Parallel Space-time Spacetrees for Simple Parabolic Poblems

Space-time schemes for parabolic partial differential equations (PDEs) promise advantages compared to traditional time stepping: First guesses of the solution's evolution in time are delivered quickly, periodic boundary conditions are straightforward to implement, backward problems interacting with the PDE can be solved directly, ... The most prominent selling points of space-time however are the increased level of concurrency and superconvergence exploiting solution smoothness in time. All these properties render space-time a promising candidate for the dawning exascale age.

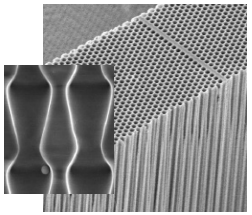
Three major showstoppers for space-time meshings do exist: memory, efficient solvers, and software complexity. We propose to pick up the concept of spacetrees, a generalisation of the well-known quadtree/octree concept for two and three dimensions, to use three-partitioning, and to apply it to a four-dimensional space-time setting: here, a bounded space-time domain is embedded into a 4d-hypercube. This cube then is cut into 3^4 equally-sized subcubes . The process continues recursively and yields a cascade of adaptive Cartesian grids. Such grids can be traversed and serialised along a space-filling curve and thus stored efficiently with basically one bit per vertex. Such a grid cascade delivers a multiscale representation of the computational domain well-suited for geometric multigrid solvers. Such a grid finally has simple tensor-product structure.

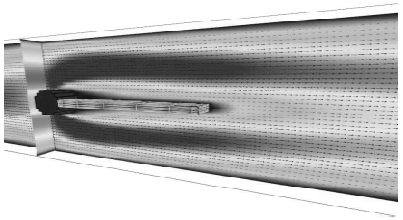
The present talk studies a plain geometric FAS solver acting on the space-time domain: We observe that the spacetree's adaptivity in both space and time can, if multiple snapshots at different time steps have to be held anyway, coarse more aggressively than time stepping and provide a nice framework to realise local time stepping in combination with dynamic adaptivity in space. We observe that the spacetree's multiscale data structure can, in combination with a full approximation storage scheme, yield first guesses of the solution quickly and, in accordance with textbook knowledge, superconverge. We observe that the additional temporal degree of freedom increases the concurrency. Different to traditional approaches, the spacetree allows us to distribute not only time slices or follow a spatial domain decomposition, but it facilitates to deploy whole space-time subdomains to different ranks. However, the load balancing for such a data structure is delicate, the adaptivity in time interplays with the communication and data flow, the h-adaptivity of the spacetree suffers from a lack of accuracy at the domain boundaries due to the O(h) accuracy, and so forth. Some of these issues and potential solutions are addressed and sketched.

Parallel Space-time Spacetrees
for Simple Parabolic Poblems

Tobias Weinzierl

Innovative Space-time Parallel Methods
Manchester, 18 June 2013





$$\int_{\Omega\times(0,T)} (\partial_t u - \Delta u, \varphi)\, d(x,t) = \int_{\Omega\times(0,}$$

$$M\left(u(t+\tau) - u(t)\right) + Lu(t+\tau) = Mf$$

$$+\tau L)u(t+\tau) =: A(\tau)u(t+\tau) = \tau Mf$$

$$u(t+\tau)^{(n+1)} = u(t+$$

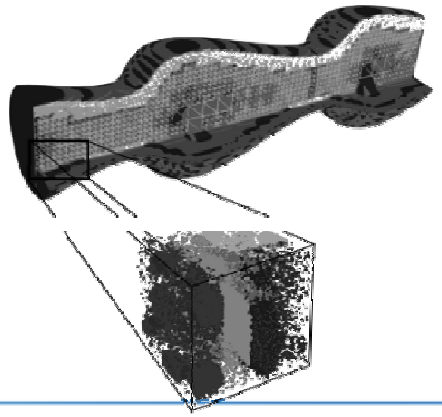
$$u(t+\tau)^{(0)} = u(t)$$

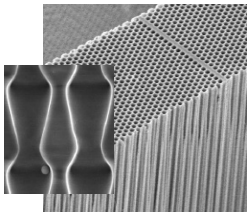
A MINIMALLY INVASIVE APPROACH TO ADAPTIVE MESH REFINEMENT WITH LOCAL TIME STEPPING FOR HYPERBOLI CONSERVATION LAW SOLVERS

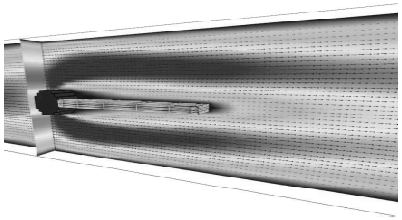
KRISTOF UNTERWEGER *, TOBIAS WEINZIERL †, DAVID I. KETCHESON ‡, AND ARON AHMADIA ‡

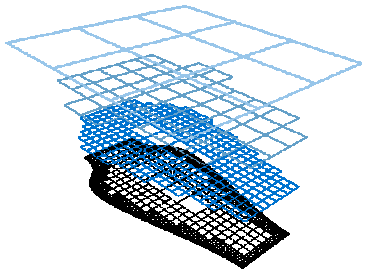
Abstract. We present implementation idioms combining a spacetree-based adaptive mesh
finement (AMR) framework with solvers for hyperbolic partial differential equations (PDEs) operat
on regular Cartesian grids. The combination is minimal invasive to the PDE developer in that o
routines written for regular Cartesian grids are used on adaptive meshes without modification. T
AMR framework traverses the adaptive grid autonomously, calls PDE-specific routines, and preser
the data consistency on the adaptive mesh. Furthermore, the framework provides a very simple p
gramming interface to manipulate the adaptive mesh. We thus lower the implementation thresh
for application specialists who want to extend existing code with AMR features. Three features
implemented in the Peano framework. As proof-of-concept, we combine Peano with Clawpack's l
Claw package, an explicit finite volume solver for general hyperbolic PDEs. **The following nee
work:** The present paper discusses the software merger by means of a classical conservative expli
time stepping scheme. For this, it proposes an implementational modification of the Berger-Coll
AMR concept that reduces the memory footprint as it does not work with completely overlappi
grids and that reduces the total solution time as it allows different subgrids of the same resolut
to advance in time with different time step sizes. As a result, the time stepping is replaced by
iterative, local time stepping scheme advancing in each grid traversal some subgrids in time. T
overall simulation then permanently spans a transient time interval shifting towards the terminat
time.

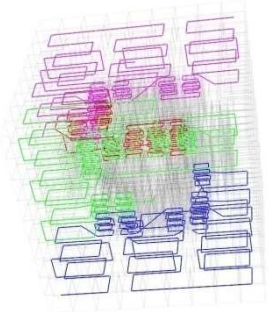
1. Introduction. Adaptive mesh refinement (AMR) is a state-of-the-art tec
nique to tackle partial differential equations (PDE) whose solution and regions
interest vary significantly in space: it applies grids with a fine resolution where the s
lution changes rapidly, is non-smooth, or exhibits complicated boundaries, right-ha
sides, and so forth. It resolves the remaining regions with coarser grids, and the
regions and their characteristics, i.e. which grid resolutions are reasonable, chan
over simulation time. As the grid resolution determines the computational workloa











n=2
n=3 = 2+1
n ← n+1

#define Dim2
#define Dim3

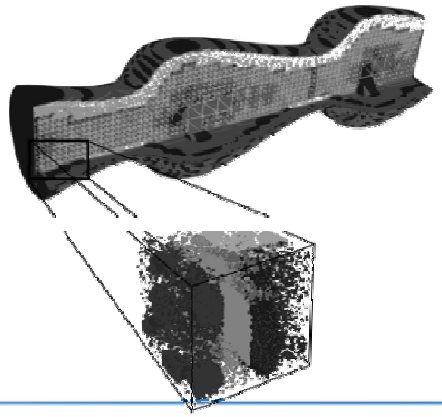
$$
\begin{aligned}
\int_{\Omega\times(0,T)} (\partial_t u - \Delta u, \varphi)\, d(x,t) &= \int_{\Omega\times(0,} \\
M\left(u(t+\tau) - u(t)\right) + Lu(t+\tau) &= Mf \\
+\tau L)u(t+\tau) =: A(\tau)u(t+\tau) &= \tau Mf \cdot \\
u(t+\tau)^{(n+1)} &= u(t+\cdot \\
u(t+\tau)^{(0)} &= u(t)
\end{aligned}
$$

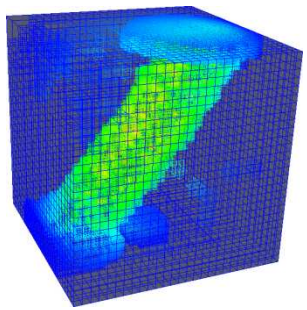
A MINIMALLY INVASIVE APPROACH TO ADAPTIVE MESH REFINEMENT WITH LOCAL TIME STEPPING FOR HYPERBOLI CONSERVATION LAW SOLVERS

KRISTOF UNTERWEGER *, TOBIAS WEINZIERL †, DAVID I. KETCHESON ‡, AND ARON AHMADIA §

Abstract. We present implementation idioms rendering a spacetree-based adaptive mesh ... finement (AMR) framework with solvers for hyperbolic partial differential equations (PDEs) operat ... on regular Cartesian grids. The combination is minimal invasive to the PDE developer in that o ... routines written for regular Cartesian grids are used on adaptive meshes without modification. T ... AMR framework traverses the adaptive grid autonomously, calls PDE-specific routines, and preser ... the data consistency on the adaptive mesh. Furthermore, the framework provides a very simple p ... grooming interface to manipulate the adaptive mesh. We thus lower the implementation threshol ... for application specialists who want to extend existing code with AMR features. Three features ... implemented in the Peano framework. As proof-of-concept, we combine Peano with Clawpack's l ... Clee package, an explicit finite volume solver for general hyperbolic PDEs. **The following nov ... work.** The present paper discusses the software merger by means of a classical conservative expli ... time stepping scheme. For this, it proposes an implementational modification of the Berger-Coll ... AMR concept that reduces the memory footprint as it does not work with completely overlappi ... grids and that reduces the total solution time as it allows different subgrids of the same resolut ... to advance in time with different time step sizes. As a result, the time stepping is replaced by ... iterative, local time stepping scheme advancing in each grid traversal some subgrids in time. T ... overall simulation then permanently spans a transient time interval shifting towards the terminat ... time.

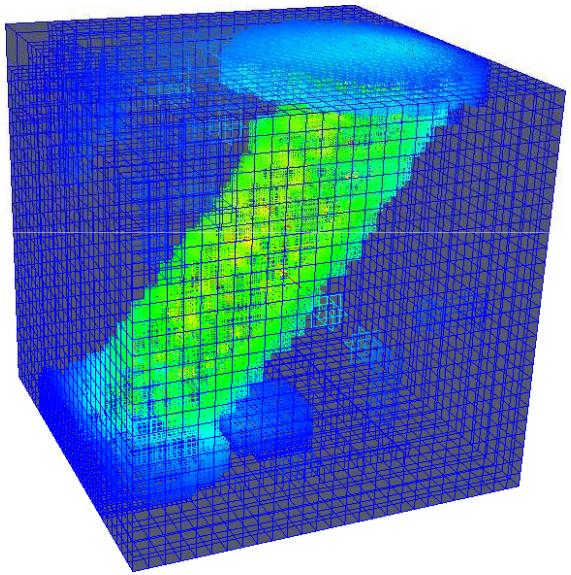
1. Introduction. Adaptive mesh refinement (AMR) is a state-of-the-art tec ... nique to tackle partial differential equations (PDE) whose solution and regions ... interest vary significantly in space: it applies grids with a fine resolution where the s ... lution changes rapidly, is non-smooth, or exhibits complicated boundaries, right-ha ... sides, and so forth. It resolves the remaining regions with coarser grids, and the ... regions and their characteristics, i.e. which grid resolutions are reasonable, chan ... over simulation time. As the grid resolution determines the computational workloa





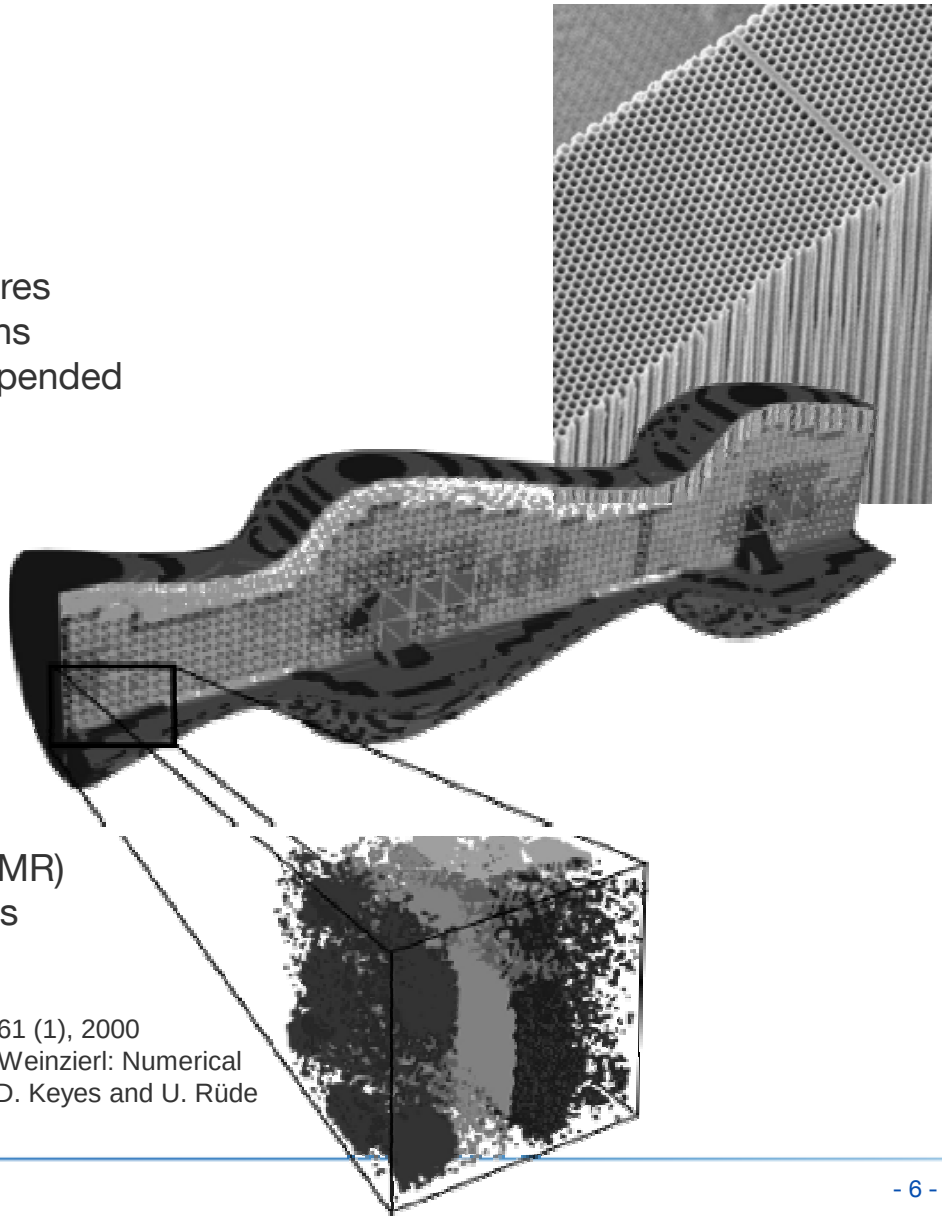
Outline

- Introduction with some historical remarks
- *k*-Spacetrees: a data structure that yields multiresolution 4d grids
- HTMG, FAS, and MLAT – for $d \geq 2$
- Space-time space-filling curves: meandering on multiple cores and multiple nodes
- Conclusion, open issues, and what was this all for



