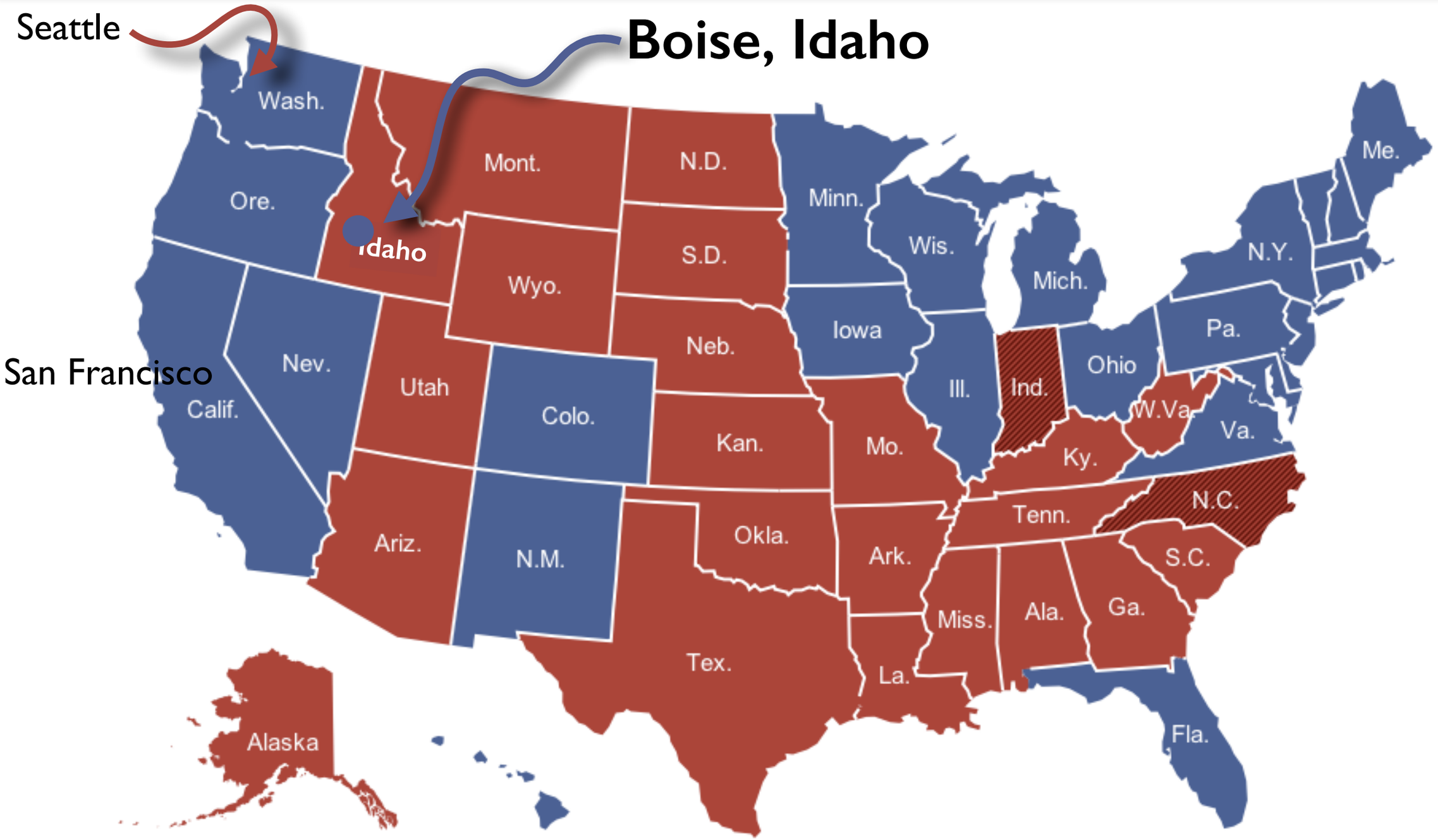
Geo-ForestClaw* : Modeling dam-break flooding using scalable, adaptive quad trees

Donna Calhoun (Boise State University)

Carsten Burstedde (Univ. of Bonn, Germany)
Melody Shih (Columbia/BSU); Kyle Mandli (Columbia Univ.); Ram Sampath (Centroid Lab); Steve Prescott (Idaho National Labs)
David George (USGS), Marsha Berger (NYU) Randall LeVeque (Univ. of Washington), Kyle Mandli (Columbia), Melody Shih (Columbia), David Ketcheson (KAUST, Saudi Arabia)

Clifford Lectures
April 5-7, 2-17
Tulane, New Orleans LA

Where is Boise?



What is in Idaho?

More about Idaho

BOISE STATE™

Albertsons®

Micron®

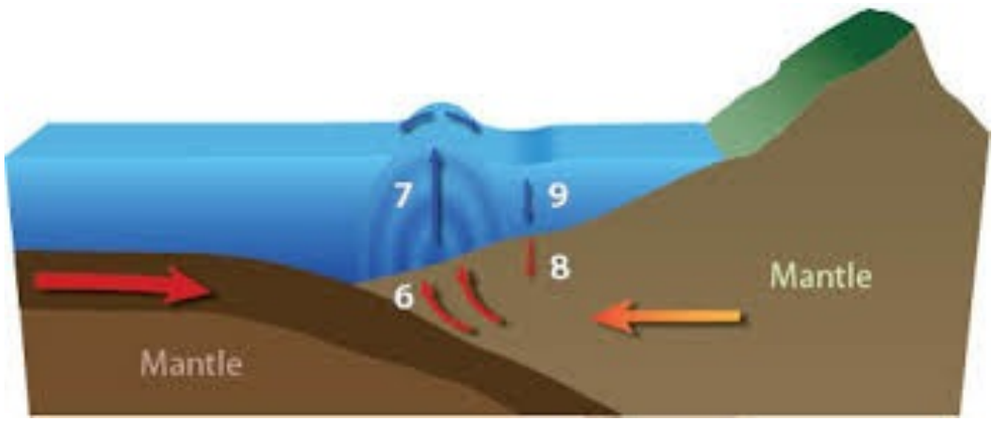
Boise Cascade
Engineered Wood Products

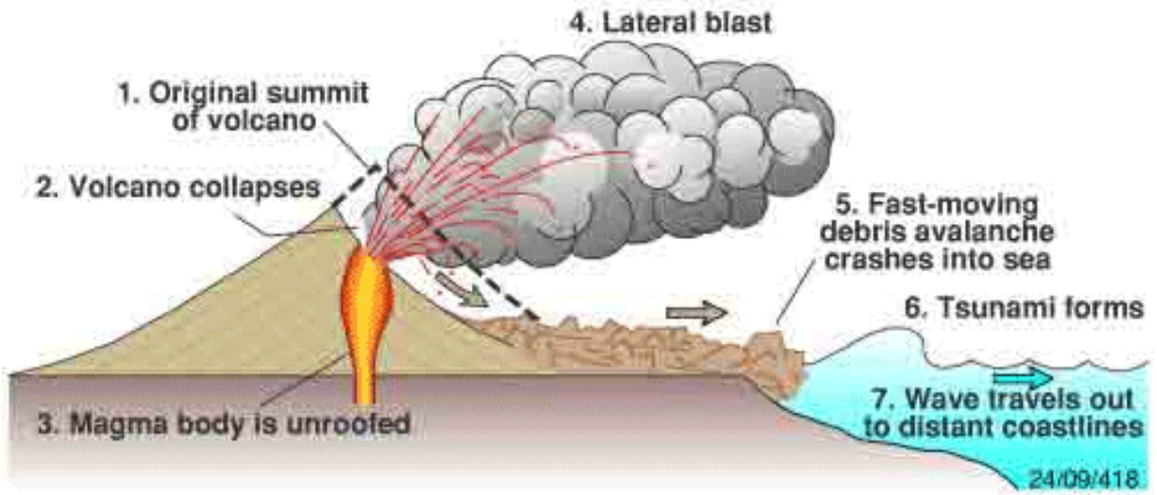
SALMON AIR

NATIONAL INTERAGENCY FIRE CENTER
Boise, Idaho

Challenges in Geophysical Hazards Modeling

- Complicated geometry that is not well known or understood.
- Coupled events requiring several different models
- Many temporal and spatial scales are involved
- Simulations should be done in real time to be most useful
- Small scales must be tracked for a long time over long distances
- Domains must be large enough to allow for numerical modeling of the entire event.

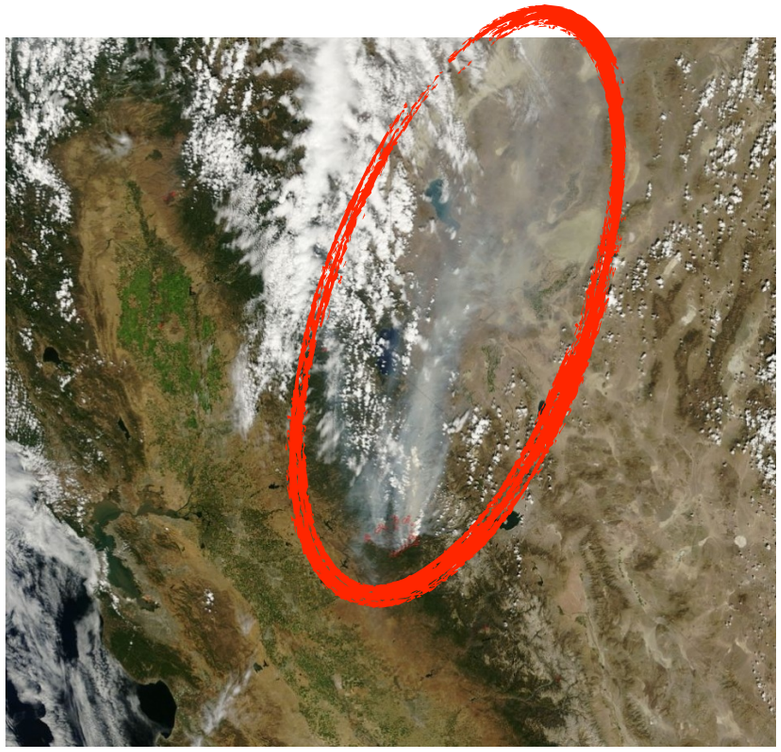




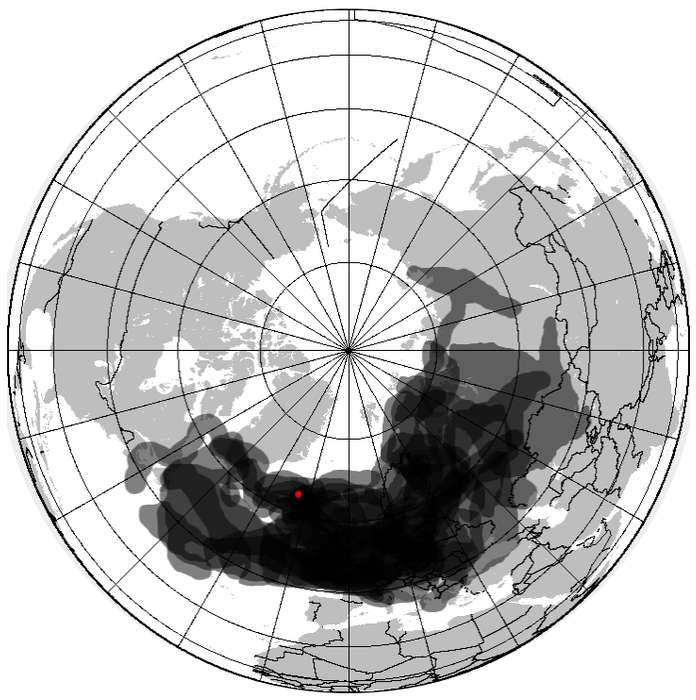
Challenges

- Small scales must be tracked for a long time over long distances
- Domains must be large enough to allow for numerical modeling of the entire event.

Computations can become very expensive without dynamic, multi-resolution capabilities.

