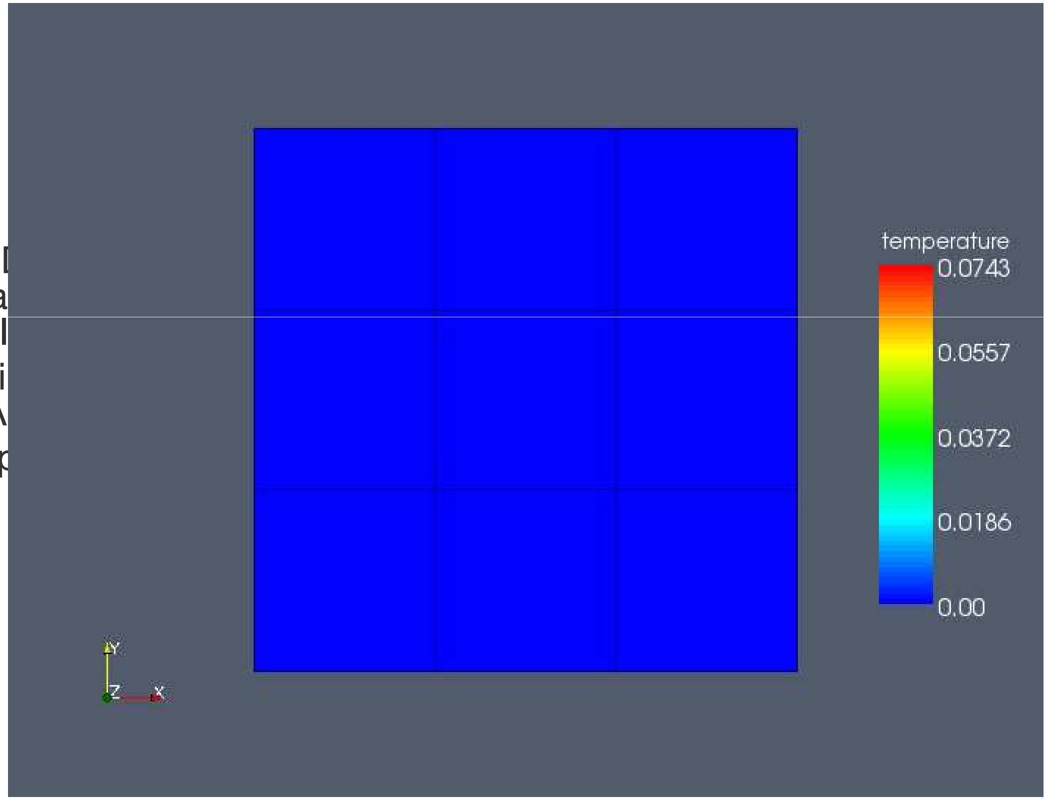
```
touchVertexFirstTime(vertex, x, h, coarser vertices, ...)
touchVertexLastTime(vertex, x, h, coarser vertices, ...)
beginIteration(state)
endIteration(state)
enterCell(cell, adjacent vertices, coarser cell, ...)
leaveCell(...)
```

1. There is a set of events defined by one interface.
2. There is one interface implementation per algorithm phase.
3. There is a runner (on rank 0) selecting one phase; the grid then calls respective event.

Multiple events (mappings) can be combined.
grid transitions (refinement/coarsening) are triggered by PDE software.

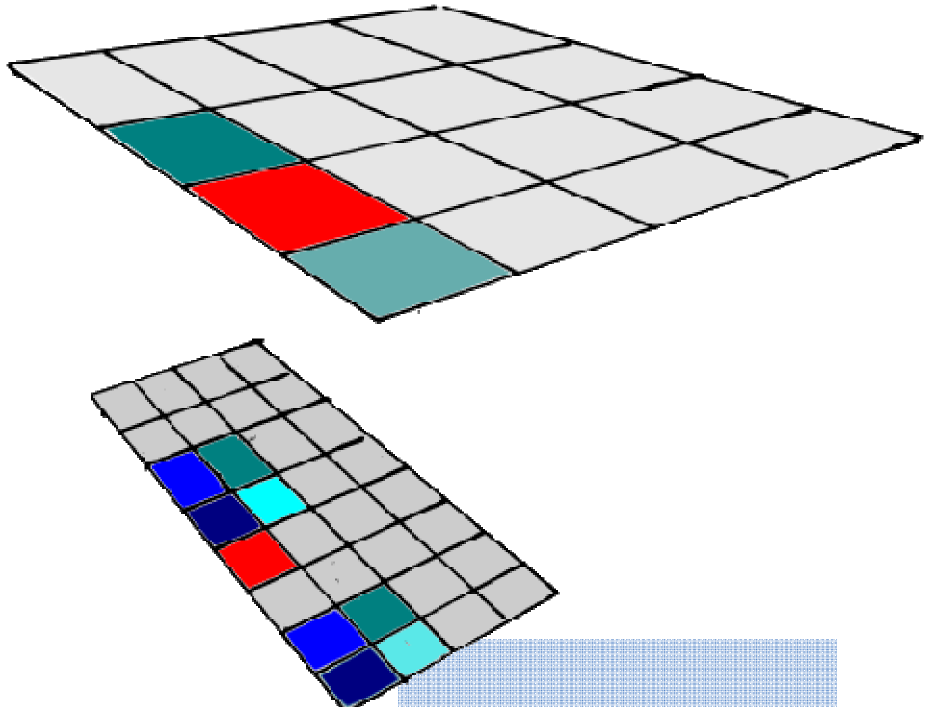
Dynamic Grid Adaption in Peano: Trigger Refinement