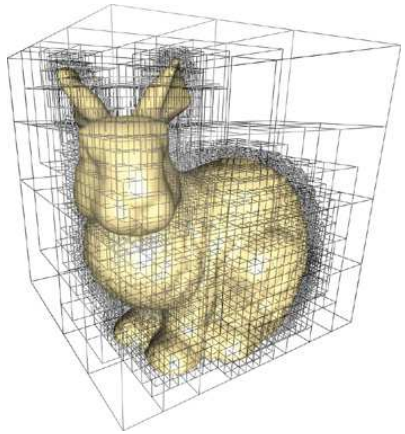
Historical remarks

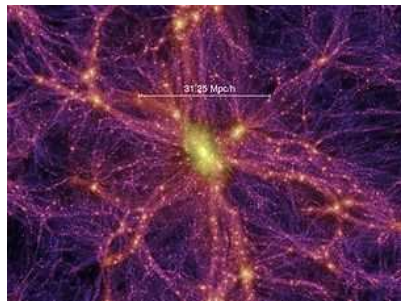
- Drift ratchet
  - Silicon wafer pierced with pores
  - Oscillating pressure conditions
  - Particles of different size suspended (diameter 0.1 µm – 1.2 µm)
- Observation
  - Particles are separated dependent on size
  - Drift ratchet acts as particle separation device
  - We neither now how nor why
- Project strategy
  - Adaptive Cartesian grids (SAMR)
  - Incompressible Navier-Stokes
  - Matrix-free linear algebra

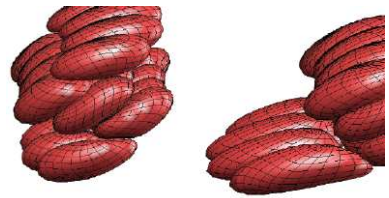


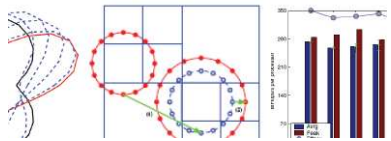
Kettner, Reimann, Hänggi, Müller: Drift ratchet, *Physical Review E* 61 (1), 2000
M. Brenk, H.-J. Bungartz, M. Mehl, I. L. Muntean, T. Neckel and T. Weinzierl: Numerical Simulation of Particle Transport in a Drift Ratchet. In C. Johnson, D. Keyes and U. Rüde (ed.), SISC, 30(6), pp. 2777–2798, 2008.

The *k*-Spacetree (octree/quadtree)



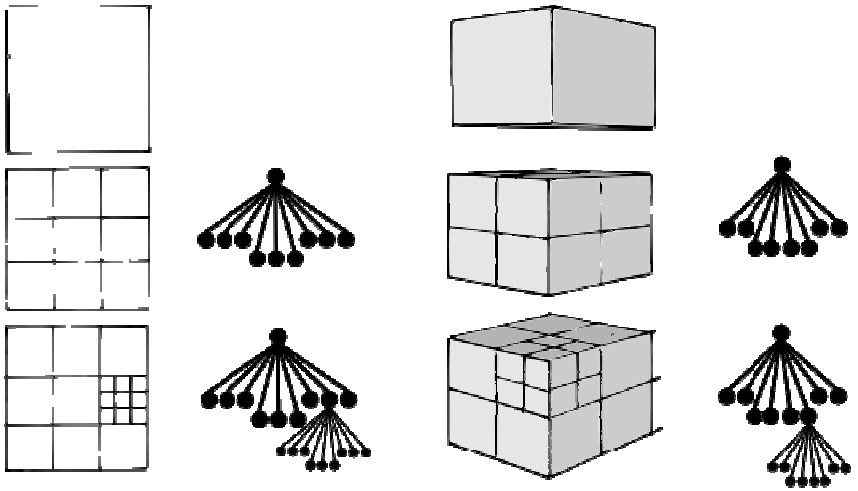






k-Spacetrees have many nice properties

- *k*-Spacetree definition
- *k*-Spacetree properties
  - Store tree instead of Cartesian grid
  - In-situ mesh generation at low setup cost
  - Multiscale representation of domain
  - Dynamic adaptivity
  - Low memory requirements (tree data structure)
  - Mathematical simplicity (hypercubes)
  - …
- *k*-Spacetree traversal
- Functional programming

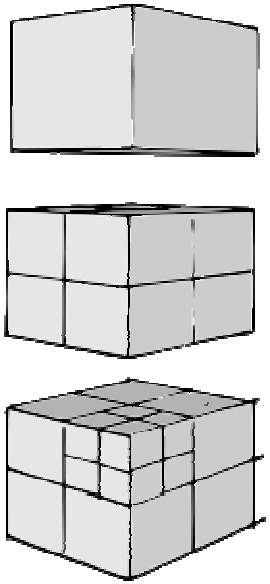


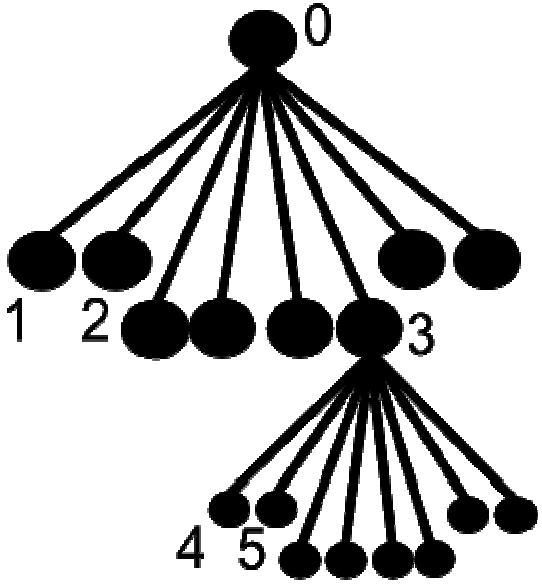
Tree traversal yields element-wise grid traversal

- *k*-Spacetree definition
- k-Spacetree properties

- *k*-Spacetree traversal
  - Depth-first traversal: linearse tree structure
  - Element-wise traversal is a standard concept for FEM
  - Tree traversal corresponds to multiscale element-wise grid traversal
  - We can embed operations of (matrix-free) solver(s) into traversal

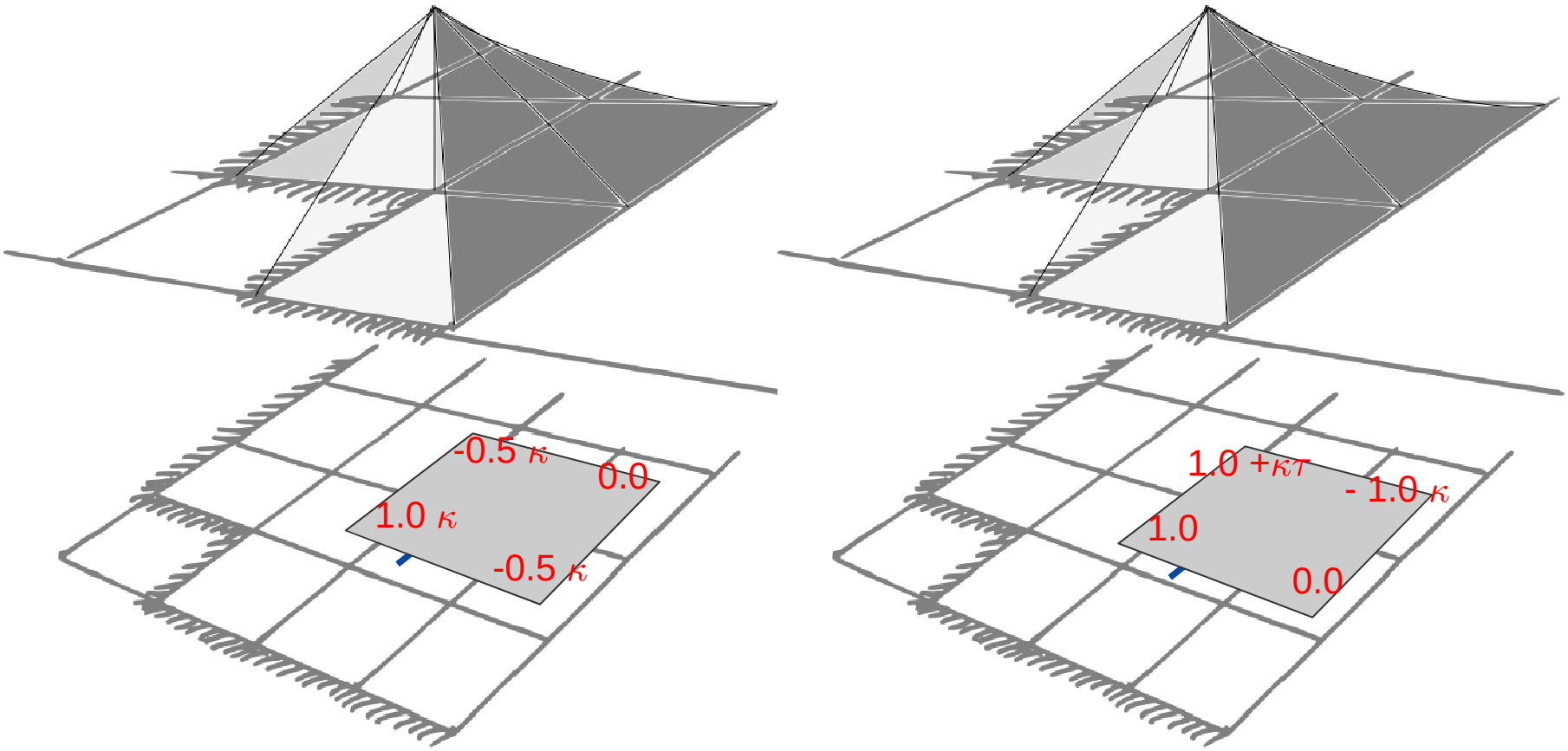
- Functional programming



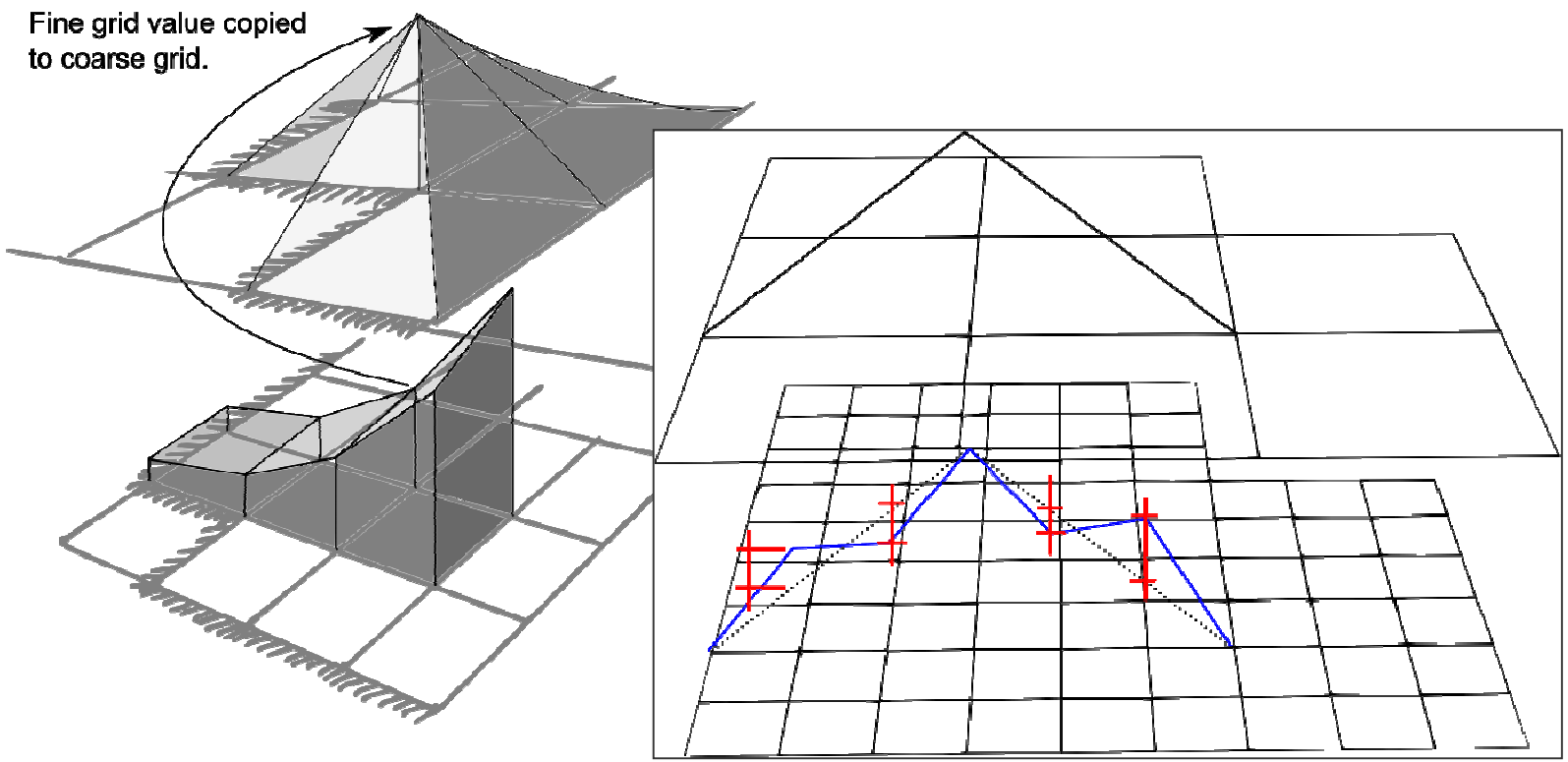


T. Weinzierl: A Framework for Parallel PDE Solvers on Multiscale Adaptive Cartesian Grids. Verlag Dr. Hut, München, 2009.
T. Weinzierl and M. Mehl: Peano - A Traversal and Storage Scheme for Octree-Like Adaptive Cartesian Multiscale Grids. In R. Tuminaro, M. Benzi, X.-C. Cai, I. Duff, H. Elman, R. Freund, K. Jordan, T. Kelley, D. Keyes, M. Kilmer, S. Leyffer, T. Manteuffel, S. McCormick, D. Silvester, H. Walker, C. Woodward and I. Yavneh (ed.), SISC, Volume 33(5), pp. 2732–2760, 2011.