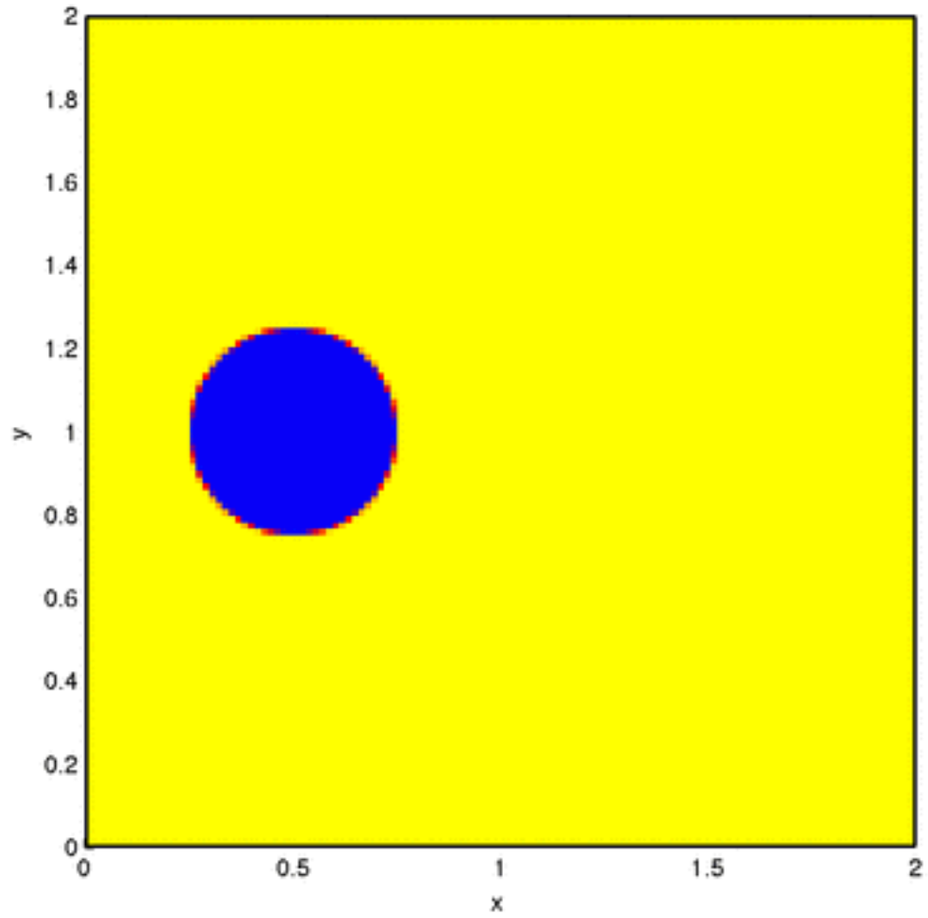


Smoke from the California Rim Fire, Sept 2013



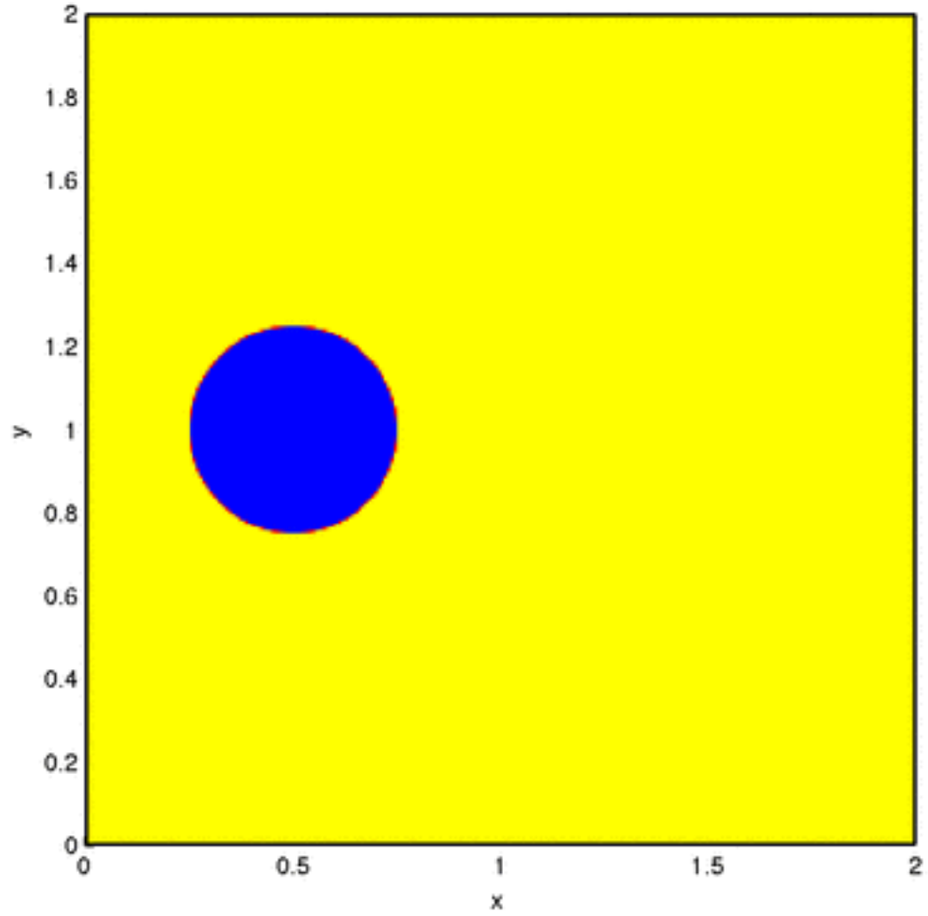
2010 Eruption in Iceland

Tracer transport



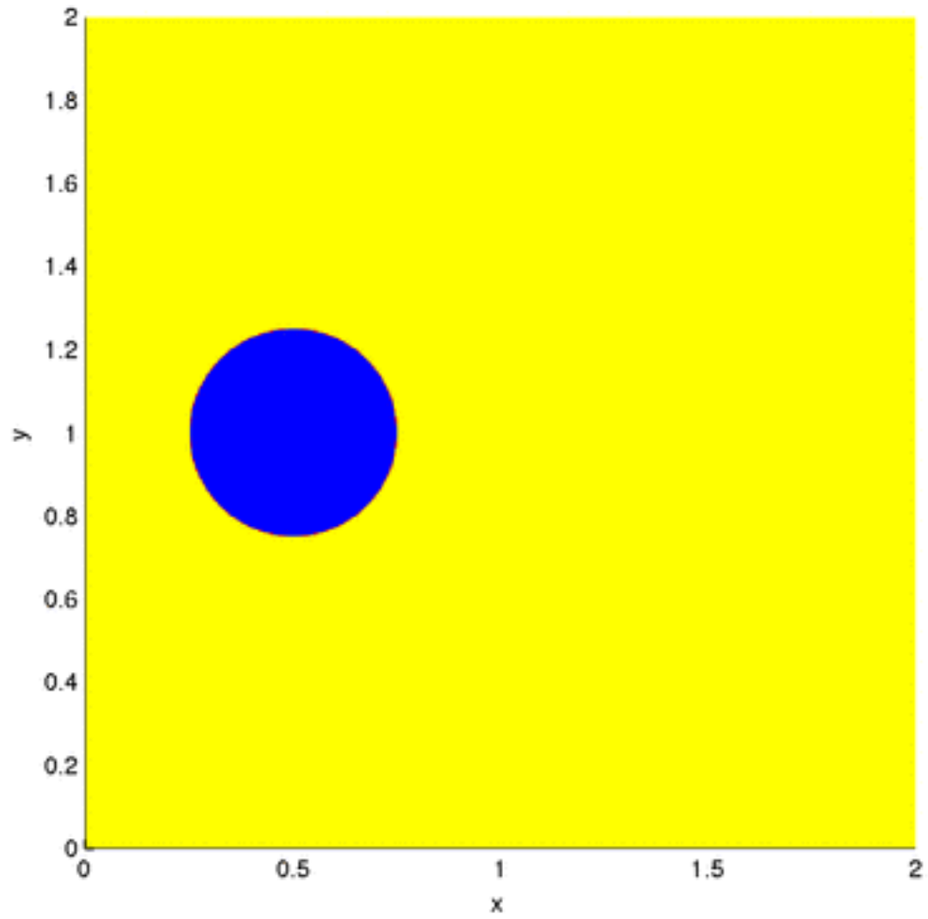
128 x 128 grid (9.3s)

Tracer transport



256 x 256 grid (70.2s)

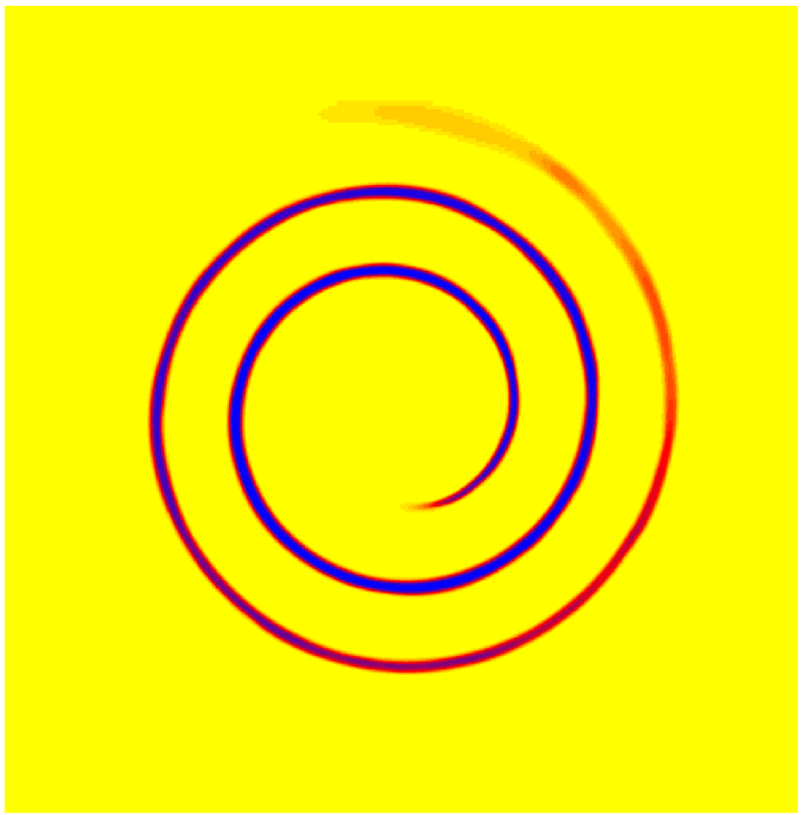
Tracer transport



512 x 512 grid (917s)

Accurate, but $$$

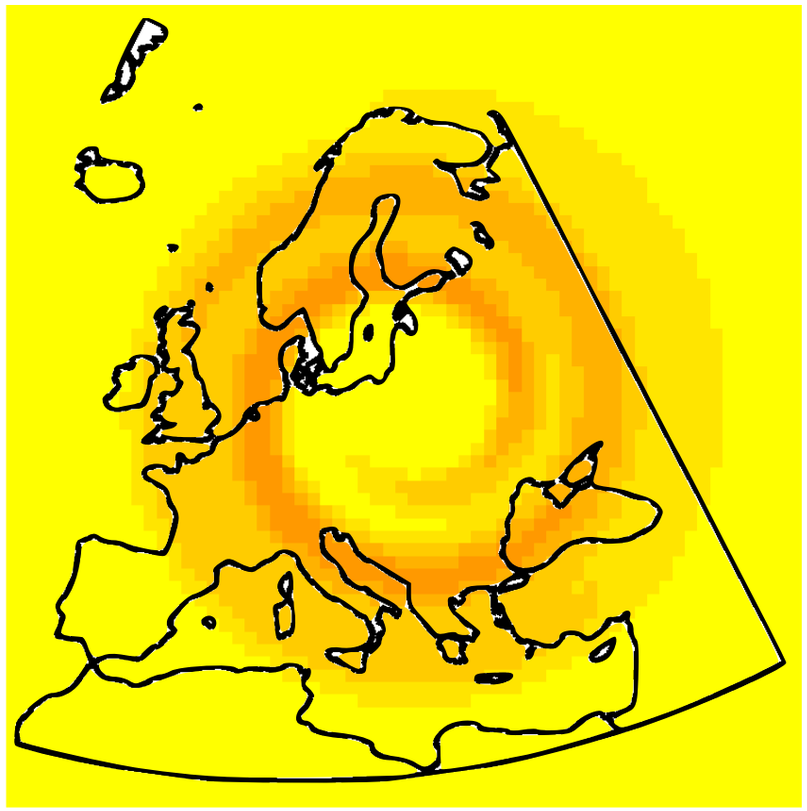
Tracer transport



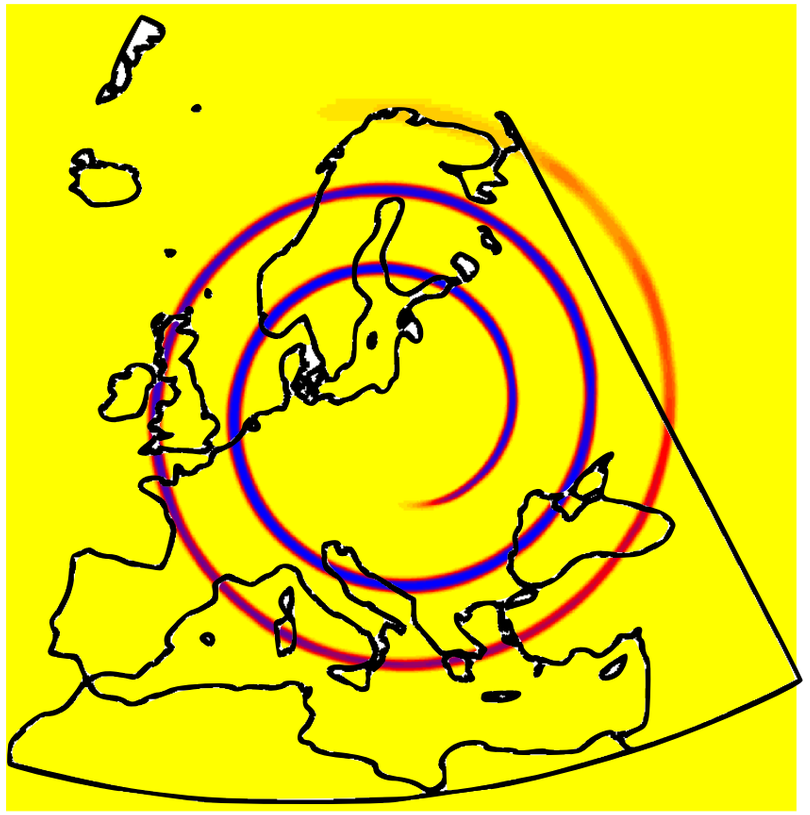
1024 x 1024 grid (9194.5s)

Accurate, but $$$

Tracer transport



64 x 64 (~550 km resolution)

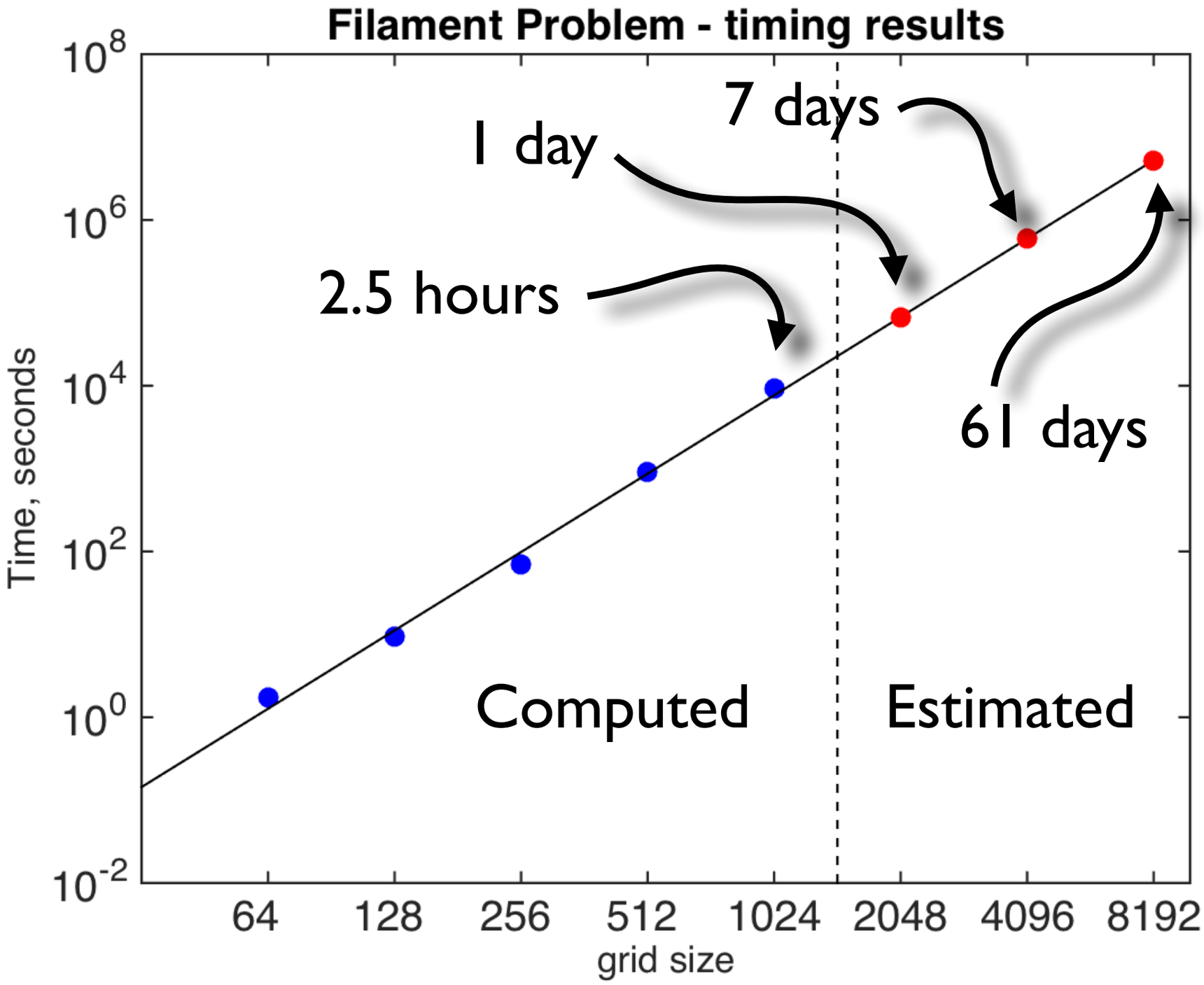


1024 x 1024 (~140 km resolution)

Tracer transport (smoke, volcanic ash, pollutants, …)

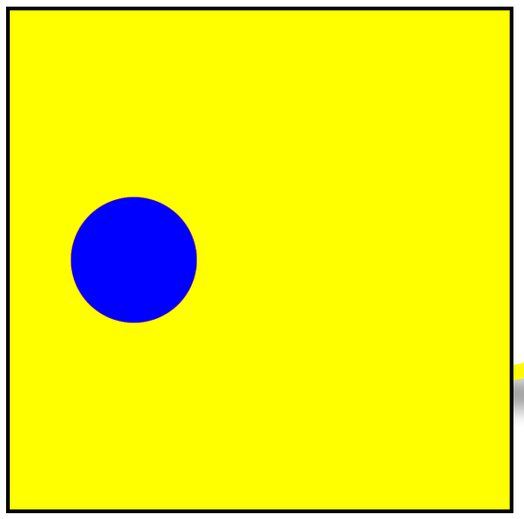
Behrens, J., Dethloff, K., Hiller, W., and Rinke, A. Evolution of small-scale filaments in an adaptive advection model for idealized tracer transport. Monthly Weather Review 128 (2000), 2976–2982.