- Architecture
- Software engineering
- Development workflow
  - Data modelling due to I
    (C++ extension with da
    compression and MPI
  - Use of (rapid) prototypi
    tool: script language (A
  - Simple event concept p
    (phase)
  - Grid manipulation
- Peano in parallel



Shared Memory Parallelisation is Hidden

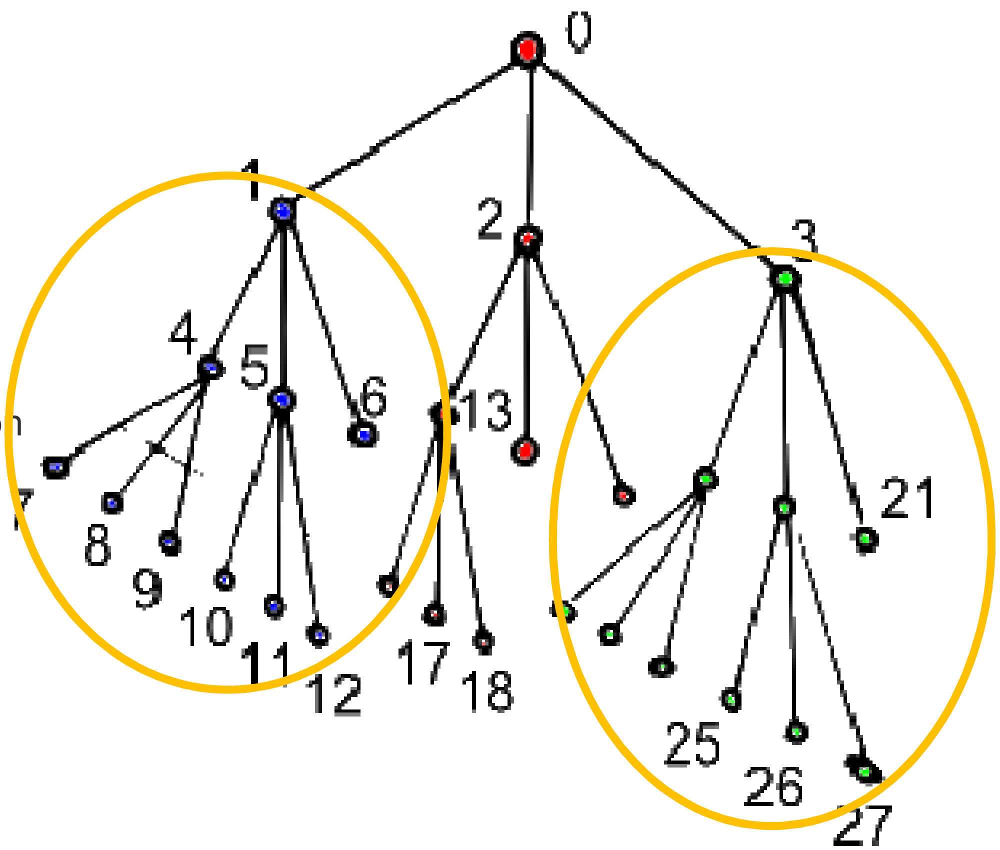
- Architecture
- Software engineering
- Development workflow
- Peano in parallel
  - The runner concept (global master)
  - Shared memory parallelisation
  - Distributed memory parallelisation