Functional programming: don‘t call us – we call you



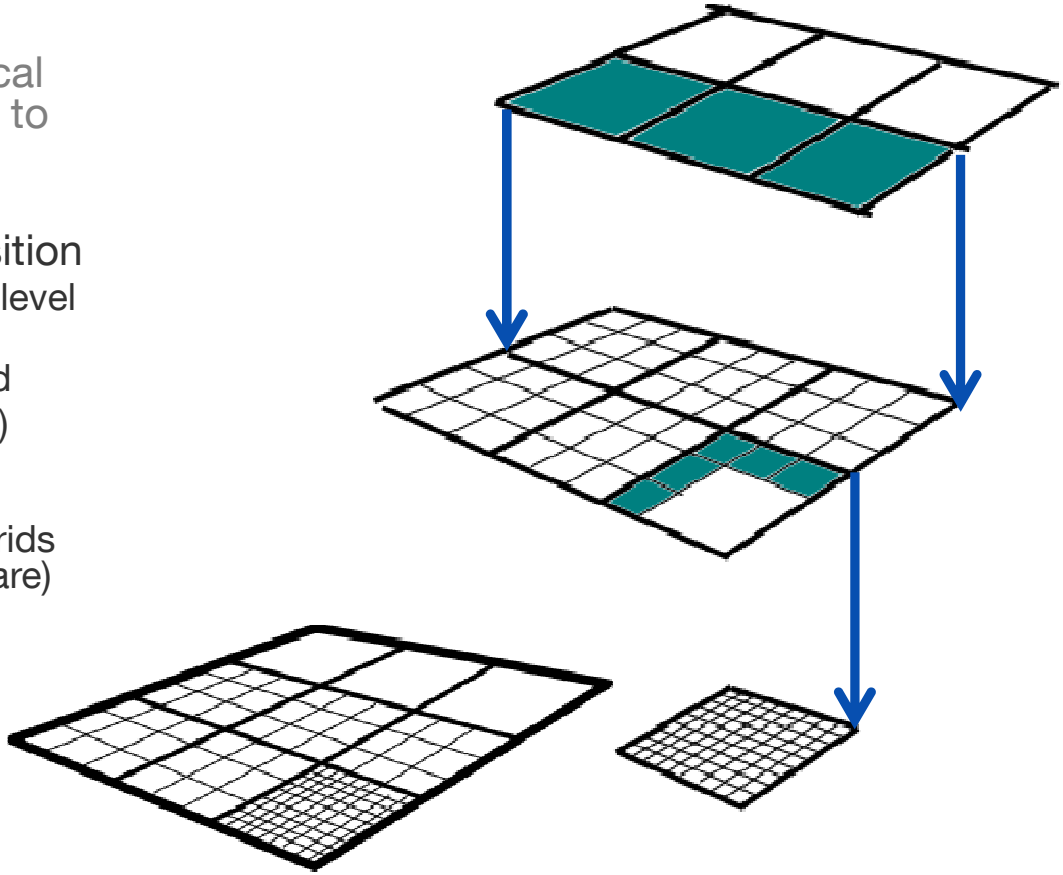
Trivial injection yields multiscale representation



M. Griebel: Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen. Teubner Skripten zur Numerik. Habilitationsschrift, TU München. Teubner, 1994.

Hierarchical generating systems tackle hanging nodes

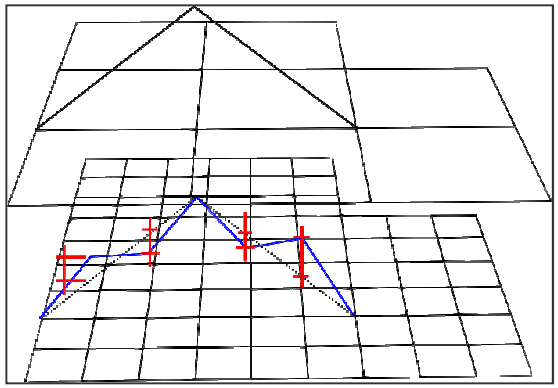
- Trivial Injection on the hierarchical generating system map system to hierarchical basis
- Overlapping domain decomposition
  - Extend any fine grid of a given level at least one element
  - Hanging nodes are interpolated
  - Solution is injected (coarsened) permanently
  - The handling of dynamically changing adaptive Cartesian grids becomes (technically; in software) trivial
  - Scheme describes local time stepping in d+1 dimensions
- From MG to FAS



A. Brandt. Multi-level adaptive technique (mlat) for fast numerical solution to boundary value problems. In Proc. 3rd Internat. Conf. on Numerical Methods in Fluid Mechanics (Paris,1972), volume 18 of Lecture Notes in Physics, pp. 82–89. Springer-Verlag, 1973

Hierarchical Transform Multigrid (HTMG)

- Trivial injection (coarsening) on the hierarchical generating system map system to hierarchical basis
- Overlapping domain decomposition
- From CS to FAS



$$
\begin{aligned}
RAPe_{\mathrm{h},\ell-1} &= Rr_{\mathrm{h},\ell} = R(b - Au_{\mathrm{h},\ell}) \\
RAP(Cu_{\mathrm{h},\ell} + e_{\mathrm{h},\ell-1}) &= Rr_{\mathrm{h},\ell} + RAPCu_{\mathrm{h},\ell} \\
u_{\mathrm{h},\ell} &= P_{\ell-1}^{\ell} C_{\ell}^{\ell-1} u_{\mathrm{h},\ell} + \hat{u}_{\mathrm{h},\ell} \\
\hat{r}_{\mathrm{h},\ell} &:= b_{\ell} - A\hat{u}_{\mathrm{h},\ell} \\
&= b_{\ell} + APCu_{\mathrm{h},\ell} - Au_{\mathrm{h},\ell} \\
&= r_{\mathrm{h},\ell} + APCu_{\mathrm{h},\ell} \\
\Rightarrow R\hat{r}_{\mathrm{h},\ell} &= Rr_{\mathrm{h},\ell} + RAPCu_{\mathrm{h},\ell} \\
&= Rr_{\mathrm{h},\ell} + ACu_{\mathrm{h},\ell}
\end{aligned}
$$

Cheap to evaluate and simple to program
if solution is coarsened anyway, i.e. if we realise
a full approximation storage scheme (FAS).

M. Griebel: Zur Lösung von Finite-Differenzen- und Finite-Element-Gleichungen mittels der Hierarchischen-Transformations-Mehrgitter-Methode. Dissertation, Technical Report, 1990

Generating system yields RHSs of FAS

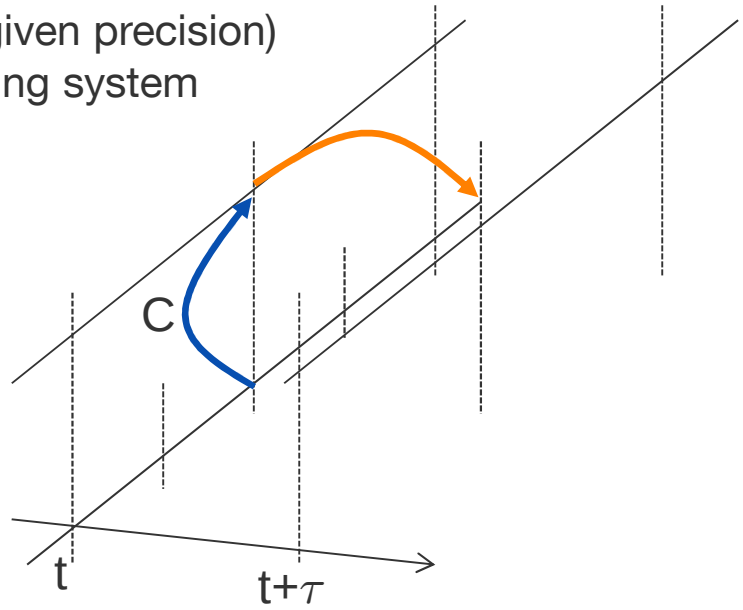
- Classical time stepping approach (implicit Euler, e.g.)

$$A(kh,\tau)u_{kh}(t+\tau) = C\cdot\ldots\cdot C\cdot C\,(\tau M f + M u(t))$$

(coarse until previous solution is available in given precision)

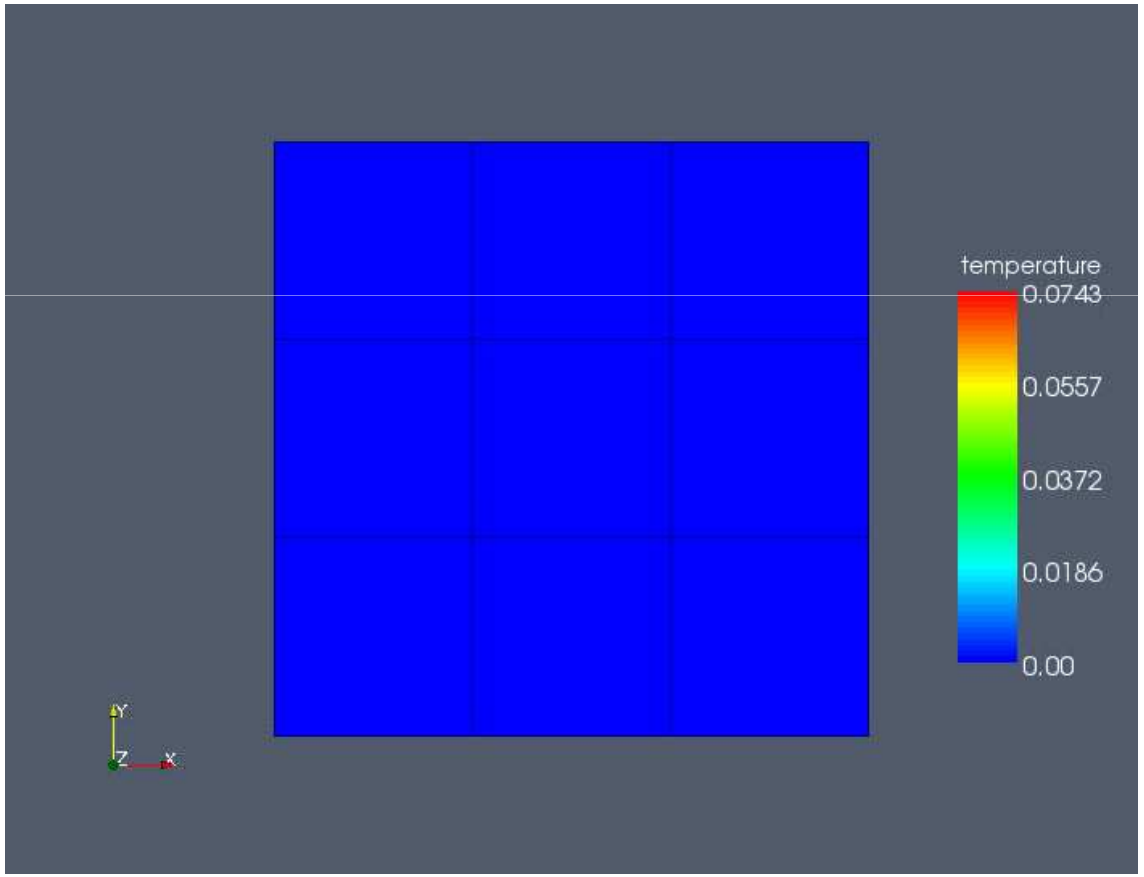
- All grid levels at t=0 available, so use generating system

$$\begin{aligned}
A(kh,\tau)u_{kh}(t+\tau) &= \tau M_{kh} C\cdot\ldots\cdot C\cdot C f + \\
&\quad M_{kh} C\cdot\ldots\cdot C\cdot C u(t) \\
A((k-1)\cdot h,\tau)u_{kh}(t+\tau) &= \tau M_{(k-1)\cdot h} C\cdot\ldots\cdot C f + \\
&\quad M_{(k-1)\cdot h} C\cdot\ldots\cdot C u(t) \\
\ldots &\quad \ldots \\
A(2h,\tau)u_{2h}(t+\tau) &= \tau M_{2h} C f + M_{2h} C u(t) \\
A(h,\tau)u_{h}(t+\tau) &= \tau M_{h} f + M_{h} u(t)
\end{aligned}$$