Filament results

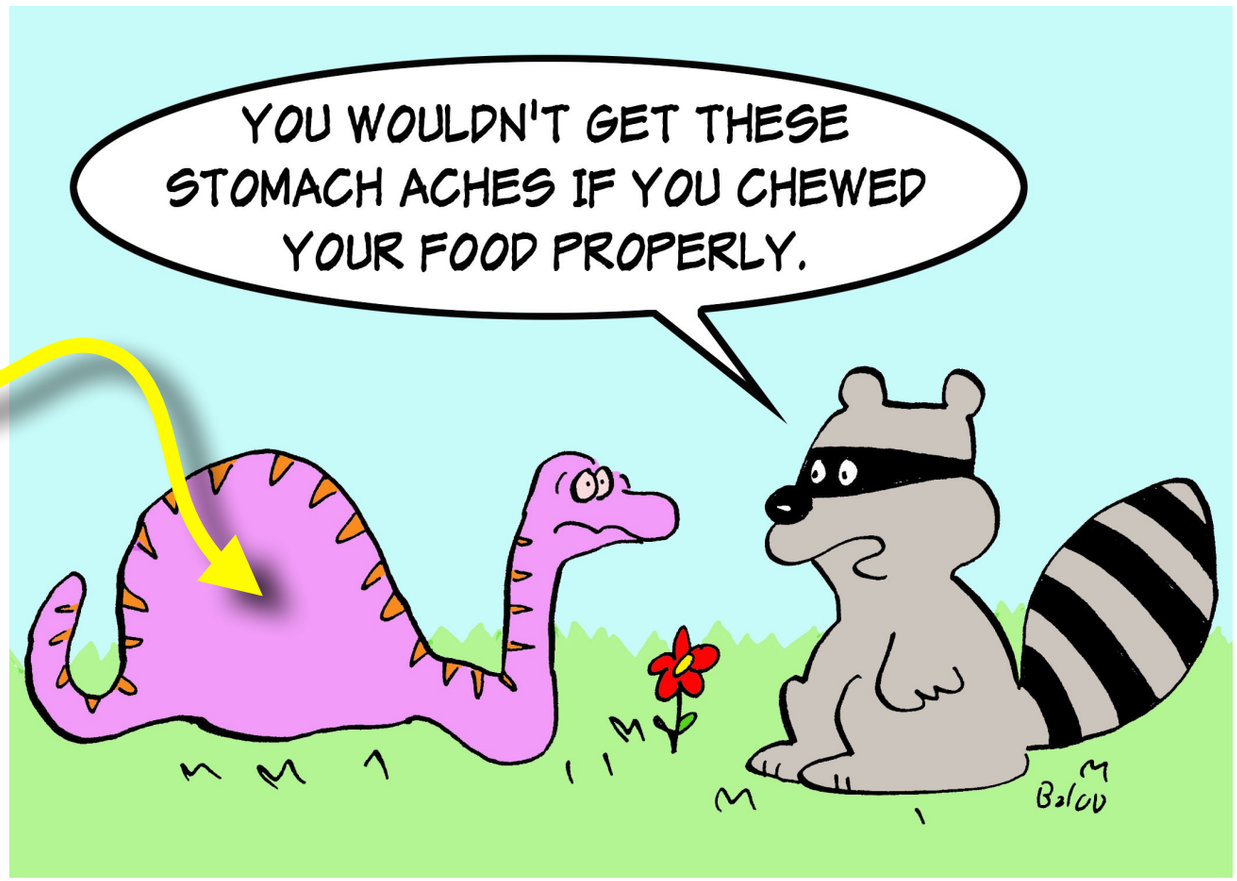


Solving on a uniform grid

About the worst thing you can do is allocate memory for a single large grid and let the system go to work on it.



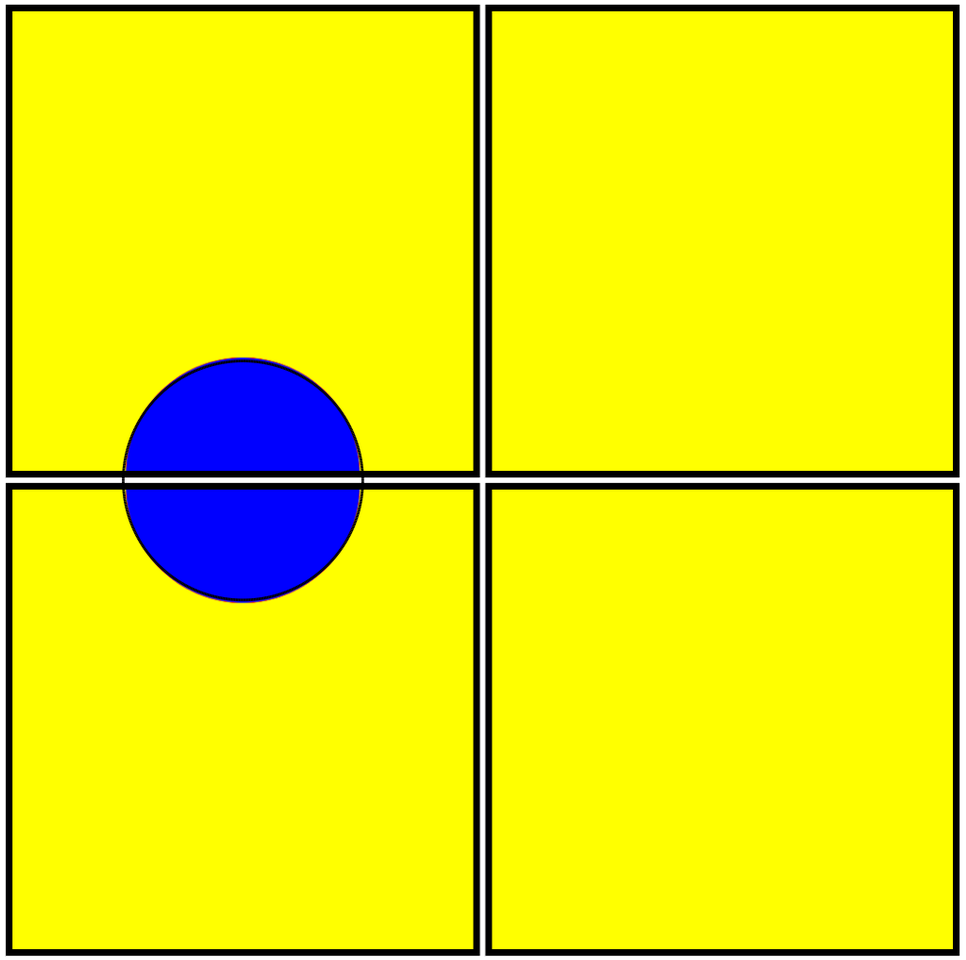
1024×1024



9194.5s (2.4hr)

Subdivide the domain

6997.1s (1.9hr)



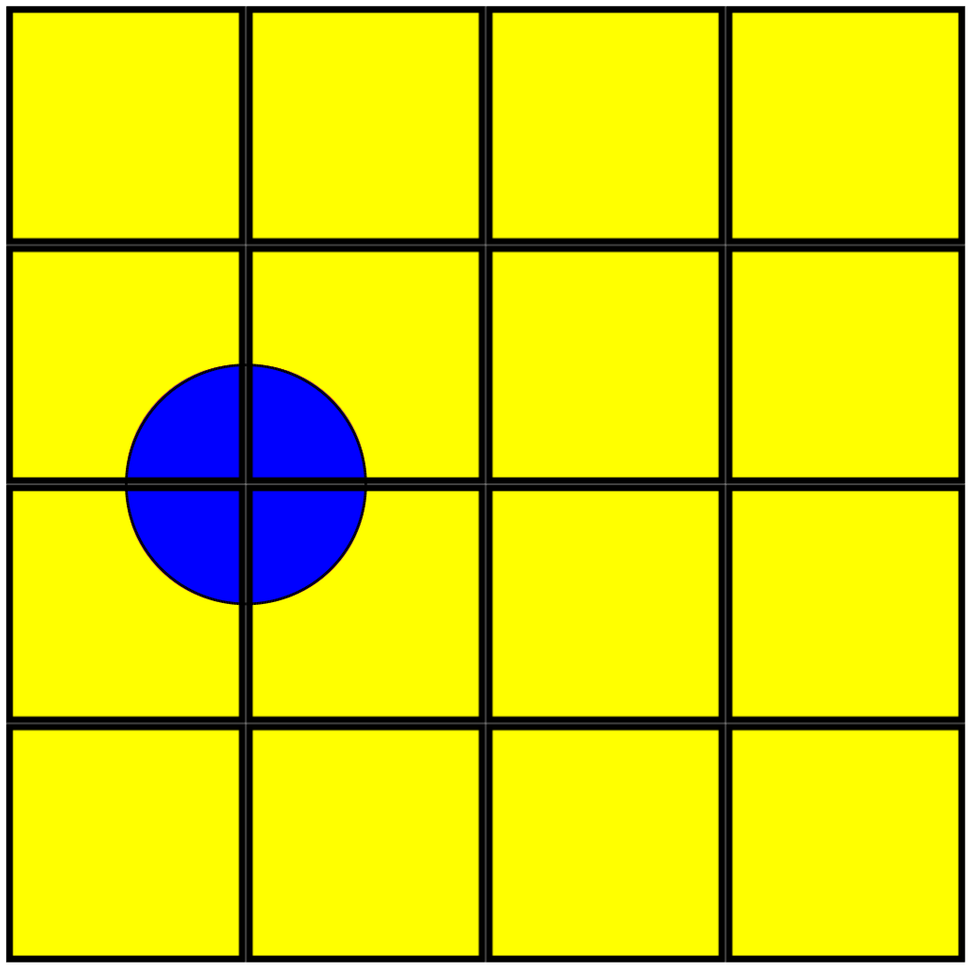
;

2×2 array of 512×512 grids

This will improve cache performance enormously

Subdivide the domain

4131.7s (69min)



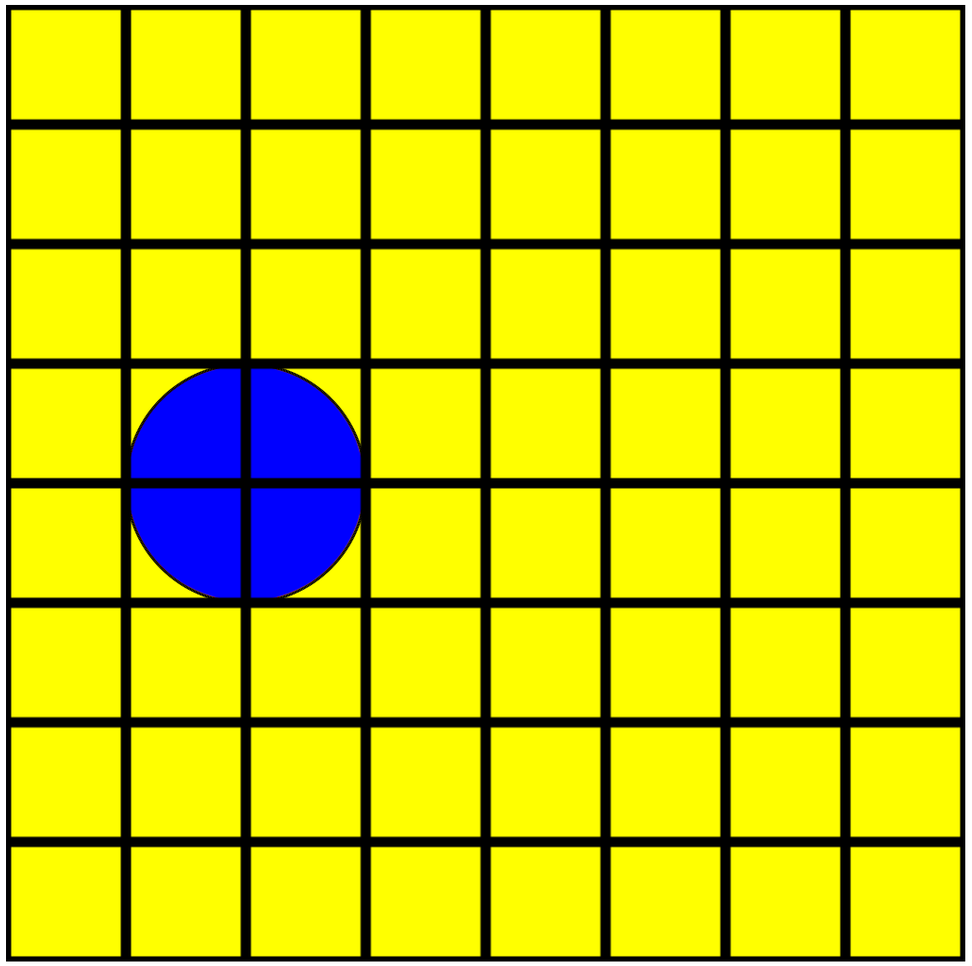
;

4×4 array of 256×256 grids

This will improve cache performance enormously

Subdivide the domain

3813.7s (64min)



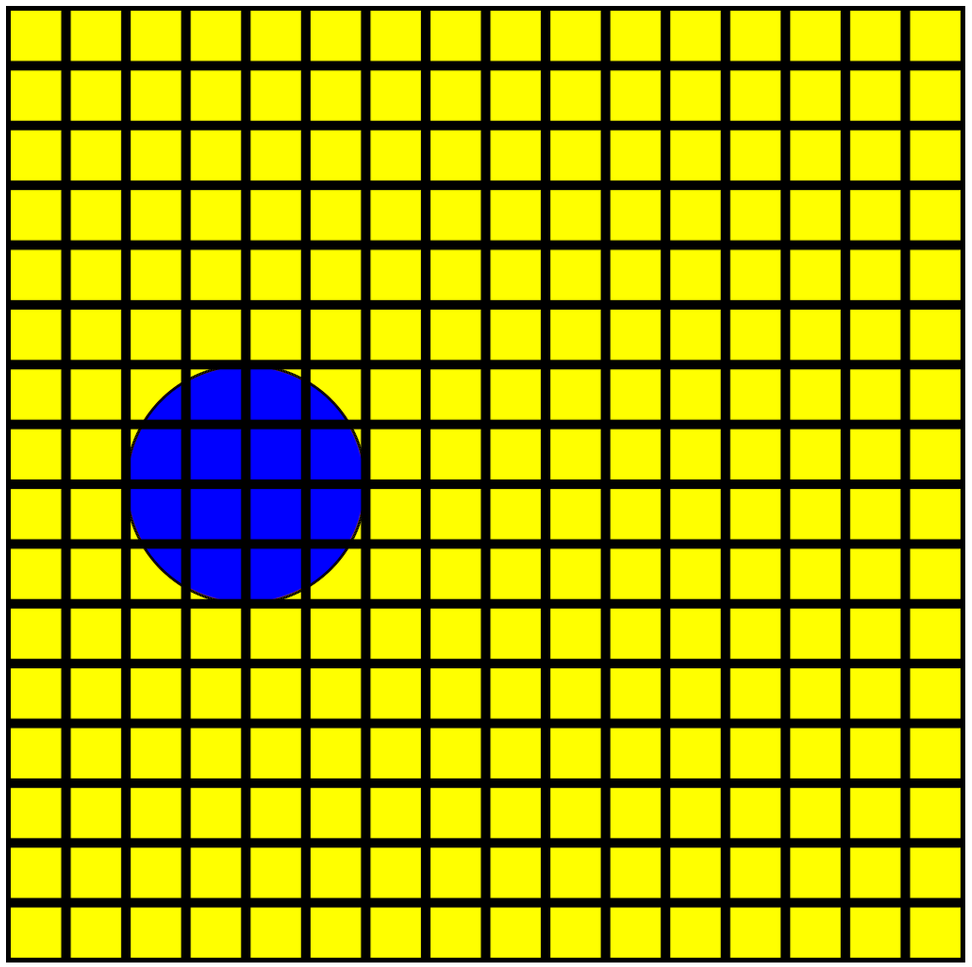
;

8×8 array of 128×128 grids

This will improve cache performance enormously

Subdivide the domain

3865.3s (64min)