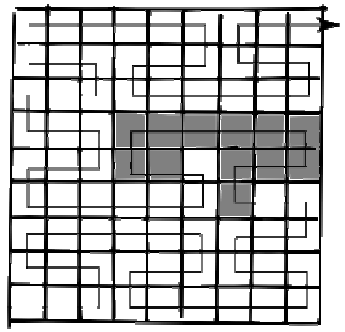
Joint work with Svetlana Nogina and Kristof Unterweger

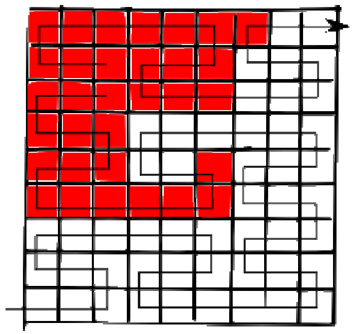
A Global State is Distributed Per Traversal

- Architecture
- Software engineering
- Development workflow
- Peano in parallel
  - The runner concept (global master)
  - Shared memory parallelisation
  - Distributed memory parallelisation

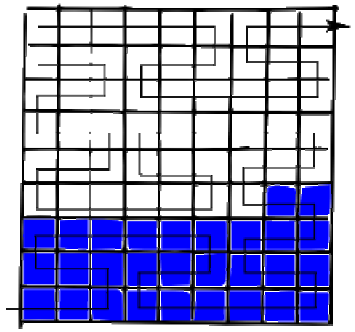


Domain Decomposition is Hidden





- Architecture
- Software engineering
- Development workflow
- Peano in parallel
  - The runner concept (global master)
  - Shared memory parallelisation
  - Distributed memory parallelisation



Based on prototypes of Markus Langlotz and Christoph Zenger

H.-J. Bungartz, M. Mehl and T. Weinzierl: A Parallel Adaptive Cartesian PDE Solver Using Space--Filling Curves. Volume 4128 of LNCS, p. 1064–1074. Springer-Verlag, Berlin, Heidelberg, 2006.

The ``Peano´´ Paradigms Revisited

I

We decide where.

II

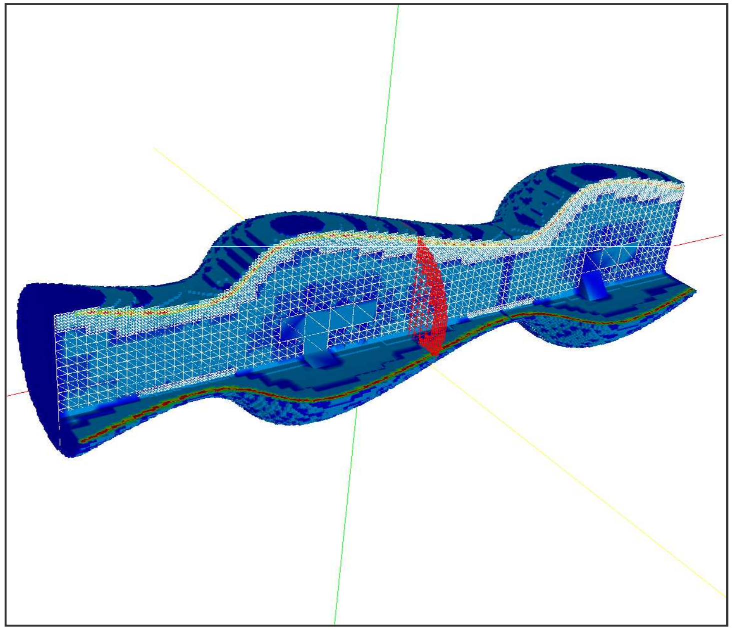
Don’t call us – we call you!

III

[...] you do not know the
day nor the hour.