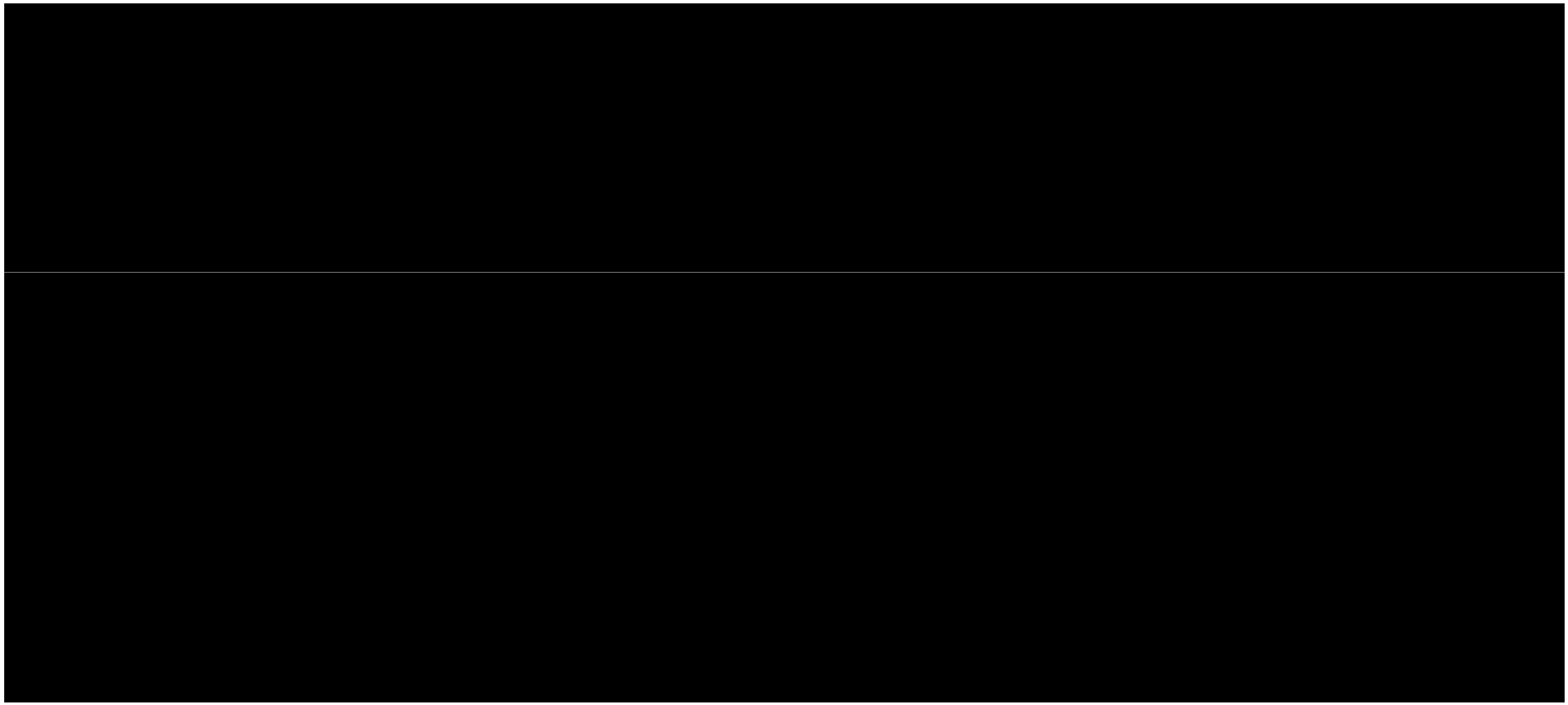


- Multiple time steps with same update rule on different scales simultaneously
- Couple scheme iteratively and refine grid dynamically

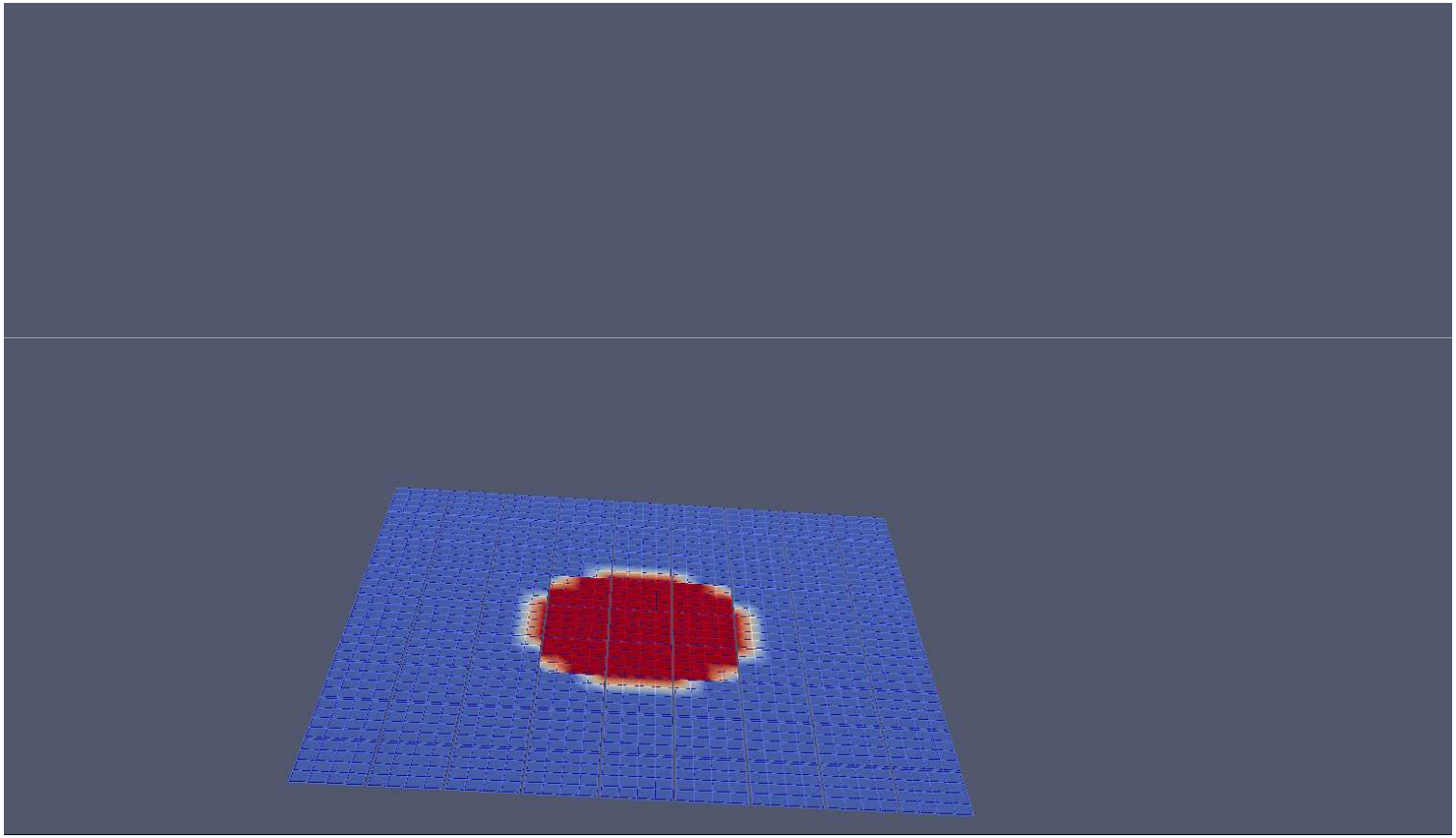
A parabolic toy demonstrator: $\partial_t u - \nabla (a \nabla) u = f$



Global FMG in time (2+1d)

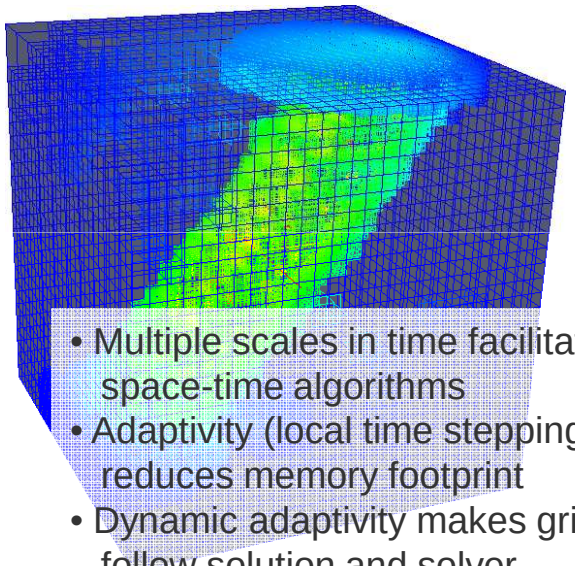


Similar software concepts hold for different PDEs



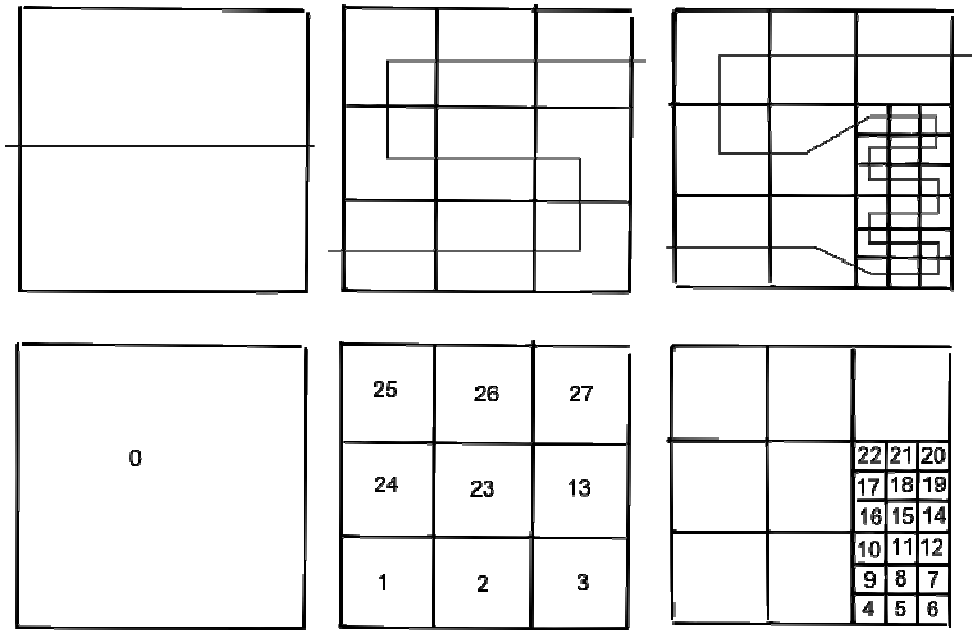
Outline

- Introduction with some historical remarks
- *k*-Spacetrees: a data structure that yields multiresolution 4d grids
- HTMG, FAS, and MLAT – for d $\geq$ 2
- Space-time space-filling curves: meandering on multiple cores and multiple nodes
- Conclusion, open issues, and what was this all for

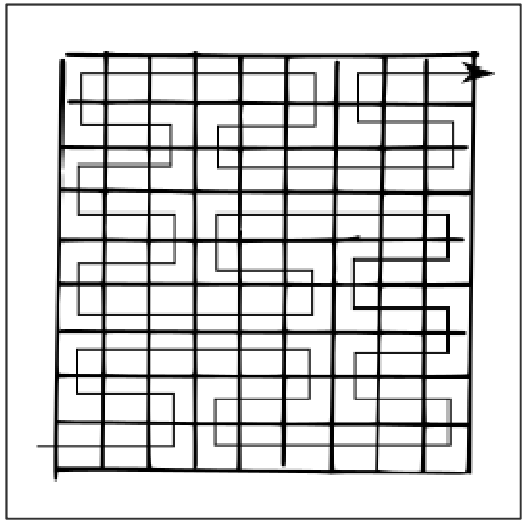


- Multiple scales in time facilitate space-time algorithms
- Adaptivity (local time stepping) reduces memory footprint
- Dynamic adaptivity makes grid follow solution and solver
- Programming model is simple (functional programming)

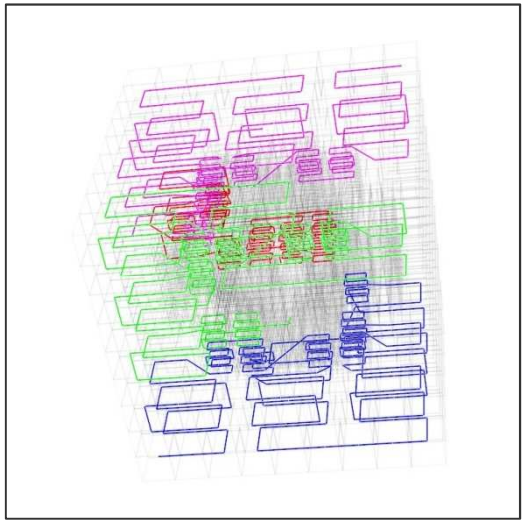
Make DFS deterministic due to Peano SFC



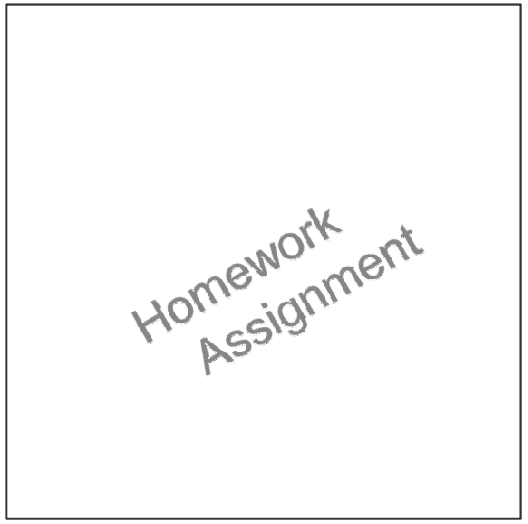
Peano curve also meanders in 4d



d=2



d=3



d=4

The memory footprint is very small

- Stacks allow you to eliminate pointers, hash tables, ..., i.e. lots of (typical) overhead
- Memory footprint illustrated
  - BoxMG, dynamically adaptive
  - Reduced floating point precision (hierarchical transform)
  - 1.61e+08 vertices on my local workstation (8 Gbyte)
  - 4.03e+07 vertices on Jugene node (VM)
  - 6.31e+06 vertices on SuperMUC core
  - 4.02e+06 vertices on one core of my local workstation
  - 2.12e+06 vertices on one core of Shaheen (SMP)
  - 1.07e+06 vertices on core with 0.5 Gbyte memory, i.e. on one core of Jugene (SMP), e.g.