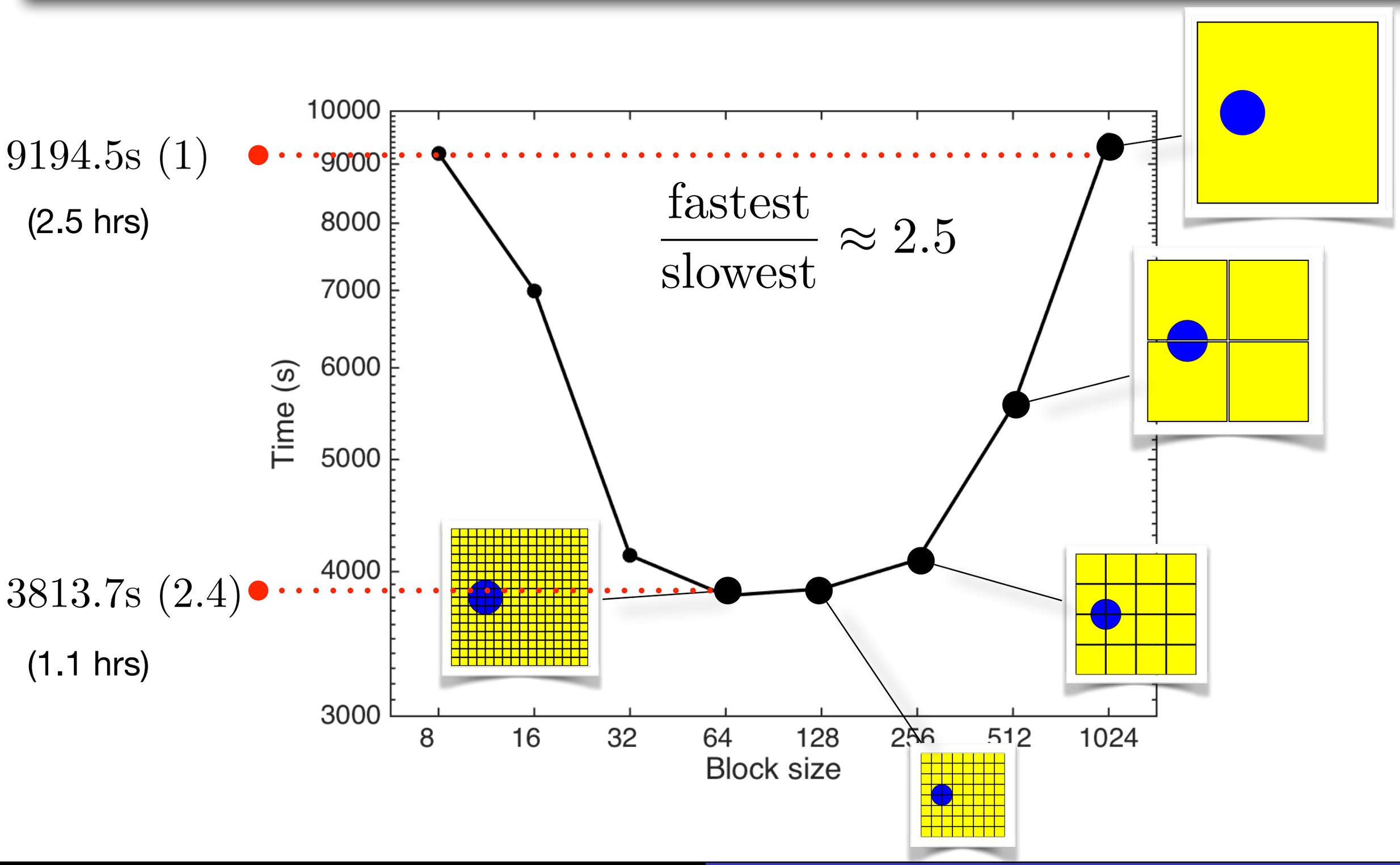


;

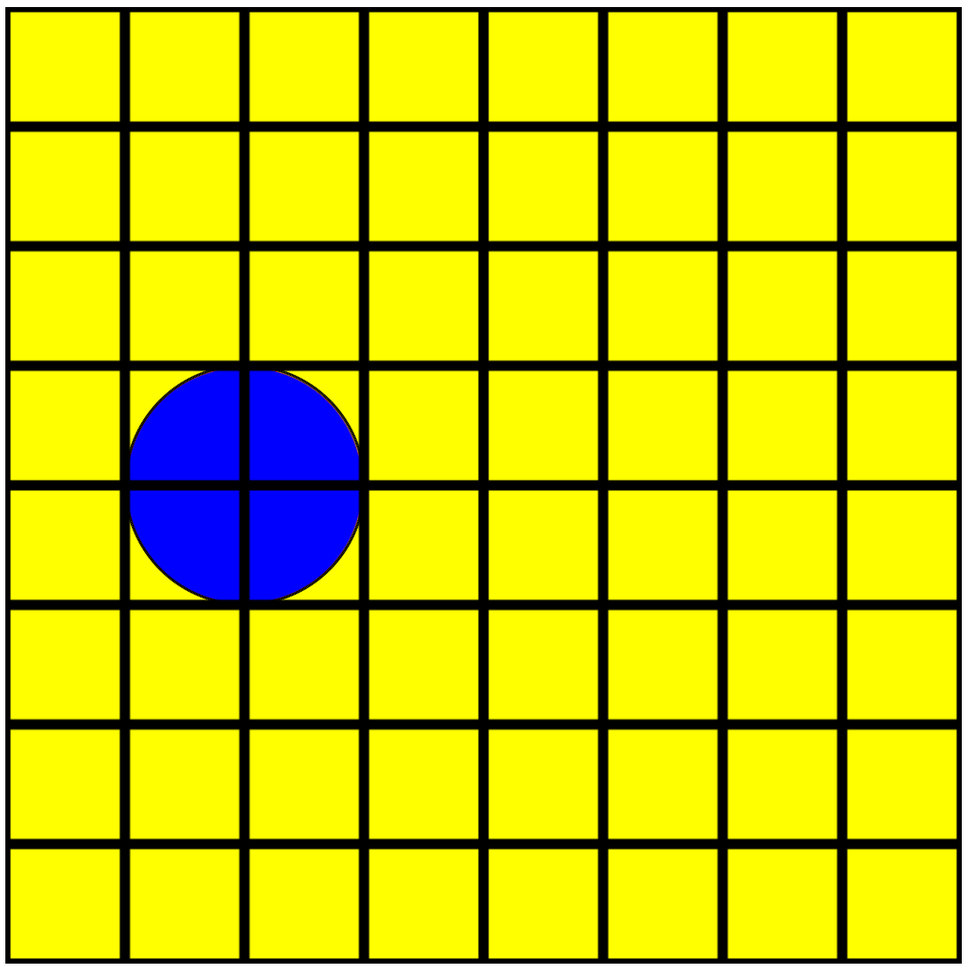
16×16 array of 64×64 grids

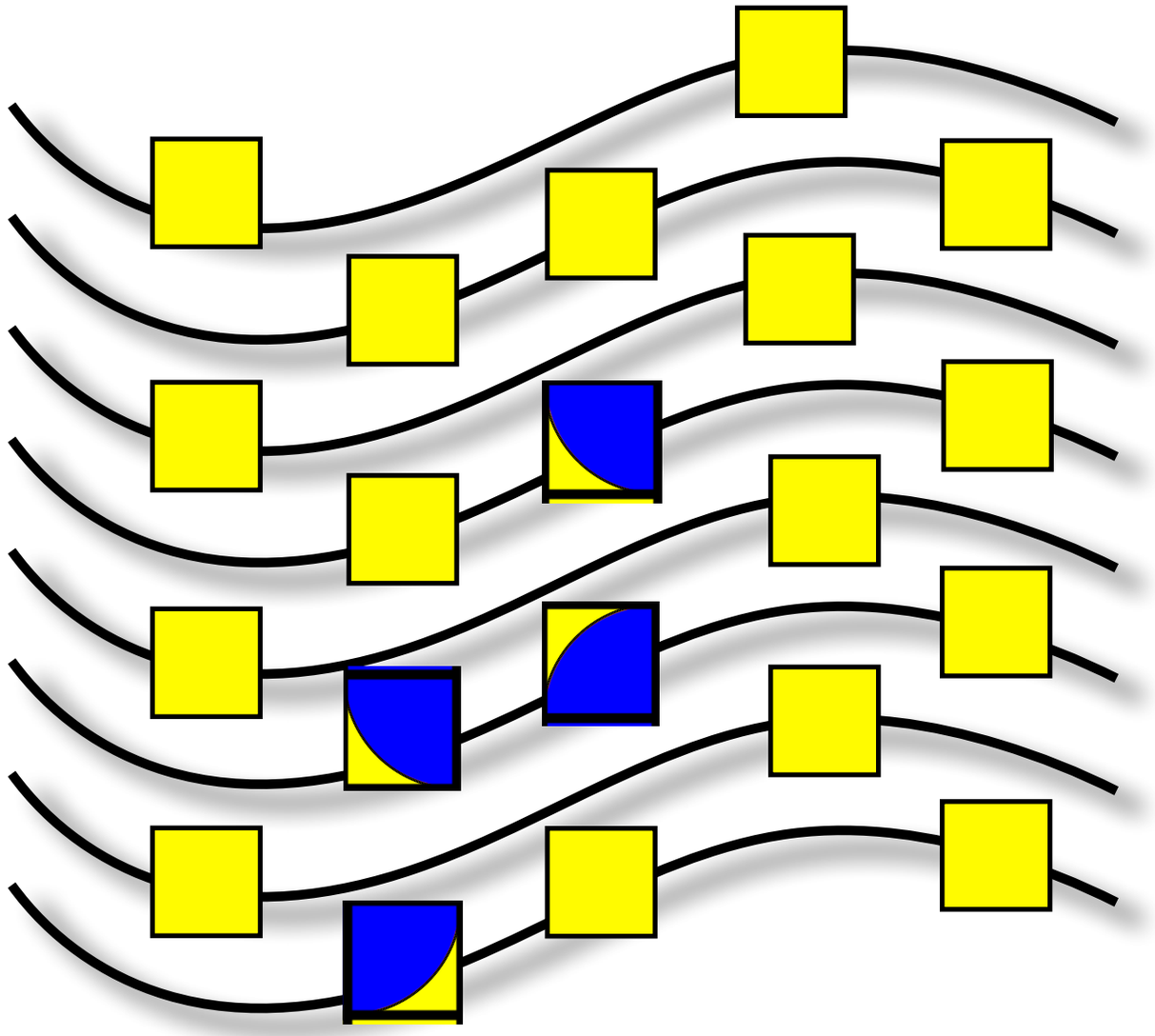
This will improve cache performance enormously

Timing vs block size



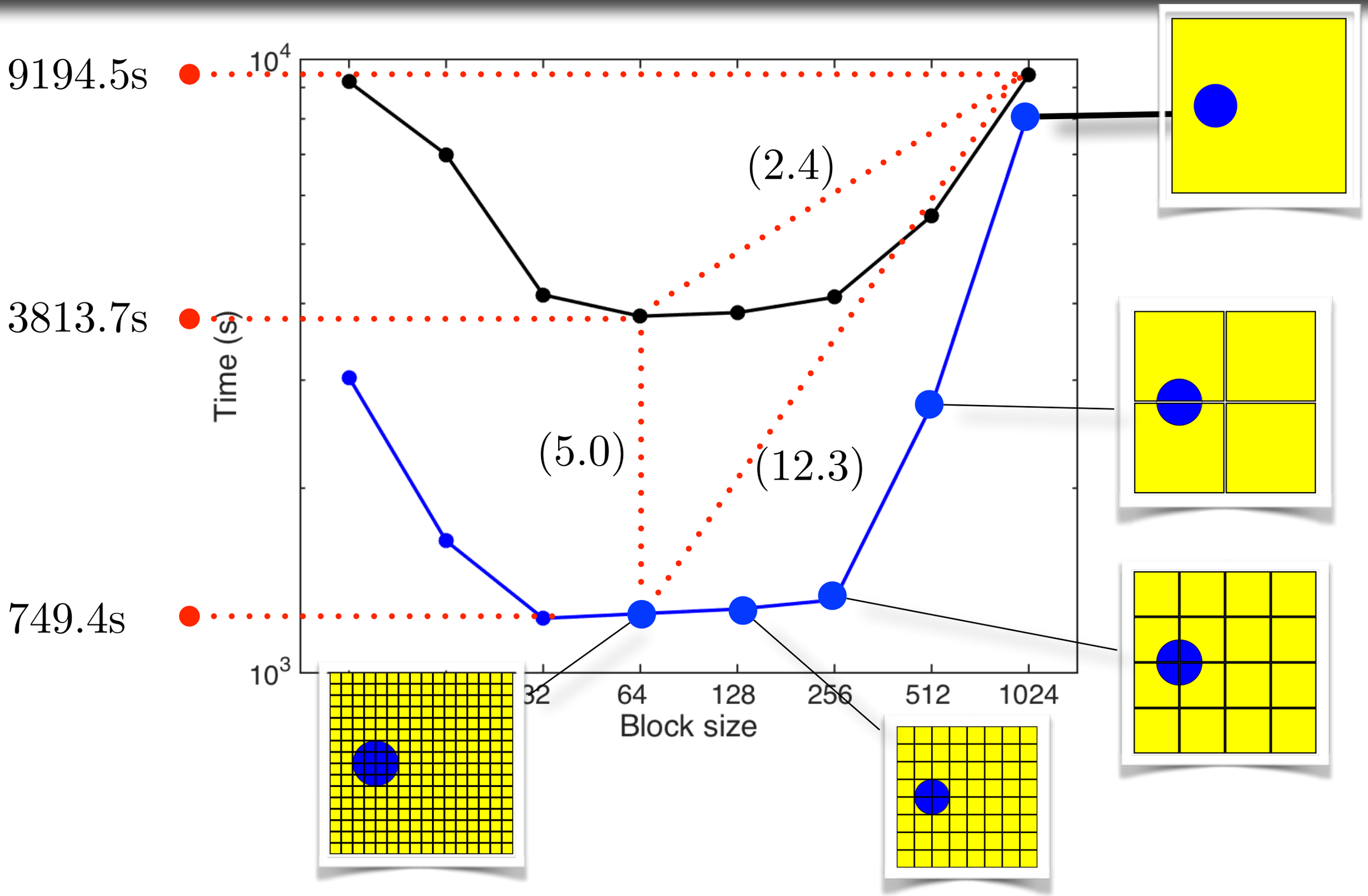
Use multiple cores





Distribute patches to parallel processing units (using OMP, MPI, CUDA, and so on). Ideally, speedup ~ number of cores used.

Speed up using 4 cores (Intel i7)



Can we improve on this?

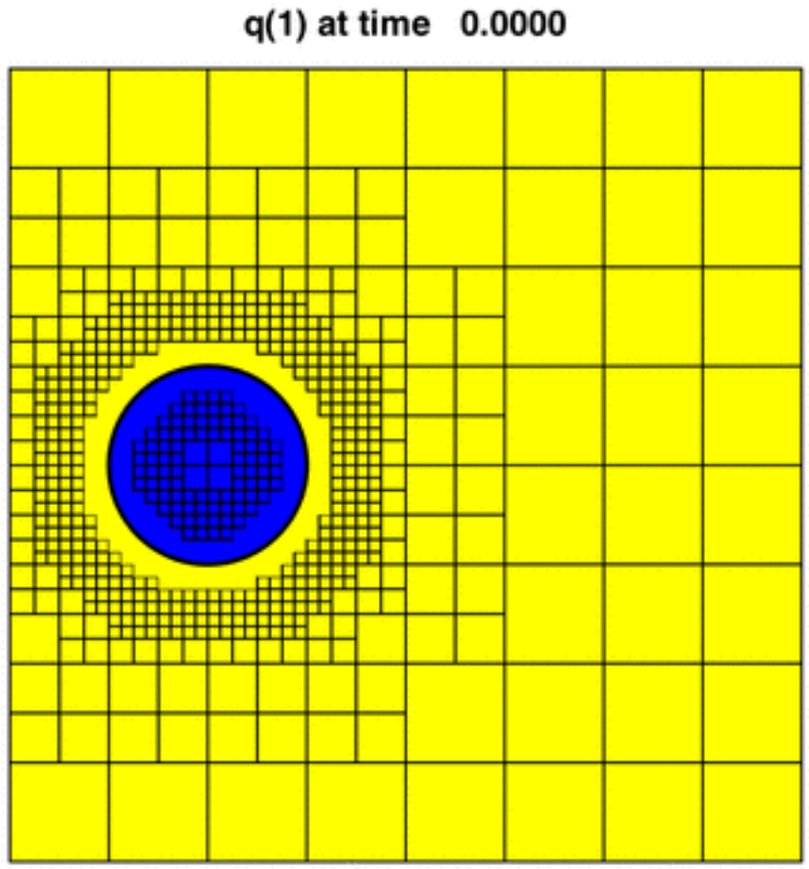
Hardware solutions : Use more computing units (cpus/threads/cores)

- Eventually, however, the communications costs will overwhelm any gains made in subdividing the domain further
- Problems always outgrow the hardware that is available

Software solutions : Reduce the “factor 8” scaling law
and put resources only where they are needed.