Outline

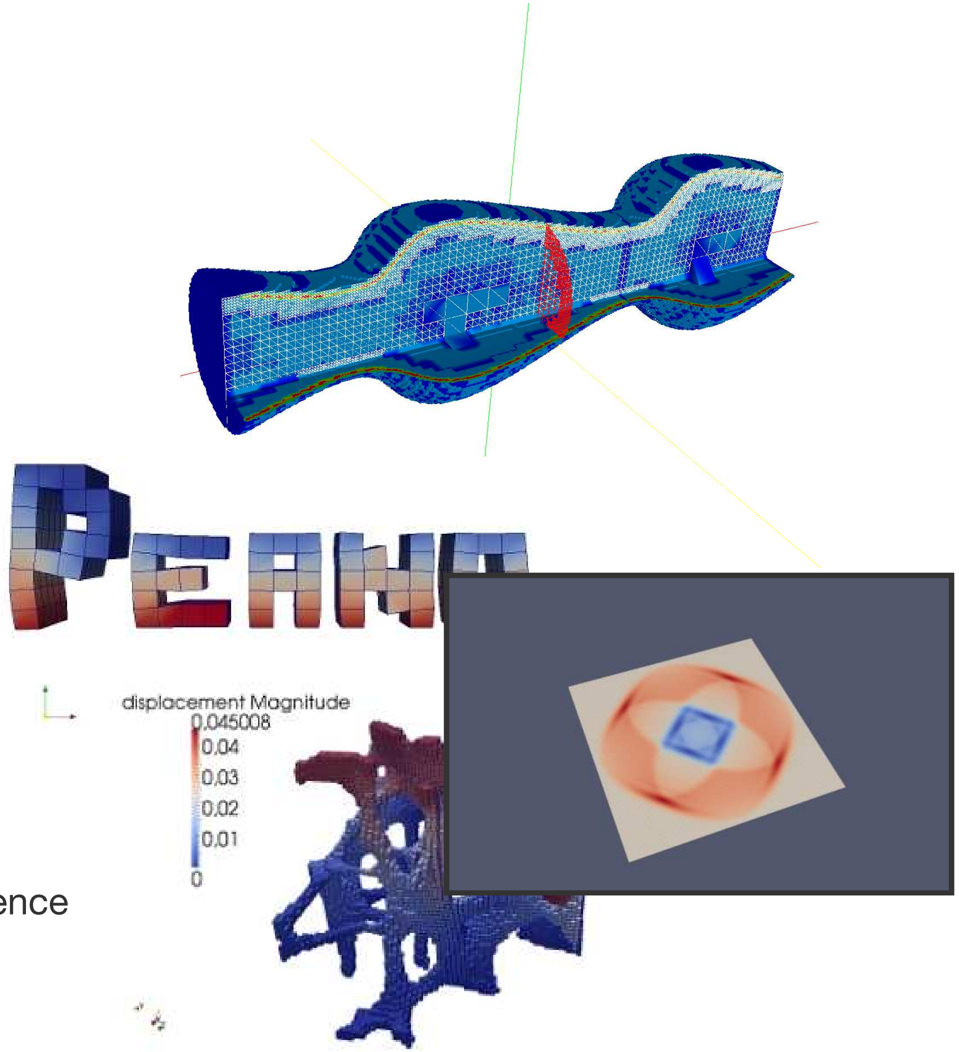
- Overview & motivation
- Peano – a tree traversal and grid management paradigm
  - Depth-first tree traversal
  - Deterministic traversal due to the Peano space-filling curve
  - A stack-based vertex management
  - Algorithmic properties
- ~~Hybrid parallelisation~~
- Peano – a PDE framework
  - Software engineering
  - Software architecture
- Wrap-up