Timestepping (matrix-free)			
d	Explicit Euler	Jacobi	BoxMG (A,P,R)
1	18	26	52
2	18	26	144
3	18	26	580

Space-time Discretisation (matrix-free)			
d	Explicit Euler	Jacobi	BoxMG (A,P,R)
1	18	18	44
2	18	18	136
3	18	18	572

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

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

SFC induces spatial and temporal data locality

Tracer particles	Mesh width			L2 data throughput	L2 cache miss rate	L3 data throughput
4e+06	0.0500	1.92e+07	281.52	39.52	0.0042	27.11
4e+06	0.0100	1e+07	155.39	53.95	0.0049	52.75
4e+06	0.0050	9.96e+06	155.17	53.99	0.0049	52.73
4e+07	0.1000	1.94e+07	284.20	387.53	0.0042	266.73
4e+07	0.0500	1.94e+07	284.48	387.64	0.0042	266.91
4e+07	0.0100	1.31e+07	192.66	410.46	0.0034	293.15
4e+07	0.0050	1.3e+07	192.57	410.17	0.0034	293.21
4e+06	0.0500	9.89e+06	245.32	77.07	0.0047	58.44
4e+06	0.0100	1.11e+05	44.53	6909.79	0.0059	4706.08
4e+06	0.0050	1.1e+05	44.48	6901.56	0.0059	4726.29
4e+07	0.1000	1.36e+07	306.95	606.77	0.0046	425.99
4e+07	0.0500	1.36e+07	306.58	607.18	0.0046	425.88
4e+07	0.0100	9.71e+05	56.08	7468.49	0.0060	5351.97

Measured on SandyBridge dual socket with Likwid.

Stationary solution with tracer particles; from B. Verleye, B. Reps, D. Roose and T. Weinzierl: Particle-in-Cell Realisations On Spacetrees. to be published.

SFC induces spatial and temporal data locality

Tracer particles	Mesh width			L2 data throughput	L2 cache miss rate	L3 data throughput
4e+06	0.0500	1.92e+07	281.52	39.52	0.0042	27.11
4e+06	0.0100	1e+07	155.39	53.95	0.0049	52.75
4e+06	0.0050	9.96e+06	155.17	53.99	0.0049	52.73
4e+07	0.1000	1.94e+07	284.20	387.53	0.0042	266.73
4e+07	0.0500	1.94e+07	284.48	387.64	0.0042	266.91
4e+07	0.0100	1.31e+07	192.66	410.46	0.0034	293.15
4e+07	0.0050	1.3e+07	192.57	410.17	0.0034	293.21
4e+06	0.0500	9.89e+06	245.32	77.07	0.0047	58.44
4e+06	0.0100	1.11e+05	44.53	6909.79	0.0059	4706.08
4e+06	0.0050	1.1e+05	44.48	6901.56	0.0059	4726.29
4e+07	0.1000	1.36e+07	306.95	606.77	0.0046	425.99
4e+07	0.0500	1.36e+07	306.58	607.18	0.0046	425.88
4e+07	0.0100	9.71e+05	56.08	7468.49	0.0060	5351.97

2d

2+1d

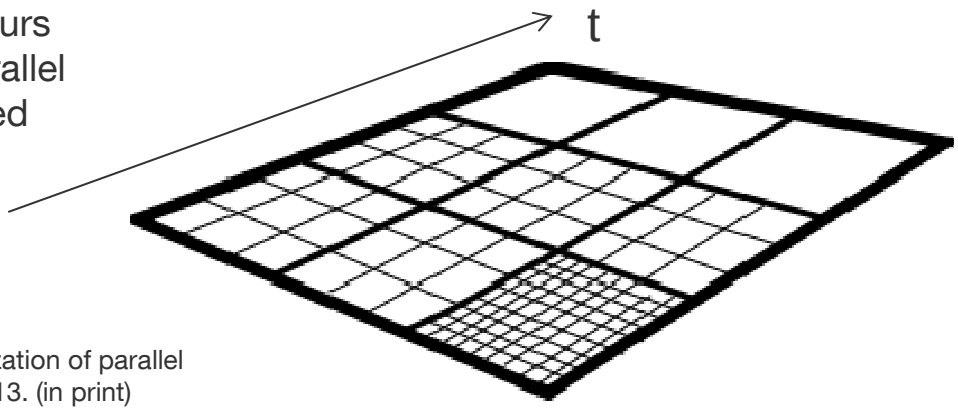
cache oblivous

Measured on SandyBridge dual socket with Likwid.

Stationary solution with tracer particles; from B. Verleye, B. Reps, D. Roose and T. Weinzierl: Particle-in-Cell Realisations On Spacetrees. to be published.

Find regular patches in space-time grid for multicore

- DFS and SFC are sequential and induce lots of integer arithmetics
- Identify regular patches (clusters)
  - In space-time domain
  - On-the-fly
  - Throw away sfc there and handle grid entities en bloc
- Multiscale RB colouring
  - Red-black style with 2^d colours
  - Invoke cell operations in parallel
  - Concurrency level is bounded by $3^k\ d\ 2^{-d}$
  - OpenMP or TBB

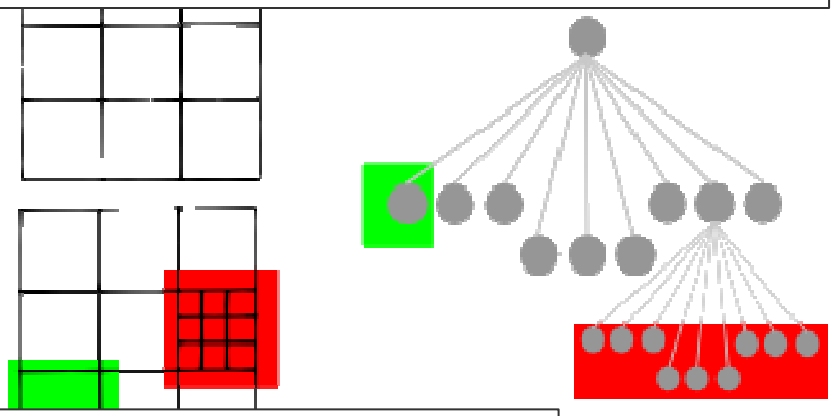


M. Schreiber, T. Weinzierl and H.-J. Bungartz: Cluster optimization of parallel simulations with dynamically adaptive grids. In EuroPar 2013. (in print)

W. Eckhardt and T. Weinzierl: A Blocking Strategy on Multicore Architectures for Dynamically Adaptive PDE Solvers. In R. Wyrzykowski, J. Dongarra, K. Karczewski and J. Wasniewski (ed.), Parallel Processing and Applied Mathematics, PPAM 2009, Volume 6068(1) of Lecture Notes in Computer Science, pp. 567--575. Springer-Verlag, 2010.

Patch identification is an analysed grammar

$$p(e) = \begin{cases} 0 & \text{for a leaf with no refinement flag set and} \\ & \text{no adjacent vertex hanging, or} \\ -1 & \text{for all other leaves.} \end{cases}$$

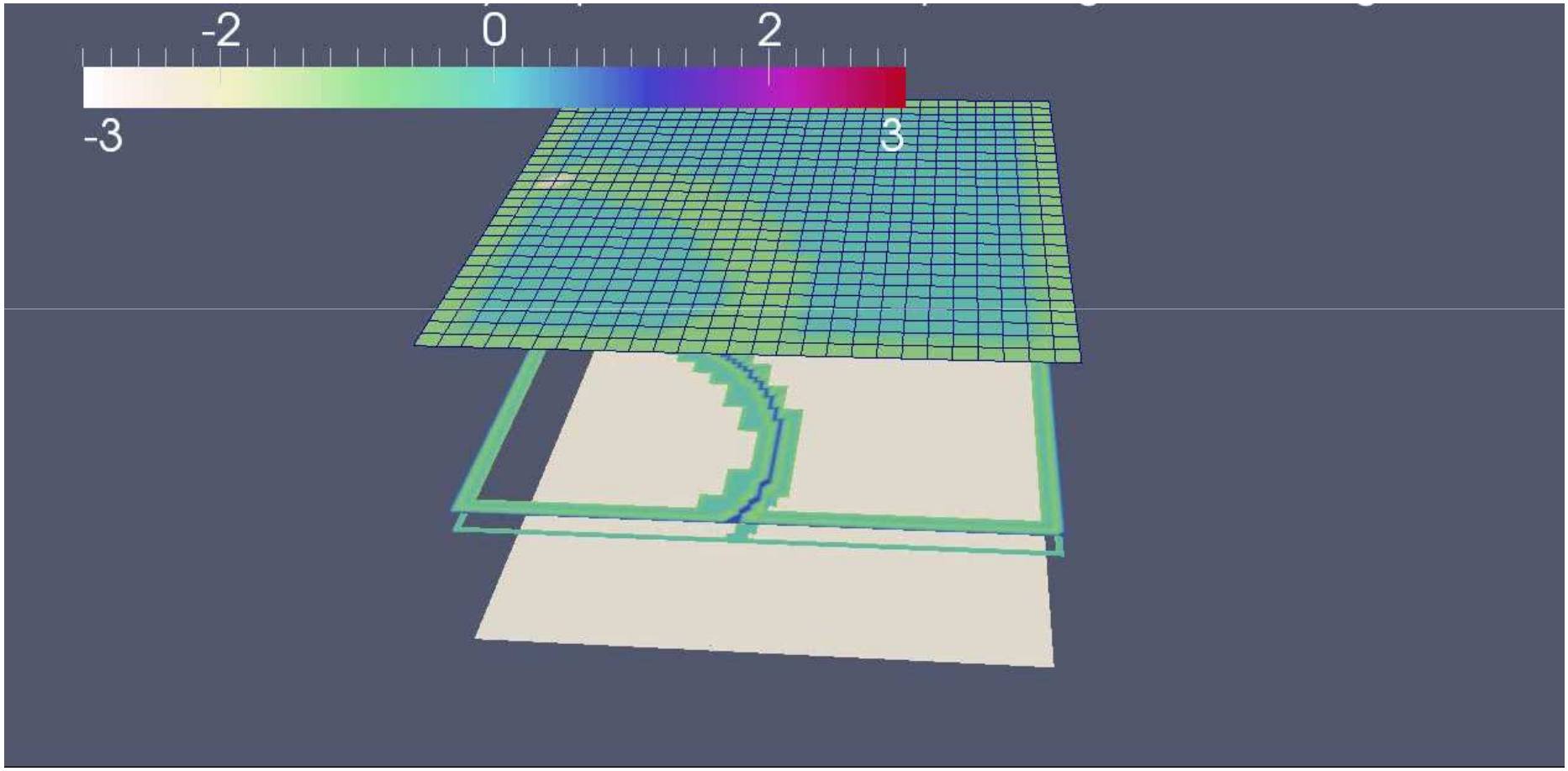


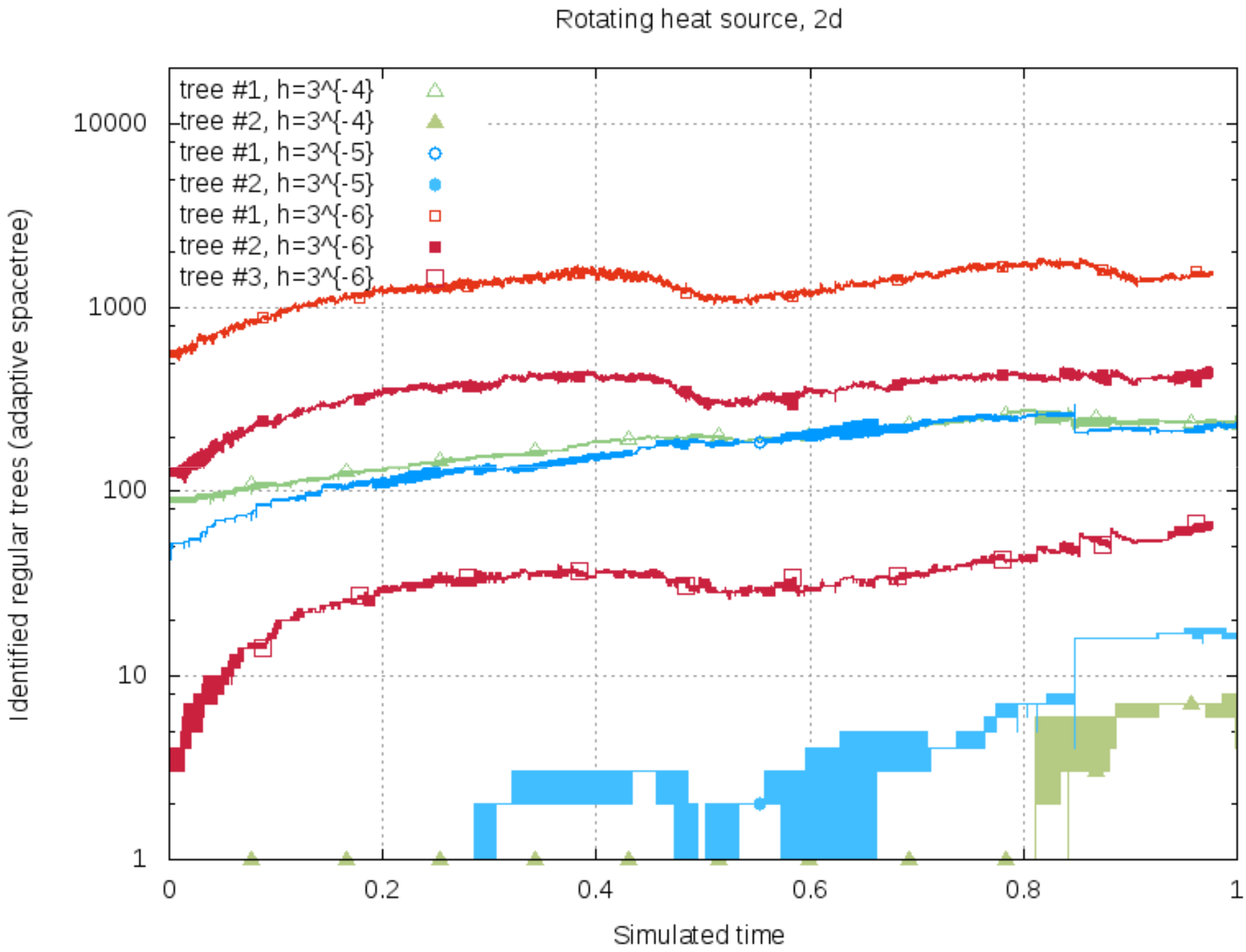
$$p(e) = \begin{cases} i+1 & \text{if} \quad \exists i \geq 0 : \forall c \sqsubseteq e : p(c) = i \\ -1 & \text{else} \end{cases}$$