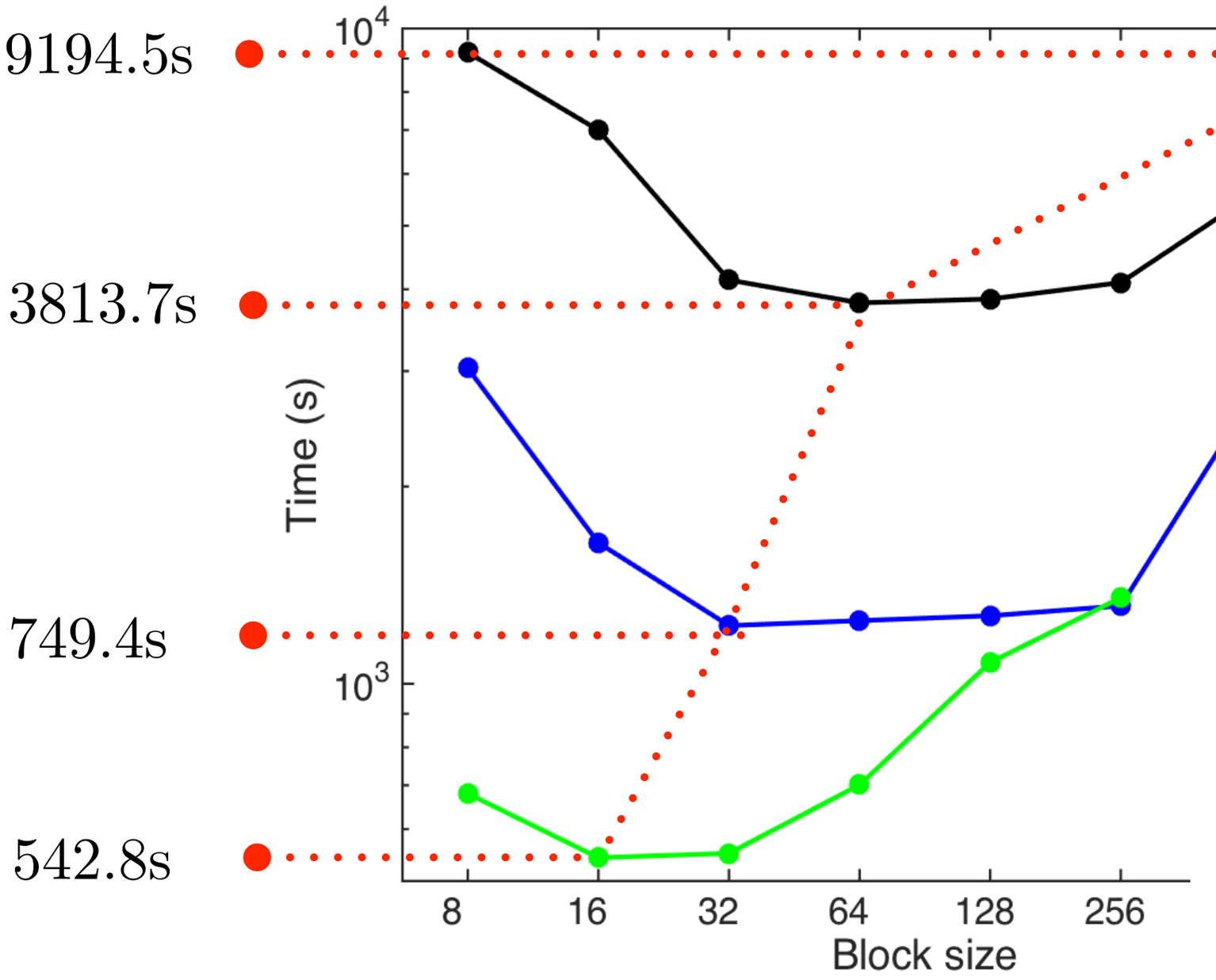
- This is much more difficult than just adding more computing units
- Should complement hardware advances

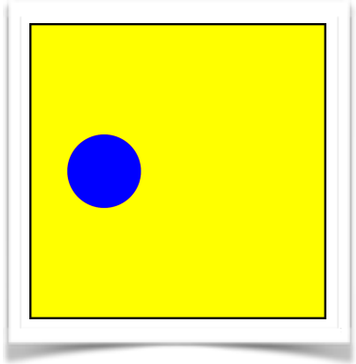
AMR simulation

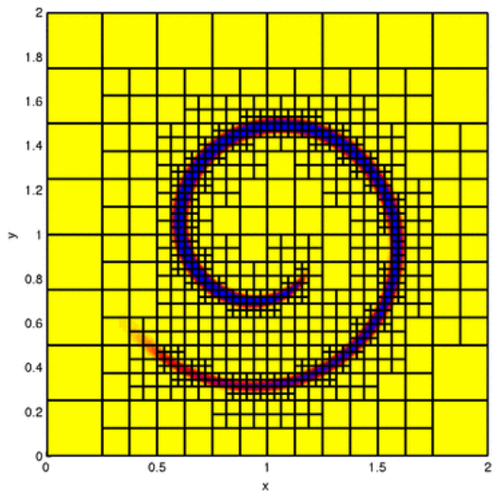


Effective 512 x 512 resolution (116s vs. 917s)

With adaptive mesh refinement







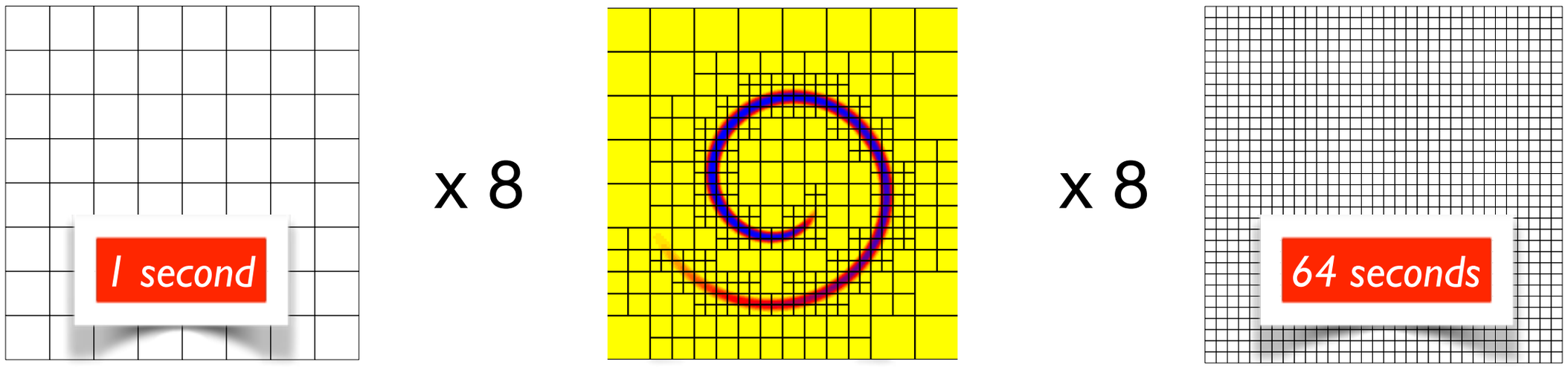
Over a factor of 15 improvement !

One more tweak …

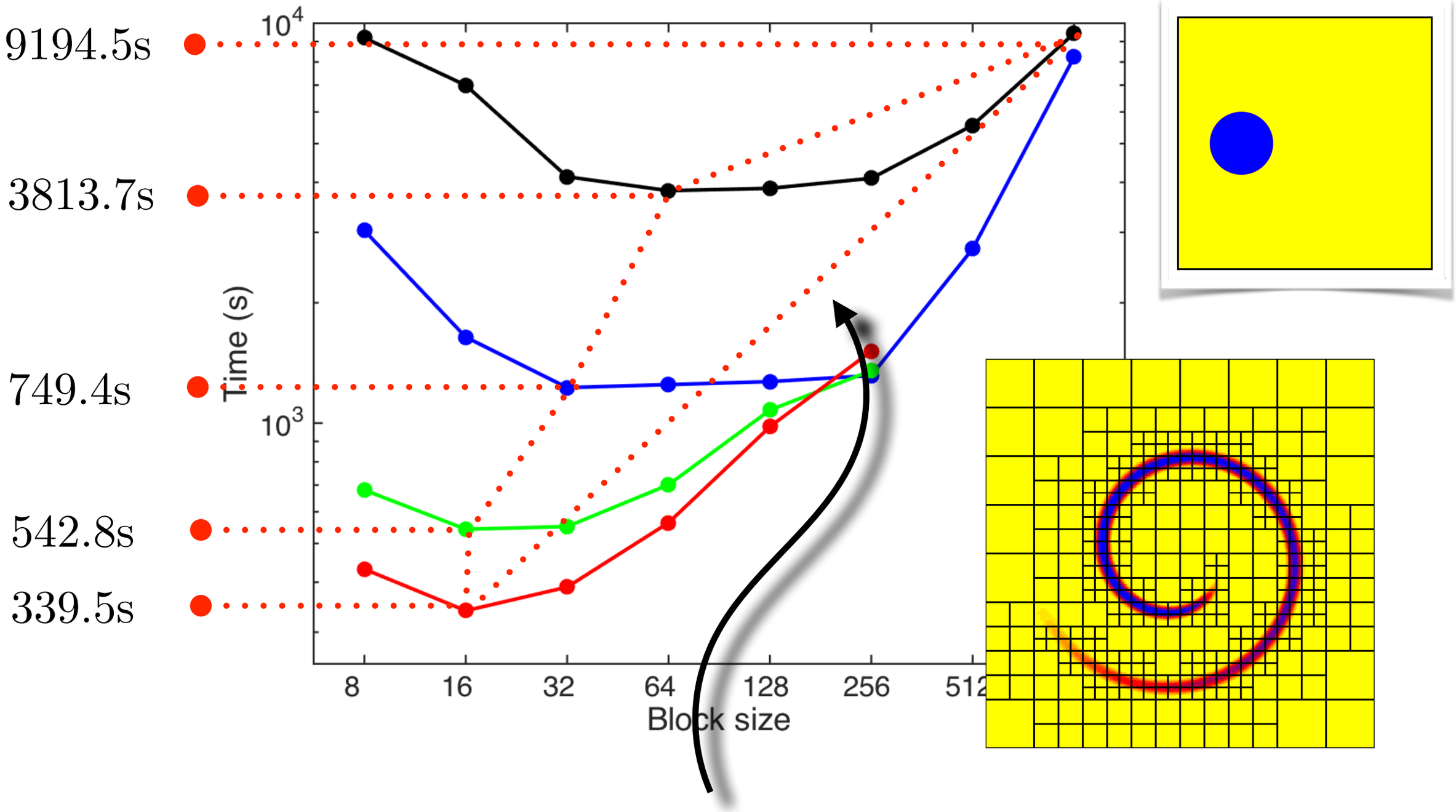
In two dimensions, if we double the resolution, we increase the computational cost by a factor of 8

(Number of grid cells increases by a factor of 4)×
(Number of time steps increase by a factor 2) =
(8-fold increase in computational expense)

The CFL condition constrains the size time step we can take on a grid. For numerical stability, smaller grid cells need smaller time steps.



Multi-rate time stepping

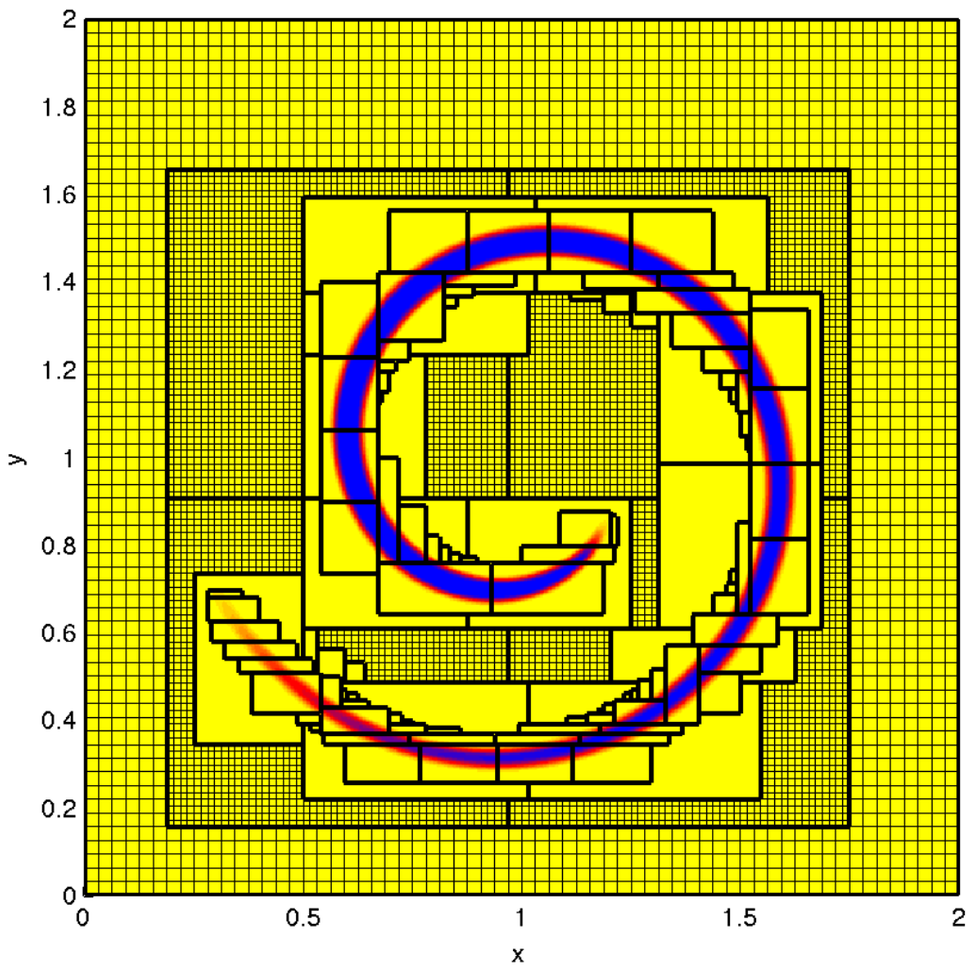


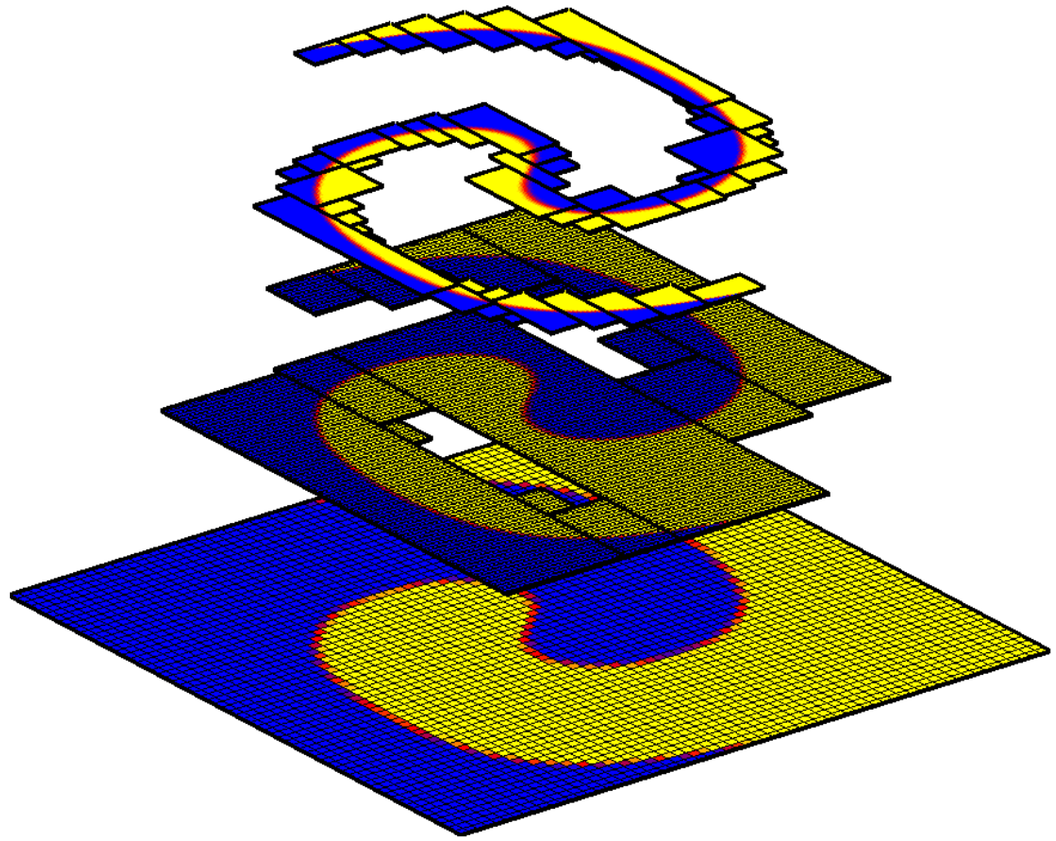
Almost a factor of 30 improvement !