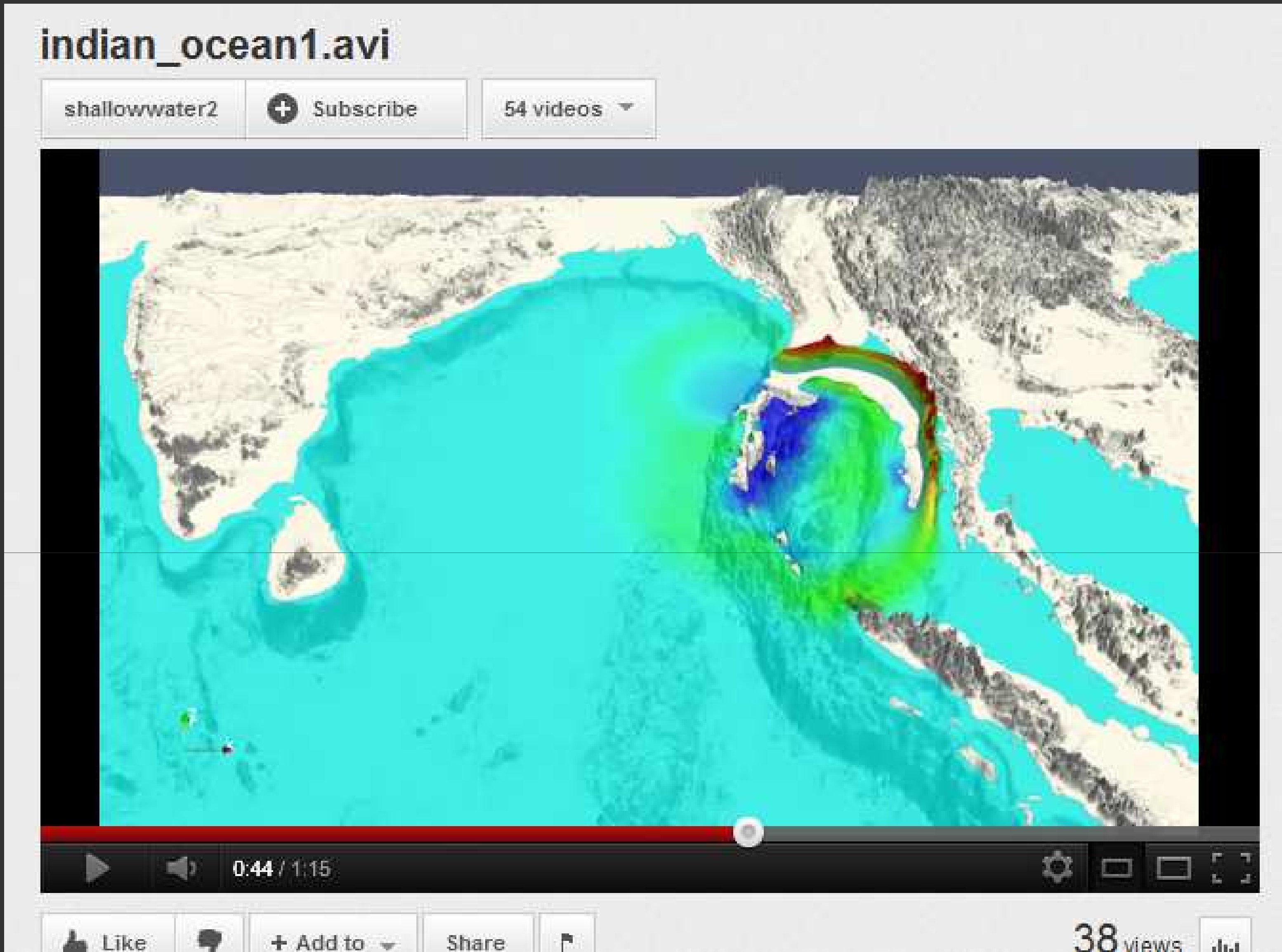


Peano – Wrap-up

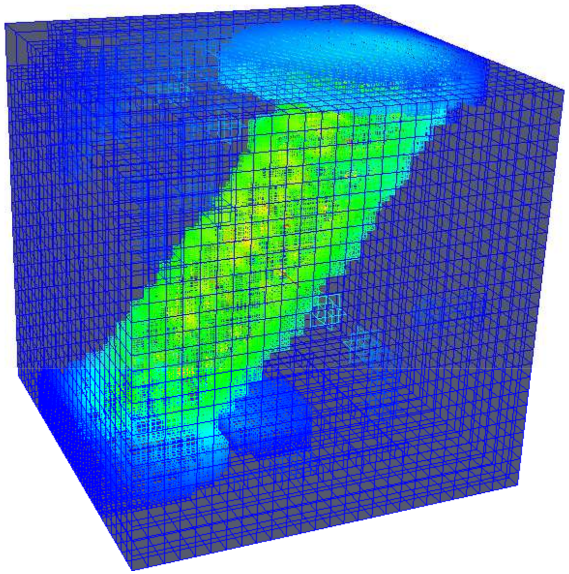
- Properties of Peano
  - Low memory requirements (tree data structure)
  - Dynamic adaptivity
  - Multiscale representation of domain
  - Simple programming model
  - Many open issues ...

- Running projects & users
  - Traditional CFD / FSI / LB (TUM, Computer Science)
  - Darcy + NS coupling (TUM together with Prof. Sun)
  - Room climate steering (Munich, Civil Engineering)
  - Space weather simulation (KU Leuven / Flanders Exascience Lab – Intel Labs Europe)
  - PeanoClaw
  - More applications (yours) are essential!





http://www.youtube.com/user/shallowwater2



Thanks are due to

- Hans-Joachim Bungartz,
- David Keyes,
- Christoph Zenger, and
- the whole Peano crew at TUM, as well as
- the users (in particular at Leuven).

This presentation is based on work supported by Award No. UK-C0020,
made by King Abdullah University of Science and Technology (KAUST).

T. Weinzierl. A Framework for Parallel PDE Solvers on Multiscale Adaptive Cartesian Grids.
Dissertation, Verlag Dr. Hut, München, 2009

T. Weinzierl and M. Mehl. Peano – a traversal and storage scheme for octree-like adaptive
cartesian multiscale grids. SIAM Journal on Scientific Computing, 33(5):2732–2760, 2011

http://www5.in.tum.de/peano