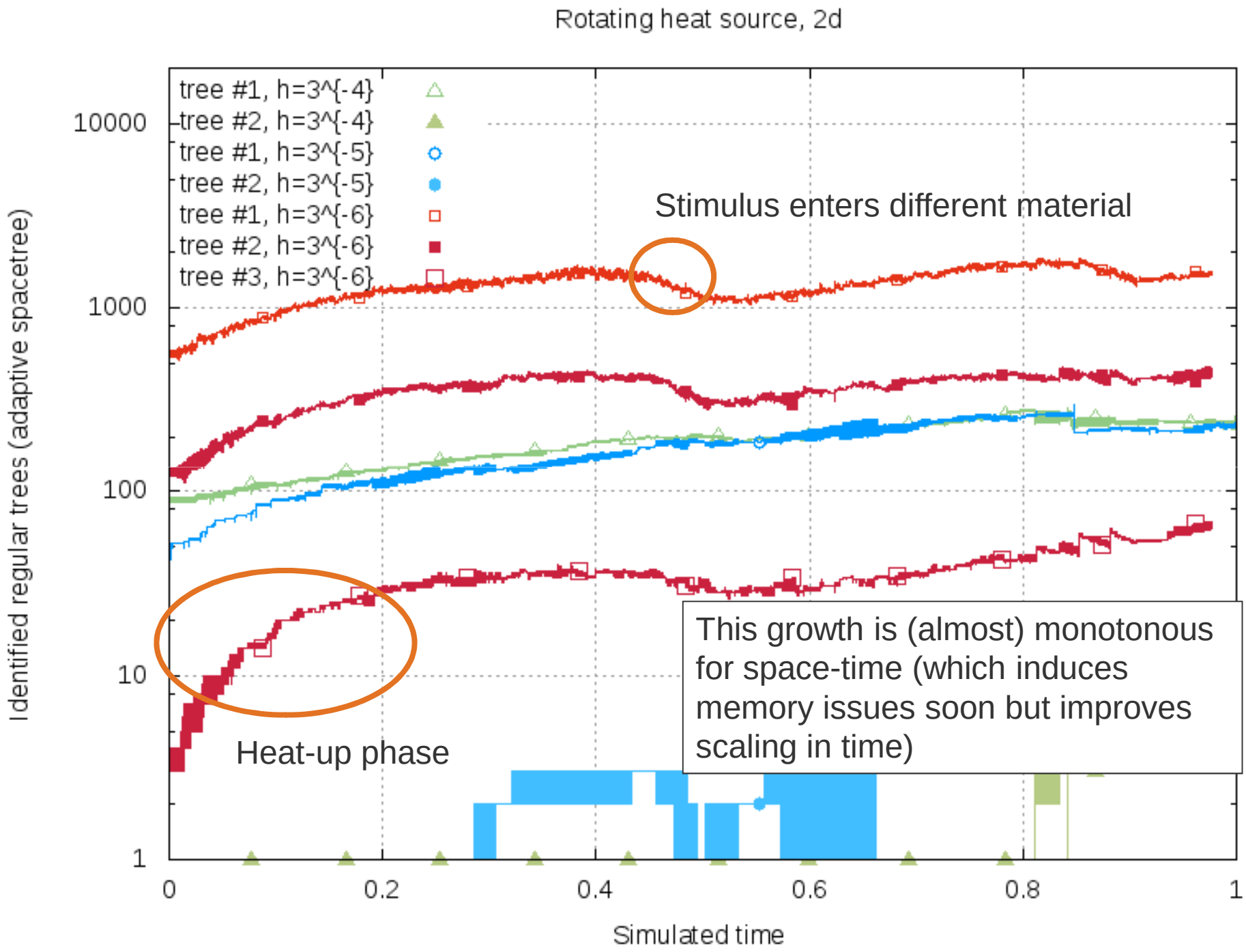
Marker semantics:

- marker *p(e)*$\geq$1: Regular subtree
- marker *p(e)*=0: Invariant (static) leaf
- marker *p(e)*=-1: Invariant or adaptive subtree

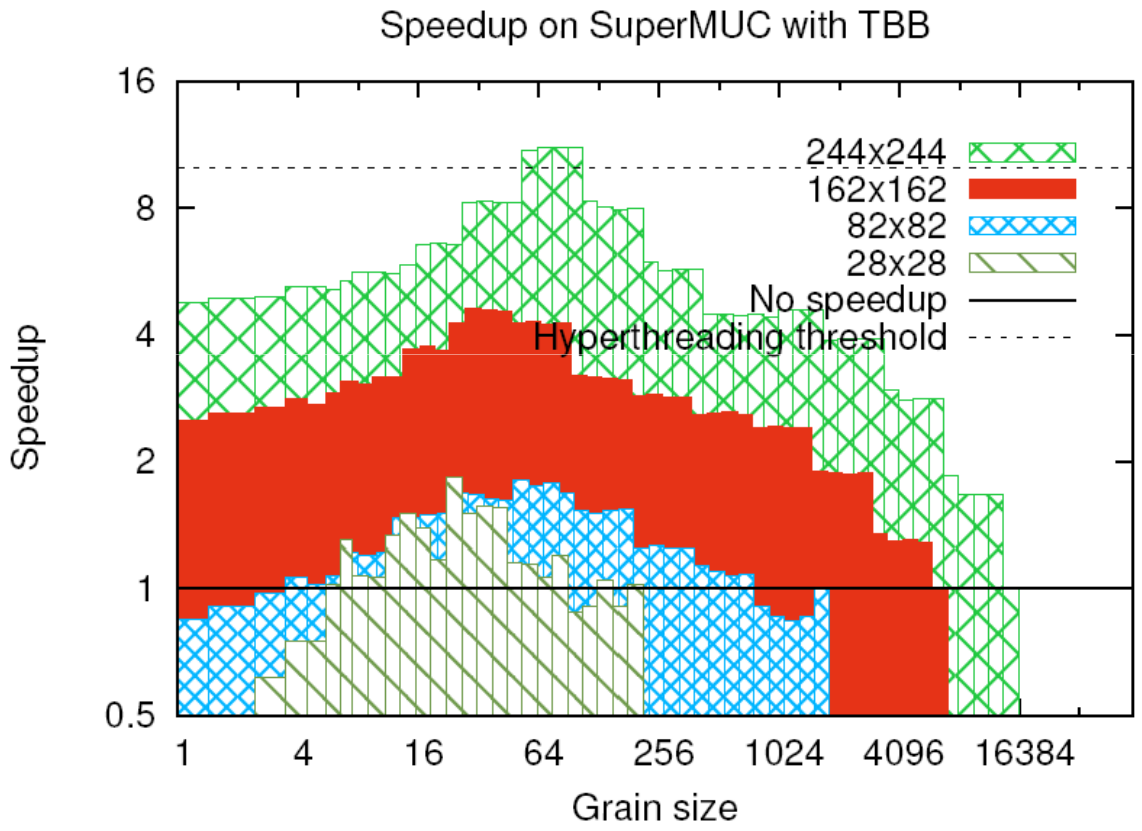
Identify regular subtrees on-the-fly




Rotating heat source, 2d
tree #1, h=3^{-4}
tree #2, h=3^{-4}
tree #1, h=3^{-5}
tree #2, h=3^{-5}
tree #1, h=3^{-6}
tree #2, h=3^{-6}
tree #3, h=3^{-6}
10000
1000
100
10
1
0
0.2
0.4
0.6
0.8
1
Identified regular trees (adaptive spacetree)
Simulated time

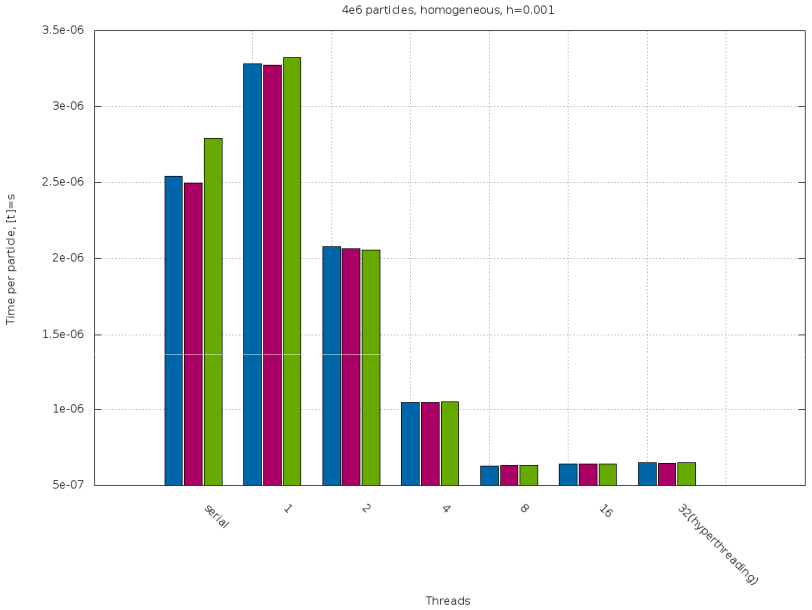

Rotating heat source, 2d
tree #1, h=3^{-4}
tree #2, h=3^{-4}
tree #1, h=3^{-5}
tree #2, h=3^{-5}
tree #1, h=3^{-6}
tree #2, h=3^{-6}
tree #3, h=3^{-6}
Stimulus enters different material
This growth is (almost) monotonous for space-time (which induces memory issues soon but improves scaling in time)
Heat-up phase
Identified regular trees (adaptive spacetree)
Simulated time
10000
1000
100
10
1
0
0.2
0.4
0.6
0.8
1

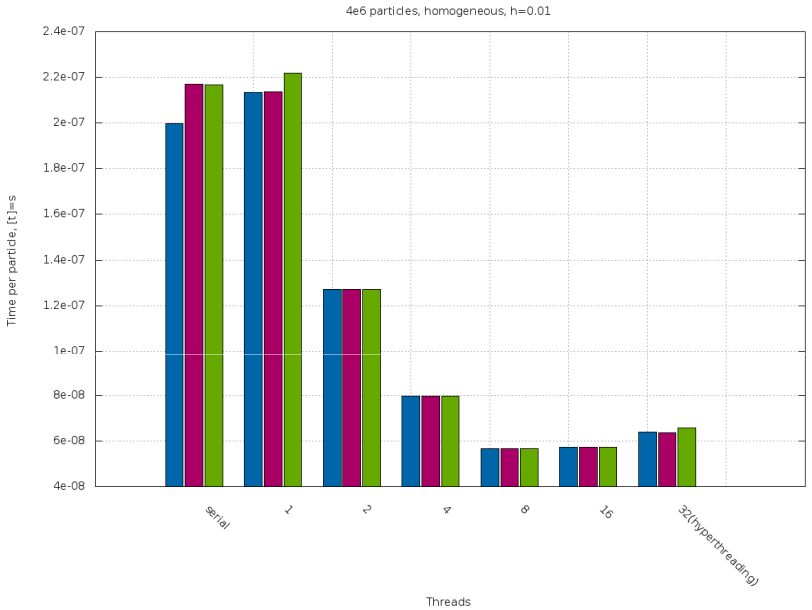
Right Choice of Grain-size (Batches) is Essential



S. Nogina, K. Unterweger and T. Weinzierl: Autotuning of Adaptive Mesh Refinement PDE Solvers on Shared Memory Architectures. In R. Wyrzykowski, J. Dongarra, K. Karczewski and J. Wasniewski (ed.), PPA 2011, Volume 7203 of Lecture Notes in Computer Science, pp. 671--680. Springer-Verlag, Heidelberg, Berlin, 2012.