Adaptive Mesh Refinement (AMR)

Overlapping patch-based AMR (Structured AMR or SAMR)

Original approach (Berger, 1984)



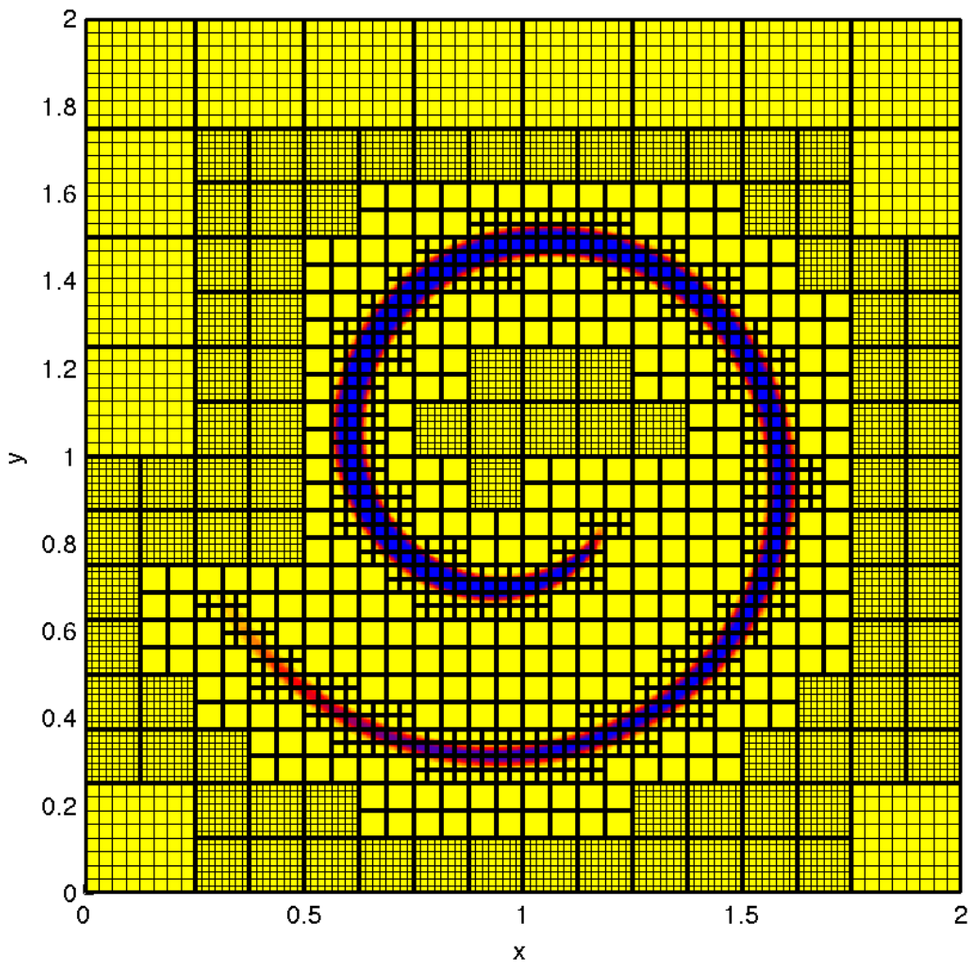


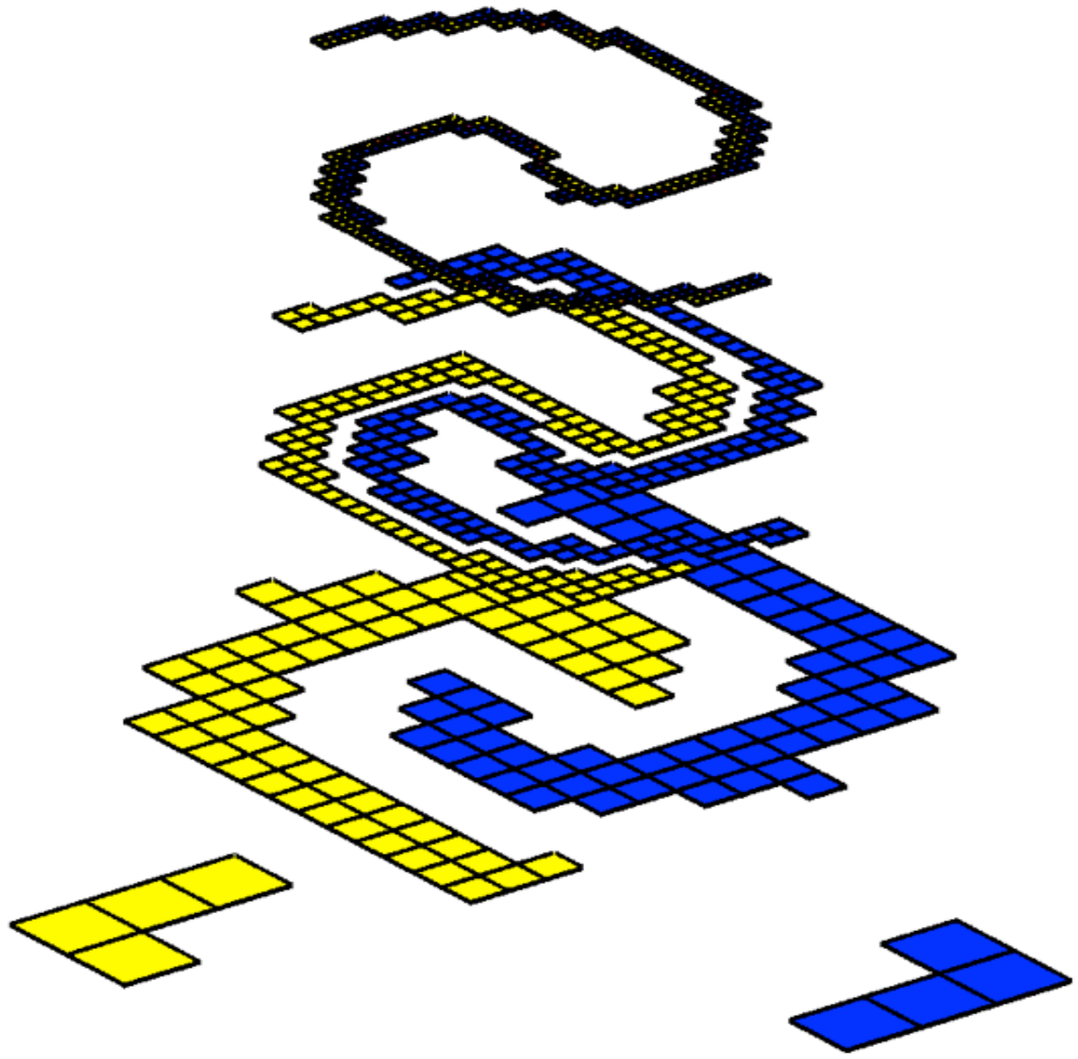
Codes : Chombo (LBL), AMRClaw and GeoClaw (UW, NYU) , **Boxlib* (LBL)**, SAMRAI (LLNL), AMROC (Univ. of South Hampton) and many others

Adaptive Mesh Refinement (AMR)

Quadtree/Octree based AMR

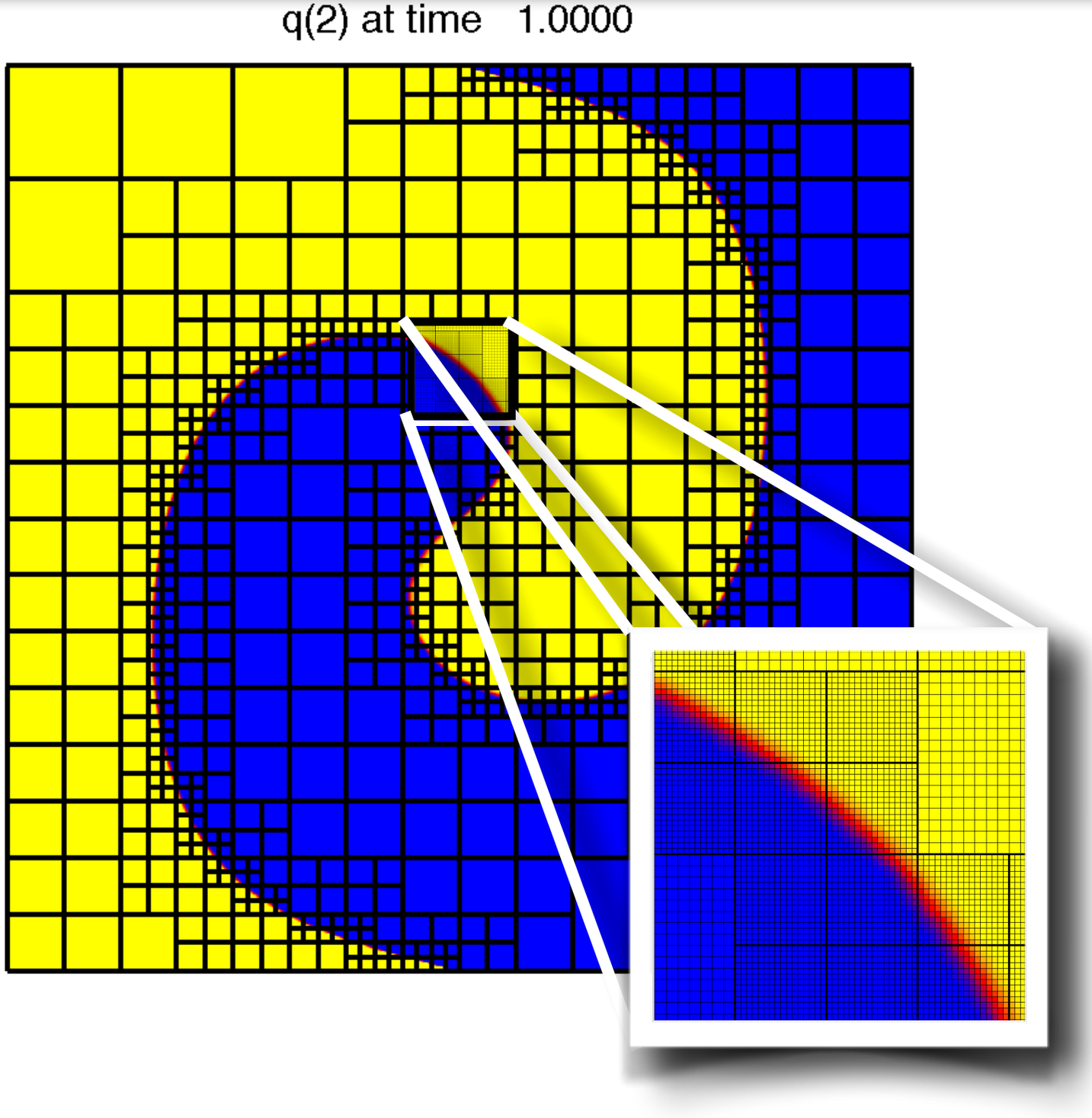
Quad-tree approach





p4est (U. Bonn), PARAMESH (U. Chicago), ForestClaw, Gerris (Paris VI), Racoon II (U. Bochum), RAMSES (U Zurich), Nirvana (Potsdam), "Building Cubes" (Tohoku)

AMR using ForestClaw



Brief history of AMR

Refinement based on quadtree and octree grid layouts

- 2000 : P. MacNiece, K. Olson et al, "**PARAMESH**: A parallel adaptive mesh refinement community toolkit" (FLASH code based on PARAMESH)
- 2002 : R. Tessyier, "Cosmology Hydrodynamics with adaptive mesh refinement. A new high resolution code called **RAMSES**" (Lausanne, Switzerland)
- 2003 : S. Popinet, "**Gerris**: A tree-based adaptive solver for the incompressible Euler equations in complex geometries" (Paris IV, France)
- 2004 : U. Ziegler, "An ADI-based adaptive mesh Poisson solver for the MHD code **NIRVANA**" (Potsdam, Germany)
- 2005 : J. Dreher and R. Grauer, "**Racoon**: A parallel mesh-adaptive framework for hyperbolic conservation laws" (Bochum, Germany)
- 2011 : C. Burstedde, L. Wilcox, O. Ghattas, "**p4est**: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees" (Univ. Texas)
- 2011 : K. Komatsu, T. Soga et al "Parallel processing of the **Building-Cube Method** on a GPU platform" (Tohoku, Japan)

2000 ⟵ ⟶ present

ForestClaw Project

A parallel, adaptive library for logically Cartesian, mapped, multi-block domains

Features of ForestClaw include :

- Uses the **highly scalable p4est** dynamic grid management library (C. Burstedde, Univ. of Bonn, Germany)
- Each leaf of the quadtree contains a fixed, uniform grid,
- Optional multi-rate time stepping strategy,
- Has **mapped, multi-block** capabilities, (cubed-sphere, for example) to allow for flexibility in physical domains,
- Modular design gives user flexibility in extending ForestClaw with Cartesian grid based solvers and packages.
- Uses essentially the same algorithmic components as patch-based AMR