Some typical speedups

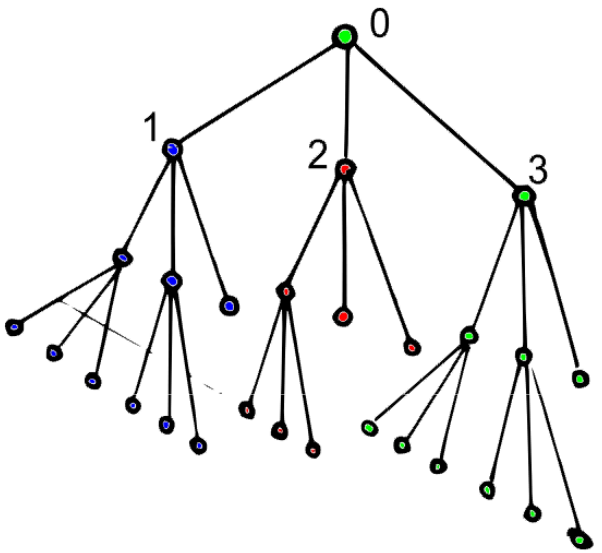


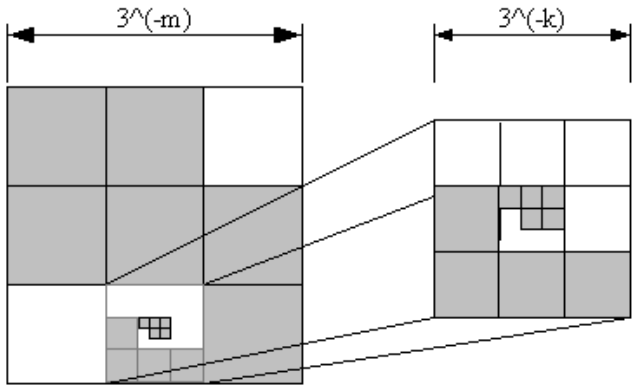


- Scalability improves the higher the regular subtree (h=0.01 (4) vs. 0.001 (>5))
- Load imbalances are less significant with threaded code
- TBB overhead is the more significant the smaller grid
- Hyperthreading here is not of great use
- Recursion unrolling suffers if we go beyond NUMA domains

SFC decomposition yields space-time subdomains

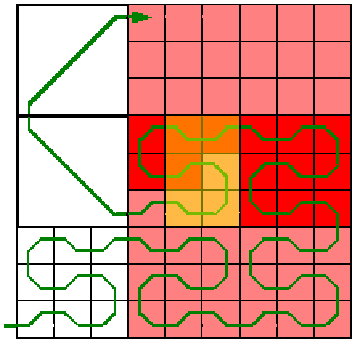
- Grid decomposition scheme
  - Subtrees are deployed to mpi ranks, i.e. no SFC cuts, however
  - Subdomains exhibit SFC layout
  - Subdomains are space-time partitions
- Space-time subdomain properties
- Open issues

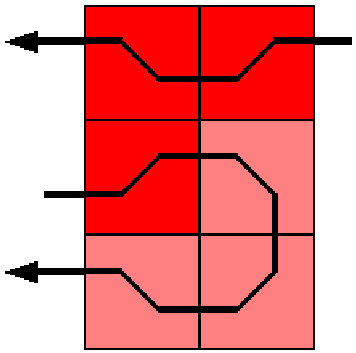


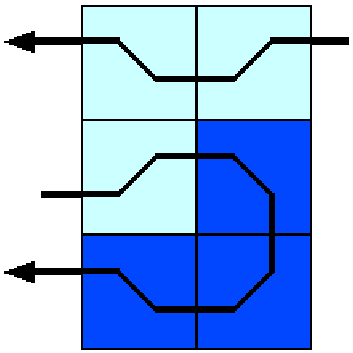


SFC decomposition yields space-time subdomains

- Grid decomposition scheme
- Space-time subdomain properties
  - Good surface-volume ratio
  - SFC meandering eliminates need for resort
  - SFC induces order on submanifold
    -> fire&forget in background
- Open issues







SFC decomposition yields space-time subdomains

- Grid decomposition scheme
- Space-time subdomain properties
  - Good surface-volume ratio
  - SFC meandering eliminates need for resort
  - SFC induces order on submanifold
    -> fire&forget in background
- Open issues
  - Global reduction is expensive
  - Load balancing is delicate
  - Is it necessary to exchange whole subdomain surface all the time?
  - Can we predict grid creation for lb?

$$\max_k \leq C\left(\frac{n}{p}\right)^{1-1/d} \qquad C \geq \sqrt[d]{\frac{2d^{d-1}\pi^{d/2}}{\Gamma(d/2)}}$$

$$\left(3^d\right)^M = n$$

Model for regular grid with supartition spanning M grid levels

$$\left(3^d\right)^{M-m-1} \leq \frac{n}{p} \leq \left(3^d\right)^{M-m}$$

$$s(n,p) \leq 2\sum_{k=m}^{M-1} s_k \leq \frac{4d}{1-3^{1-d}} 3^{(1-d)m}$$

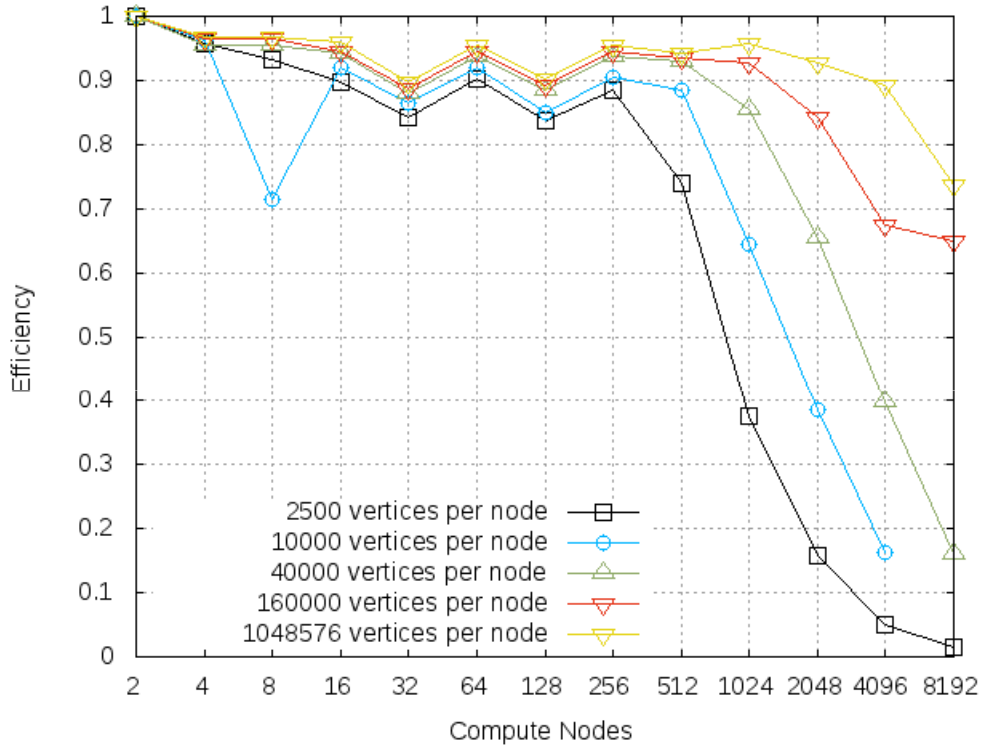
Upper bound for constant

$$s' \leq \frac{4d}{1-3^{1-d}} 3^{(1-d)m+M(d-1)} \leq \frac{4d}{1-3^{1-d}} 3^{d-1}\left(\frac{n}{p}\right)^{1-1/d}$$

$$s \leq \frac{3^{d-1}}{1-3^{1-d}} \underbrace{p^{1/d-1}}_{\leq 1} s' + 3\cdot 2^{d-1}\log_3 \sqrt[d]{p}$$

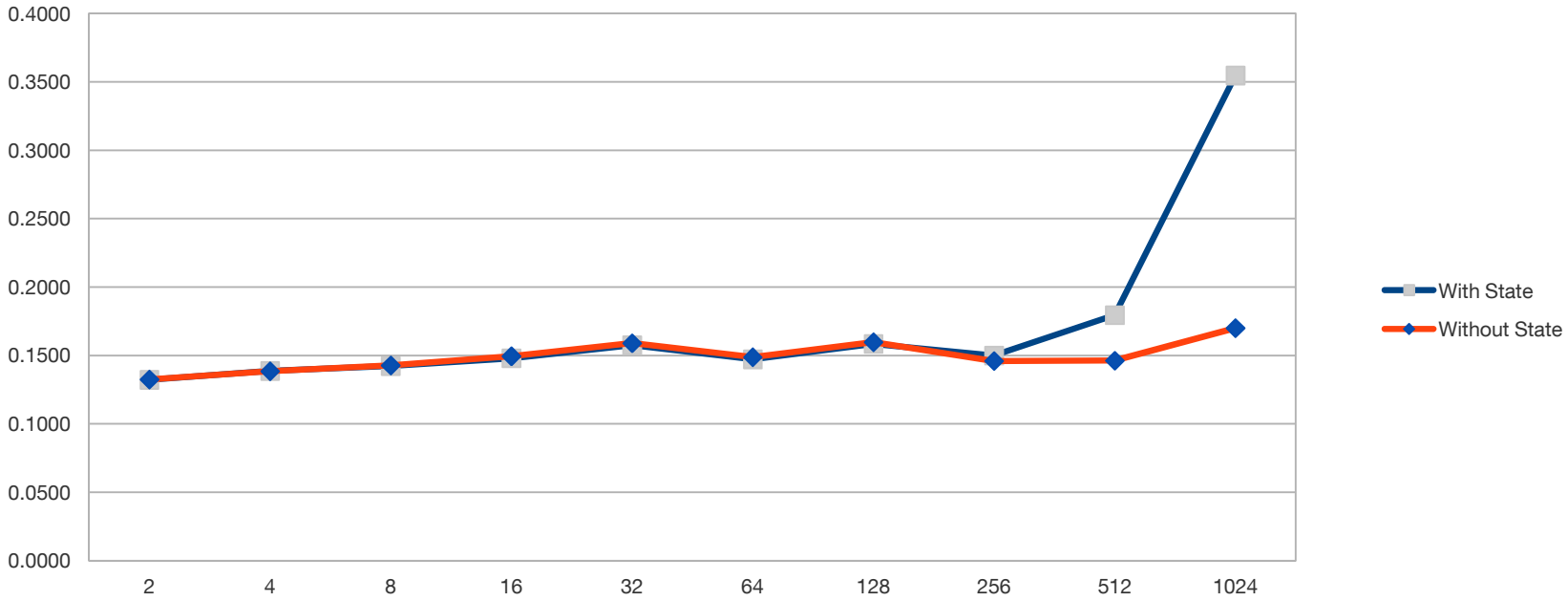
H.-J. Bungartz, M. Mehl and T. Weinzierl: A Parallel Adaptive Cartesian PDE Solver Using Space--Filling. In W. E. Nagel, W. V. Walter and W. Lehner (ed.), Euro-Par 2006, Parallel Processing, 12th International Euro-Par Conference, Volume 4128 of LNCS, pp. 1064–1074, 2006.

Strong efficiency



BlueGene/P (Jülich/KAUST)
d=2+1
V-cycle (i.e. stationary grid with optimal load balancing)

Weak scalability with(out) global synchronisation



BlueGene/P (Jülich/KAUST)
State represents global reduction data (such as a global space-time residual)
Problem size: $\mathbf{72}^2$ spatial grid (d=2+1)
Time [t]=s per multigrid cycle (i.e. stationary grid with optimal load balancing) (*)

(*) Results due to Kristof Unterweger

Weak time per cell without reduction

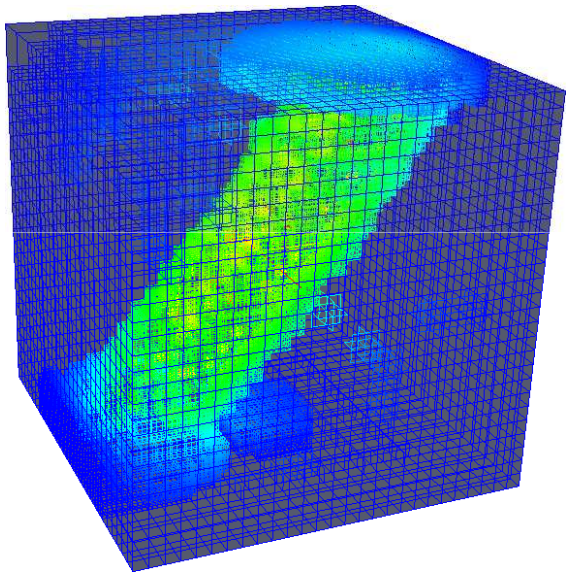
Nodes (SMP)	Cells per Node	Time per cell	Cells per Node	Time per cell	Cells per Node	Time per cell
2	2.5205E+03	2.6284E-05	4.0045E+04	2.5990E-05	1.0484E+06	3.9832E-06
4	2.5000E+03	1.3850E-05	4.0000E+04	1.3628E-05	1.0486E+06	2.0642E-06
8	2.4851E+03	7.1802E-06	4.0045E+04	6.8073E-06	1.0484E+06	1.0315E-06
16	2.5000E+03	3.7375E-06	4.0000E+04	3.4527E-06	1.0486E+06	5.2617E-07
32	2.5028E+03	1.9853E-06	3.9974E+04	1.8506E-06	1.0487E+06	2.8017E-07
64	2.5000E+03	9.2969E-07	4.0000E+04	8.6719E-07	1.0486E+06	1.3162E-07
128	2.5028E+03	4.9866E-07	4.0009E+04	4.5873E-07	1.0485E+06	6.9699E-08
256	2.5000E+03	2.2813E-07	4.0000E+04	2.1665E-07	1.0486E+06	3.2879E-08
512	2.4984E+03	1.1433E-07	3.9991E+04	1.0931E-07	1.0485E+06	1.6635E-08
1024	2.5000E+03	6.6406E-08	4.0000E+04	5.9479E-08	1.0486E+06	8.3863E-09

Total time [t]=s per time step (non-linear rhs, a posteriori dynamic adaptivity built in but disabled)
V(2/2) ; BoxMG memory footprint
Averaged snapshots of different time stepping and space-time variants
BlueGene/P (at Jülich and KAUST; SMP; OpenMP switched off)
No blocking global reduction involved (*)

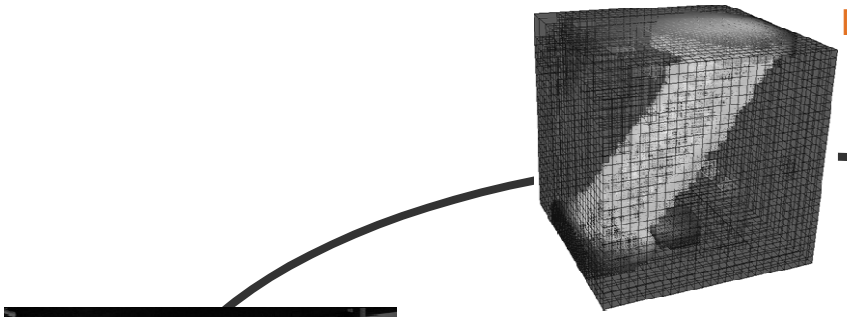
(*) Results due to Kristof Unterweger

Outline

- Introduction with some historical remarks
- *k*-Spacetrees: a data structure that yields multiresolution 4d grids
- HTMG, FAS, and MLAT – for $d \geq 2$
- Space-time space-filling curves: meandering on multiple cores and multiple nodes
- Conclusion, open issues, and what was this all for