ForestClaw development supported by the National Science Foundation

www.forestclaw.org

Improving computations

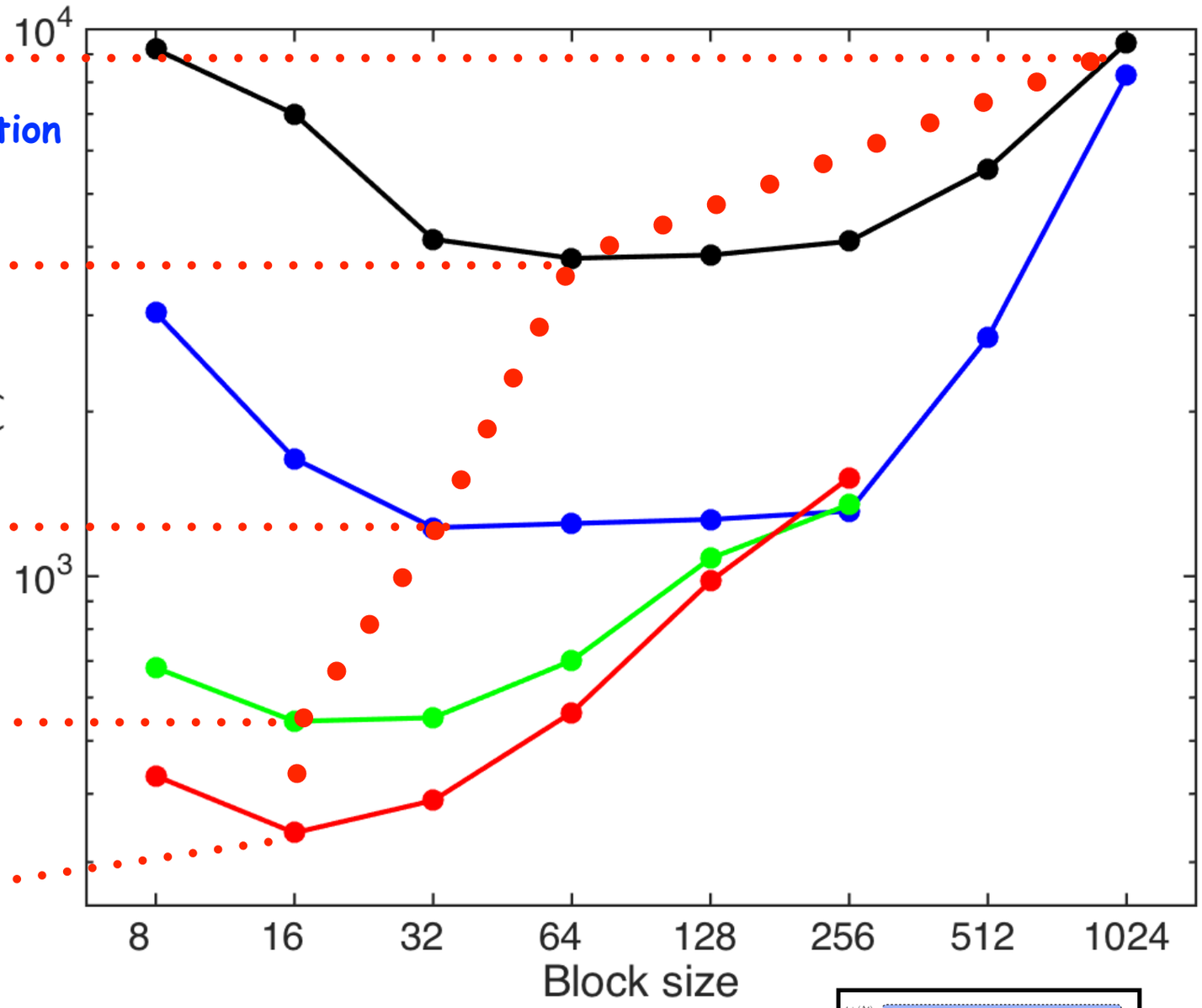
~2.5 h
Single grid calculation

~1 h
Uniformly refined

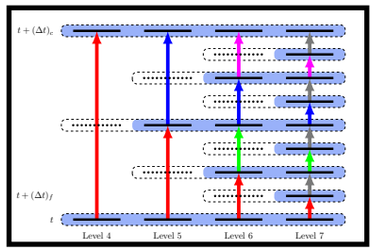
~12.5 min
Parallelism

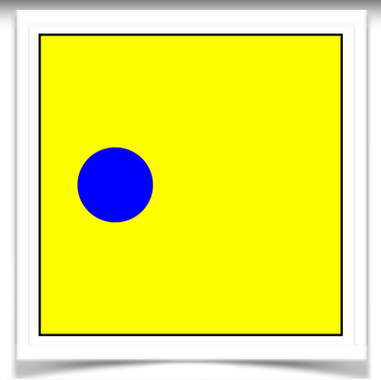
~9 min
Spatial adaptivity

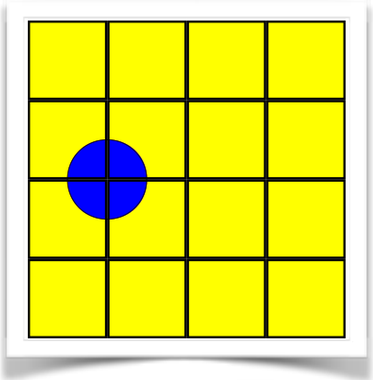
~5.5 min
Adaptive time stepping

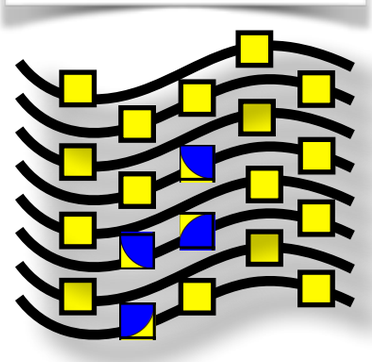


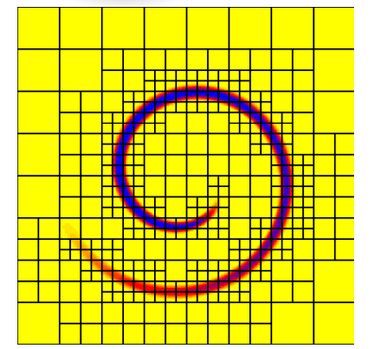
Almost 30 times improvement











ForestClaw

ForestClaw supports these modes for improving computations :

- **Domain decomposition** for better cache performance.
- **Parallelism**. Space-filling curves allow for load balancing and non-square parallel regions (e. g. easy to run on 17 processors)
- **Dynamic spatial adaptivity** for following solution features of interest.
- **Adaptive time stepping** to reduce communication between grids by reducing number of ghost cell communications, and reduce number of grids that need to be updated.

Optimizations below the patch level are not yet handled

- Loop optimizations (beyond what Clawpack/Geoclaw already do)
- Acceleration using GPUs, MICs, FPGAs and so on,
- Blocking within patches to reduce ghost cell communication

ForestClaw Solvers (so far)

p4est

ForestClaw

clawpack4.6

clawpack5

geoclaw

Tasks (tagging, building patches, etc) customized through use of function pointers stored in "virtual tables"

ForestClaw is mostly C; original solvers libraries left largely untouched (in original Fortran)

Time stepping, dynamic grid management, input/ output.

Patches

Solvers

GeoClaw Extension of ForestClaw

What stays the same

- Geo-ForestClaw uses all of existing Riemann solvers in GeoClaw, along with all bathymetry handling routines.

What had to be modified

- ForestClaw requires "coarsening" criteria, since we don't store underlying coarse grid meshes
- Gauges had to re-implemented for the quadtree mesh, but this leveraged the underlying **p4est** fast search algorithms.
- Customized averaging and interpolation routines to take into account bathymetry

Problems?

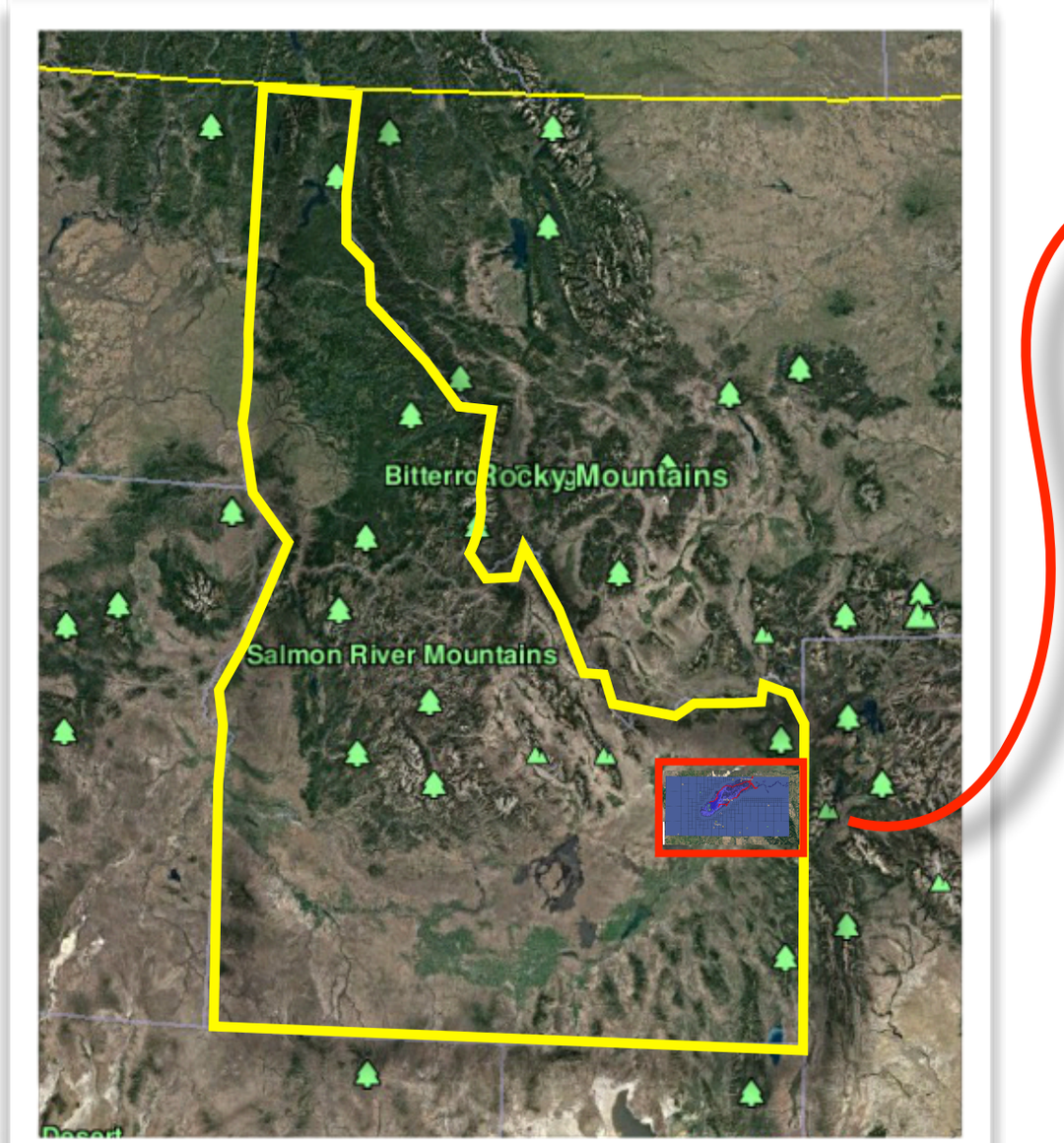
- F90, in particular modules! Argument free subroutines make it difficult to allow same subroutines to operate on multiple versions of the data.

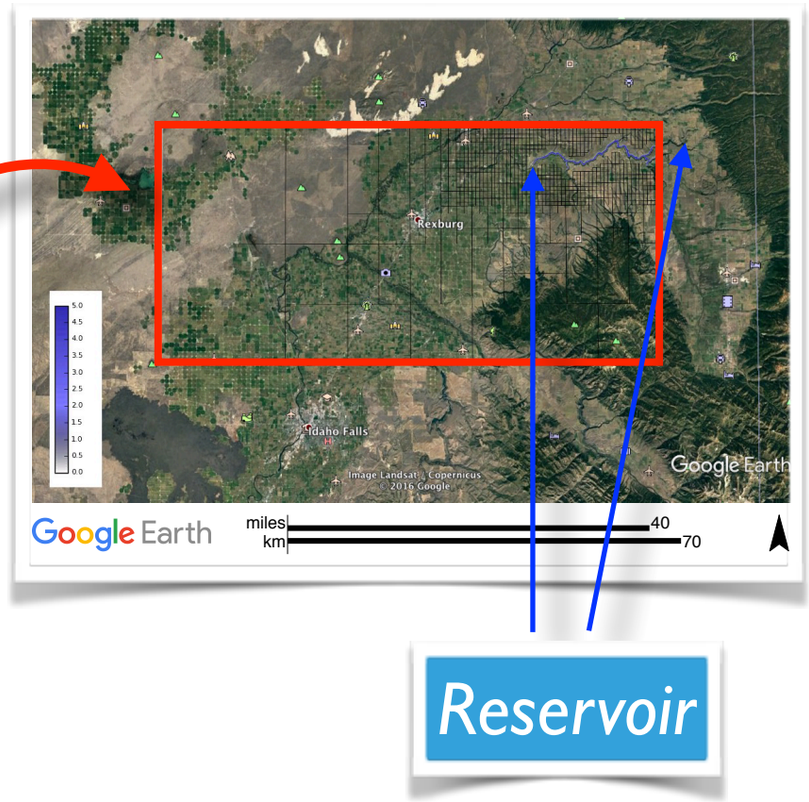
Teton Dam Failure, June 5, 1976

In collaboration with Idaho National Laboratories, we have been using GeoClaw to simulate the Teton Dam failure

- On June 5, 1976, the Teton Dam in eastern Idaho failed.
- 11 people died and $2b in damages; several cites were inundated, including Rexburg, ID.
- Historical data used as validation for using GeoClaw to study potential flooding of nuclear power plants,

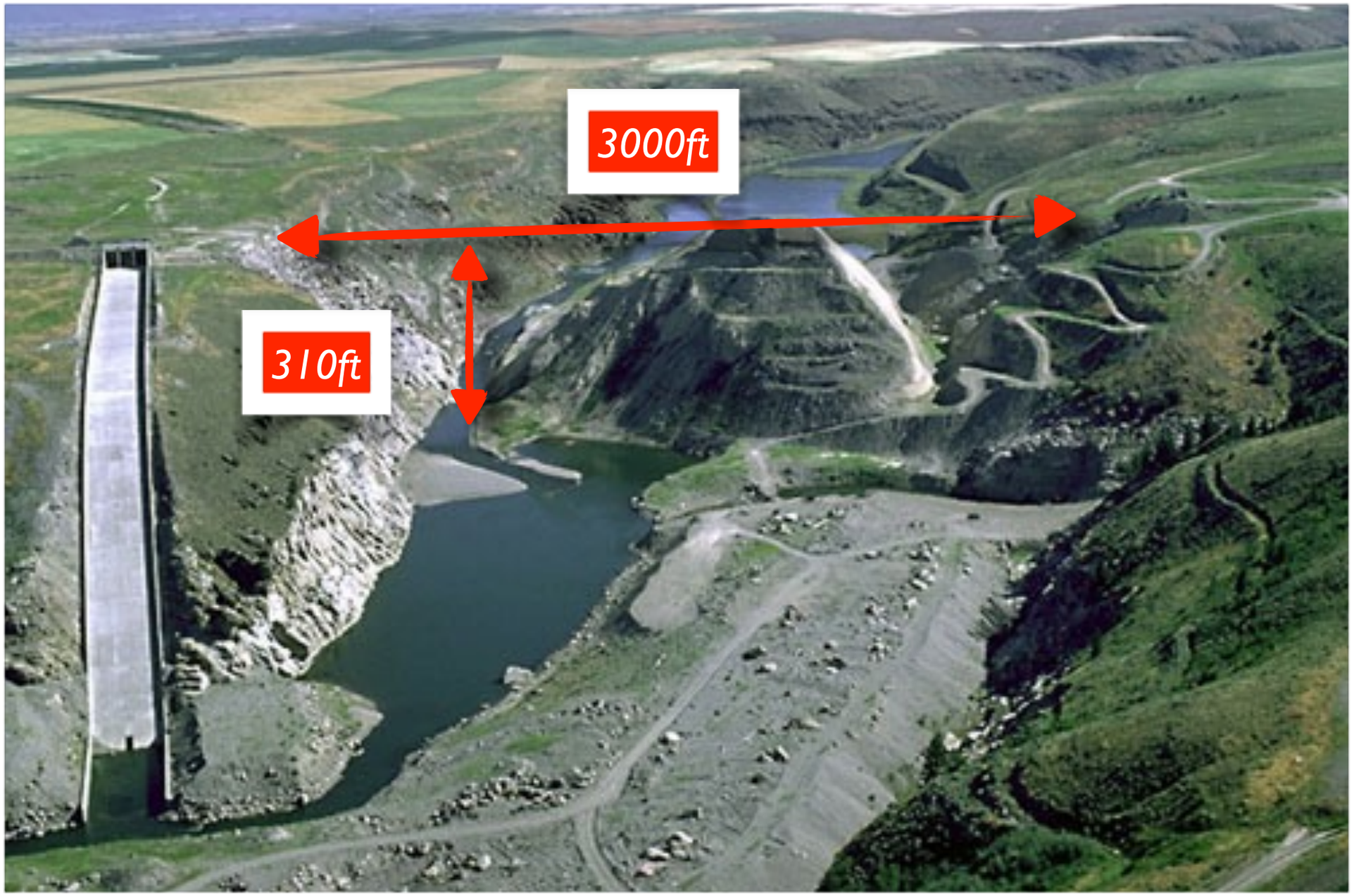
Teton Dam Failure, June 5, 1976





Computational Area : 88km x 42km

Teton Dam Failure, June 5, 1976



Teton Dam Failure, June 5, 1976

8 minutes before dam failure

Teton Dam Failure, June 5, 1976

~11:52 AM, June 5, 1976

Teton Dam Failure, June 5, 1976

By WaterArchives.org from Sacramento, California, USA - [IDAHO-L-0010] Teton Dam Flood - Newdale, CC BY-SA 2.0,