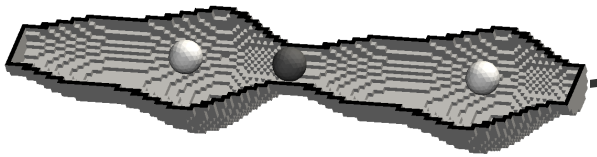


Things that work …



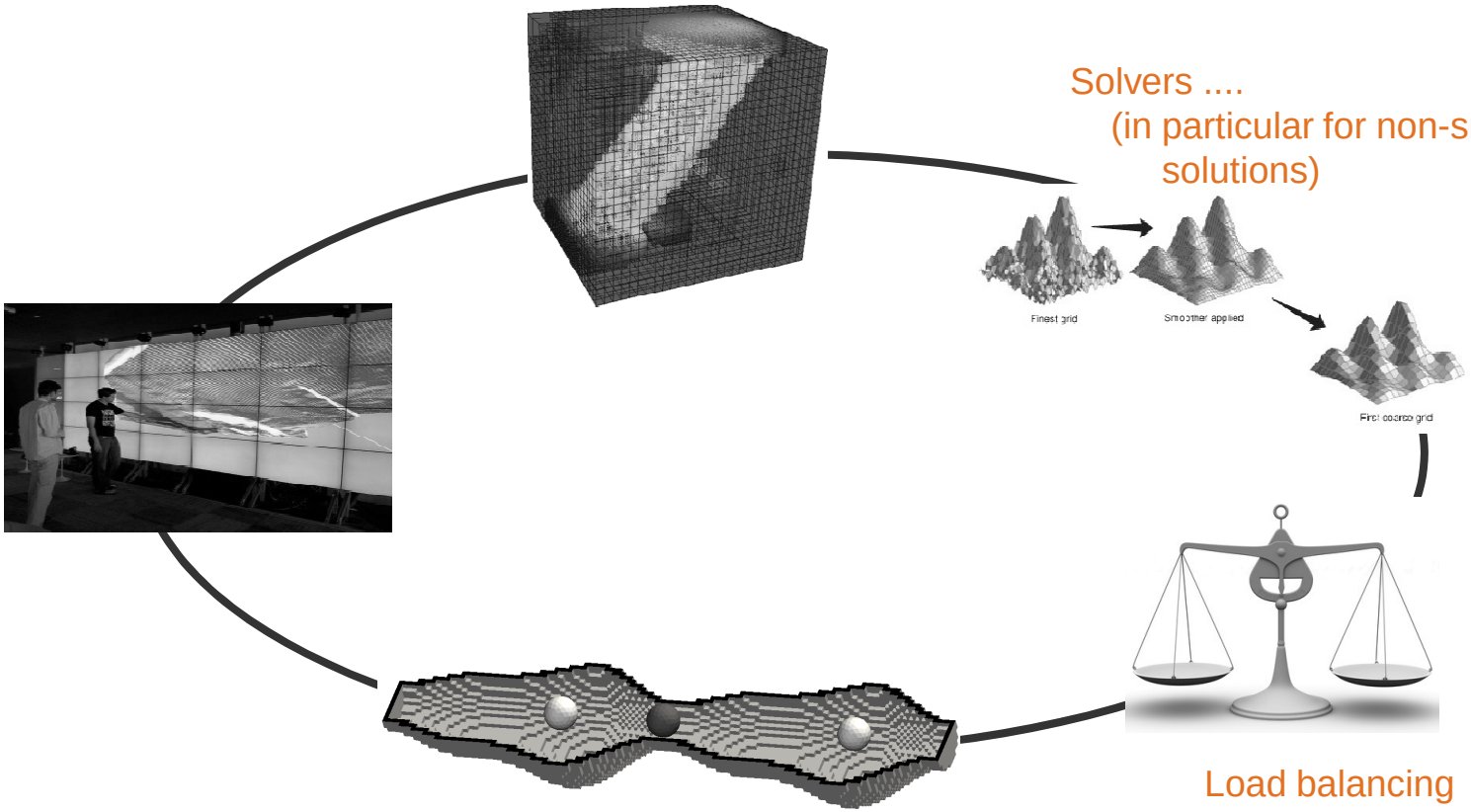
Dynamic adaptivity
&
local time stepping

Delivery of quick guesses
(for computational steering, e.g.)

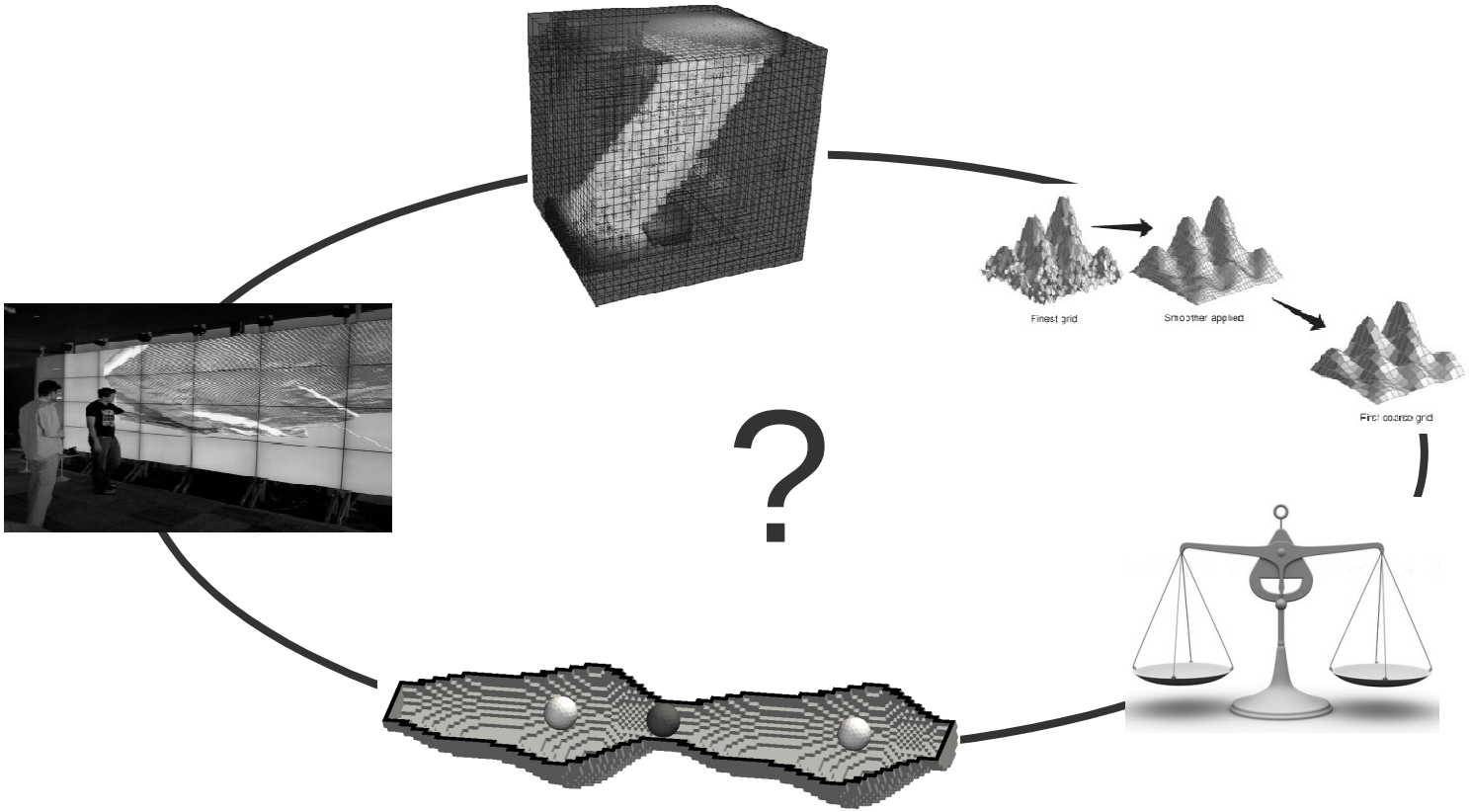


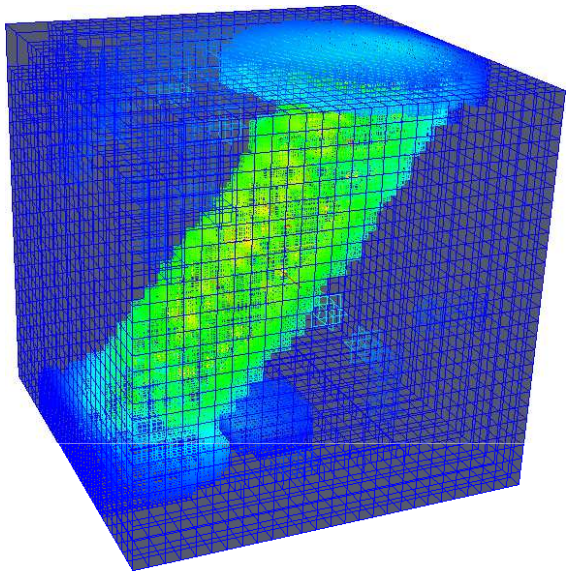
Steady cycle systems

Things that could work better



The application





All software is available at

www.peano-framework.org

Compare Different Solvers for One Time Step

- Classical time stepping scheme
- Different solvers
  - Implement pure Jacobi
  - Implement V(1,1), V(2,2), V(3,3) with **R=P**^{T} and P induced by ϕ(x)

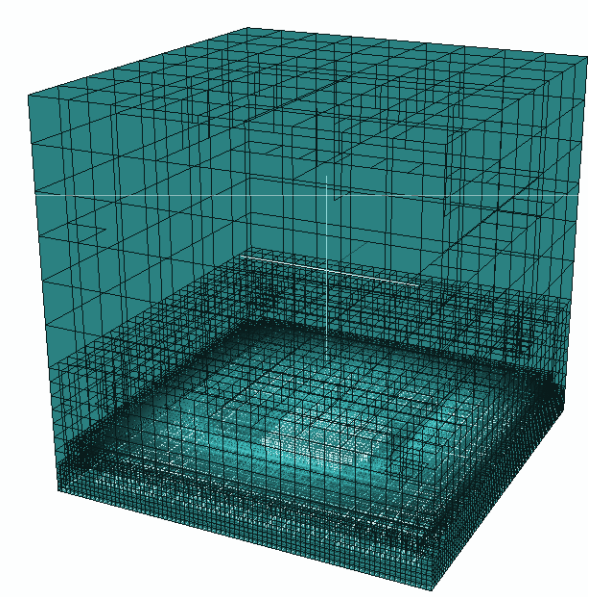
Stencil updates; fading sinus experiment; reduce error by 10 digits (*)

	Jacobi		$V(1,1)$-cycle		$FMG(1,1)$-cycle	
τ	$h=\frac{1}{3^4}$	$h=\frac{1}{3^5}$	$h=\frac{1}{3^4}$	$h=\frac{1}{3^5}$	$h=\frac{1}{3^4}$	$h=\frac{1}{3^5}$
$\frac{1}{10000}$	$3.13 \cdot 10^5$	$1.93 \cdot 10^7$	$4.13 \cdot 10^5$	$4.93 \cdot 10^6$	$7.55 \cdot 10^5$	$7.42 \cdot 10^6$
$\frac{1}{1000}$	$2.79 \cdot 10^6$	$1.88 \cdot 10^8$	$1.02 \cdot 10^6$	$9.86 \cdot 10^6$	$8.37 \cdot 10^6$	$7.42 \cdot 10^6$
$\frac{1}{500}$	$5.44 \cdot 10^6$	$3.68 \cdot 10^8$	$1.41 \cdot 10^6$	$1.35 \cdot 10^7$	$9.08 \cdot 10^6$	$7.42 \cdot 10^6$
$\frac{1}{400}$	$6.73 \cdot 10^6$	$4.55 \cdot 10^8$	$1.51 \cdot 10^6$	$1.35 \cdot 10^7$	$9.08 \cdot 10^6$	$7.42 \cdot 10^6$
$\frac{1}{300}$	$8.83 \cdot 10^6$	$5.86 \cdot 10^8$	$1.61 \cdot 10^6$	$1.46 \cdot 10^7$	$9.08 \cdot 10^6$	$7.42 \cdot 10^6$
$\frac{1}{200}$	$1.28 \cdot 10^7$	$8.69 \cdot 10^8$	$1.61 \cdot 10^6$	$1.46 \cdot 10^7$	$9.09 \cdot 10^6$	$7.42 \cdot 10^6$
$\frac{1}{100}$	$2.35 \cdot 10^7$	$9.99 \cdot 10^9$	$1.81 \cdot 10^6$	$1.57 \cdot 10^7$	$9.09 \cdot 10^6$	$7.42 \cdot 10^6$

(*) Results due to Tobias Köppl

Experiences with Global FMG I

- Cooling plate
  - Yields adaptive space-time grid
  - Compared to adaptive time stepping
    - with time stepping criterion (analyse $|u(t+\tau)-u(t+\tau/2|$) -> bigger and bigger time steps
    - with coarsening criterion -> decreasing no of grid points
    - with stopping criterion for equation system solver -> decreasing iteration numbers
  - $h = \tau = 27^{-1}$: 4.4
  - $h = \tau = 81^{-1}$: 6.8
  - $h = \tau = 243^{-1}$: ?
- Rotating heat source



Experiences with Global FMG II

- Cooling plate
- Rotating heat source
  - Yields adaptive space-time grid
  - Compared to adaptive time stepping
    - time stepping criterion -> time step size almost constant
    - with coarsening criterion -> total number of vertices per time slice remains almost constant
    - with stopping criterion for equation system solver -> same no of iterations for each time slice
  - $h = \tau = 27^{-1}$: 7.5
  - $h = \tau = 81^{-1}$: 3.8
  - $h = \tau = 243^{-1}$: ?