Historical Data

Table 2 Teton Dam Failure Summary Flood Data (Primary source: USGS Open-File Report 77-765)					
Location	Miles from Dam	Flood Arrival Time	Flood Arrival Travel Time (time from embankment breach)	Peak Flow (cubic feet per second)	Flood Description
Teton Canyon	2.5	12:05 p.m. June 5	8 minutes	2,300,000	50 to 75 ft wall-of-water
Near mouth of Teton Canyon	5.0	12:20 p.m.	23 minutes		
Wilford	8.4				120 of the 154 homes "completely swept away"
Town of Teton	8.8	12:30 p.m.	33 minutes	1,060,000	Only tiny fraction flooded
Sugar City	12.3	About 1:30 p.m.	1.5 hours		15-foot wall-of-water
Rexburg	15.3	About 2:30 p.m.	2.5 hours		6 to 8 feet in a few minutes
Roberts	43.1	9:00 p.m.	9 hours		
Idaho Falls	63.0	1 a.m. June 6	13 hours	90,500	
Shelley	71.2	2 a.m.	14 hours	67,300	Peak 21 hours after arrival. 0.5 feet per hour average rate of rise.

W. Graham, "Reclamation : Managing water in the west, The Teton Dam Failure - An effective warning and evacuation", U.S. Department of the Interior, Bureau of Reclamation, Denver Colorado

Inundation map

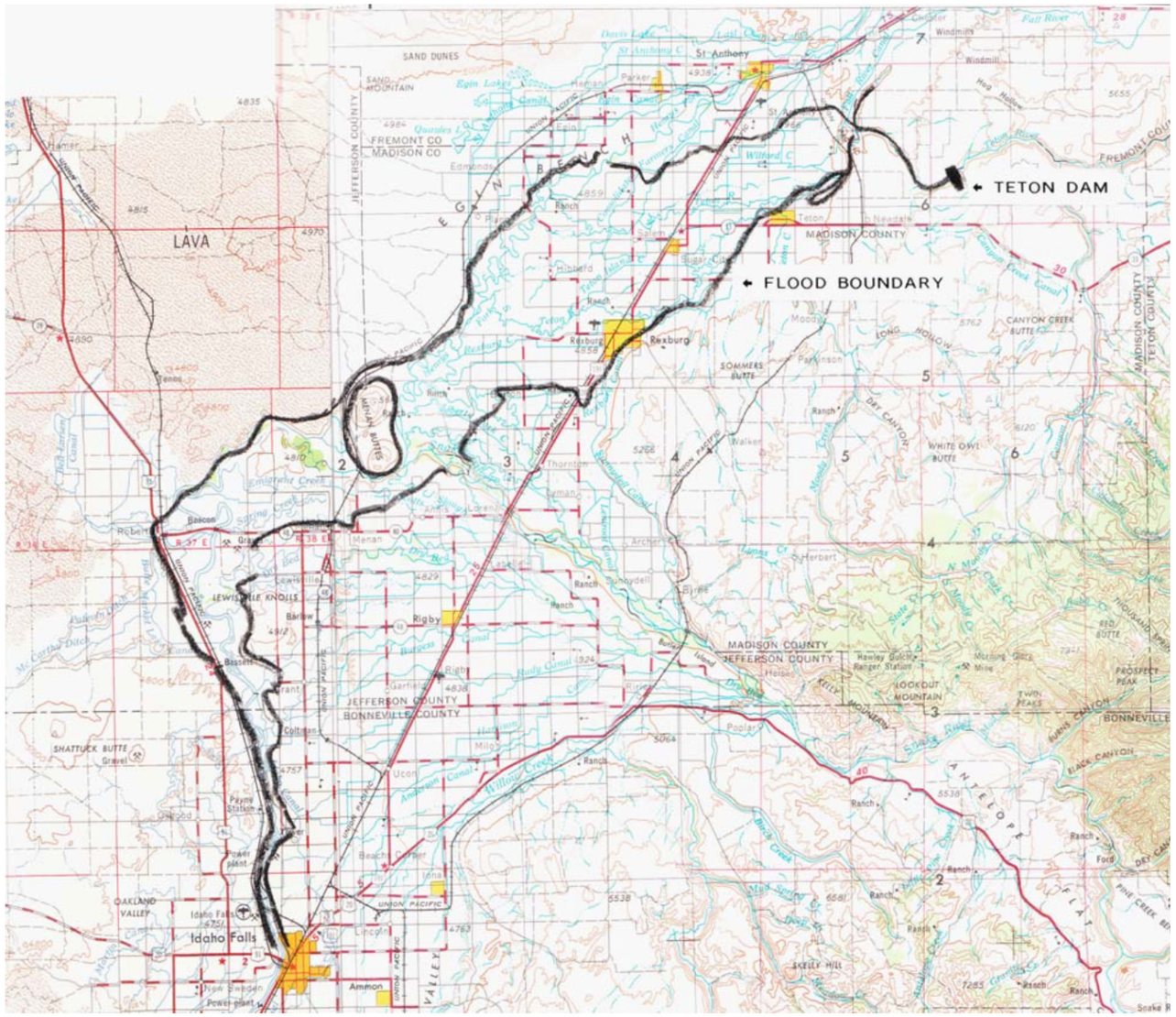


Figure 2 – Teton Dam Failure Inundation Map from Teton Dam to Idaho Falls

W. Graham, "Reclamation : Managing water in the west, The Teton Dam Failure - An effective warning and evacuation", U.S. Department of the Interior, Bureau of Reclamation, Denver Colorado

Eye witness accounts

In Idaho Falls, eye-witnesses report four feet of water covers the west side of town. This part of Idaho Falls is directly along the Snake River. The 17th Street bridge in Idaho Falls was dynamited late Sunday afternoon, to prevent water from backing up behind debris caught on the structure.

Waters in Rexburg are reported to be receding, from a Saturday night high of over eight feet. Several fires, which were still burning early Sunday morning, have also been extinguished.

— GenDisasters.com

One of the most famous photographs from the disaster was taken just as a wave of water breached the Broadway Bridge.

Bower said he had been standing next to the bridge for a time on June 6, watching as the water steadily rose. He sidled up to a police officer who was preventing anyone from crossing.

— "40 years later, remembering Idaho's Teton Dam collapse", Idaho Statesman, June 5, 2016

Simulations using ForestClaw

Problem set up using GeoClaw extension of ForestClaw

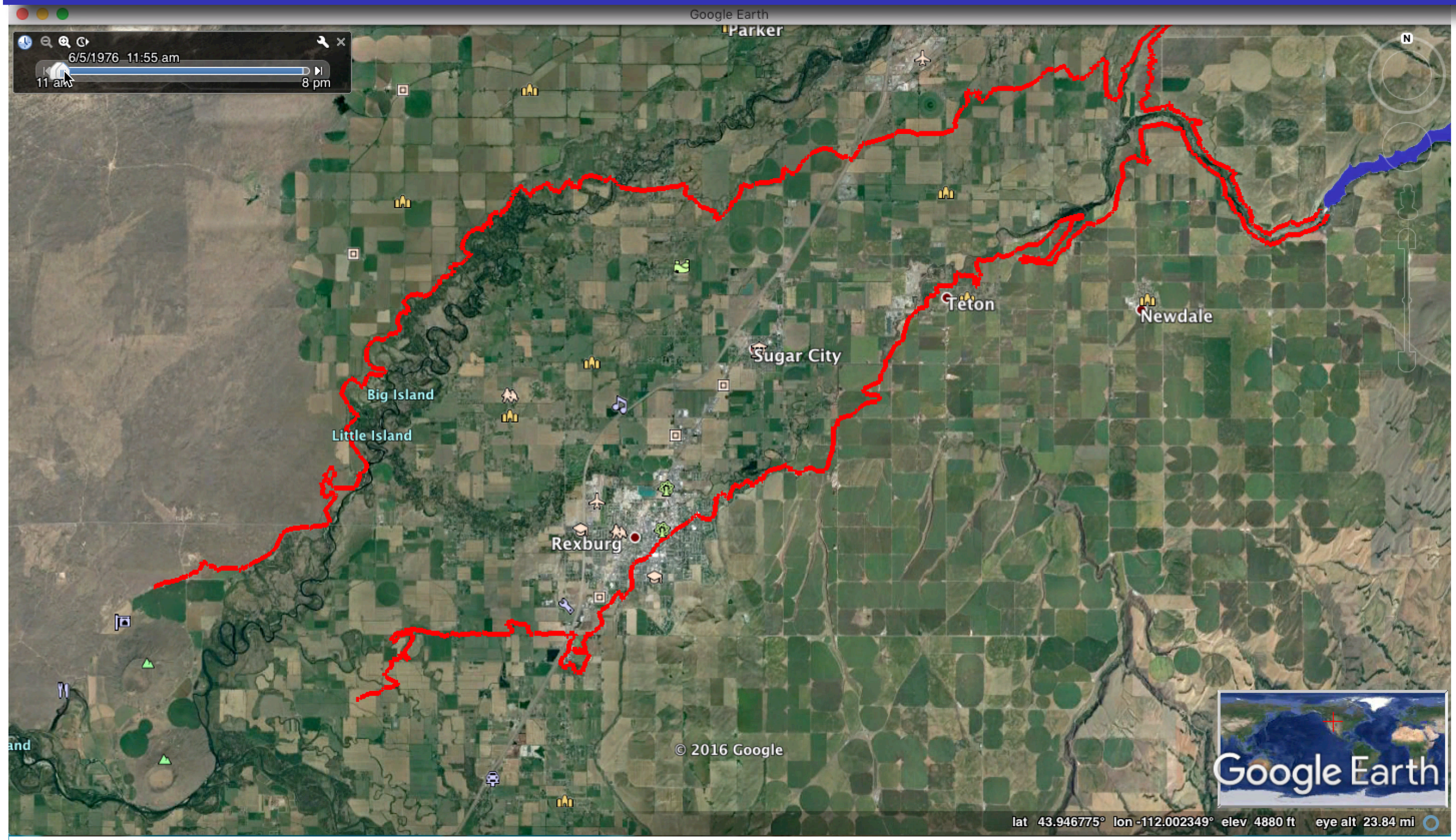
Simulation details :

- Run at 10m effective resolution (8192 x 4096)
- 16 hours of simulation time
- Manning coefficient set to 0.025
- Results compared with historical flood boundaries and arrival times
- No detailed modeling of the dam failure itself

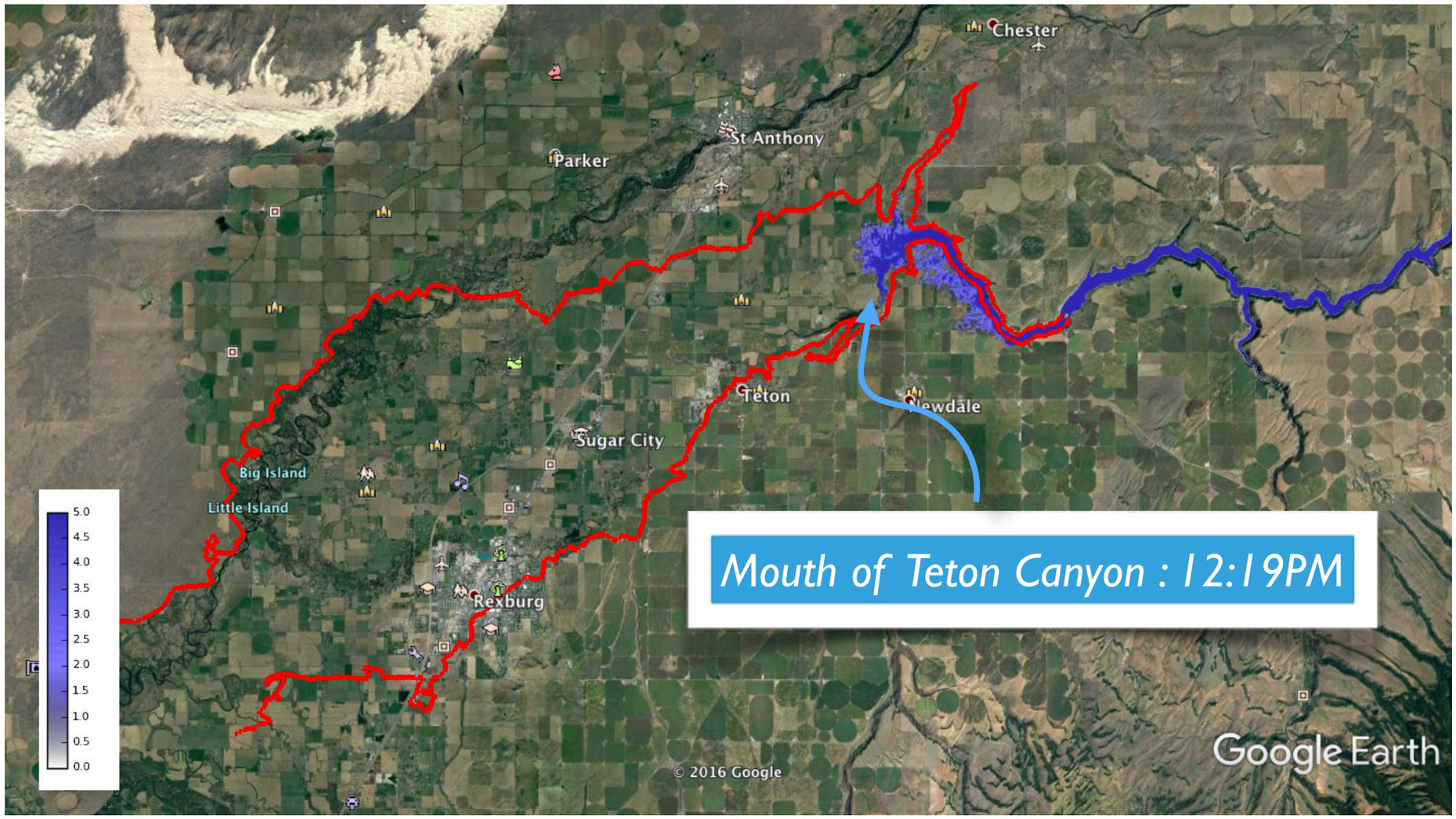
Numerical parameters

- 7 levels of refinement
- standard ‘feature-based’ refinement based on wave speeds and depth
- 2 blocks or quad-trees used to grid the domain

Simulation results



Simulation results

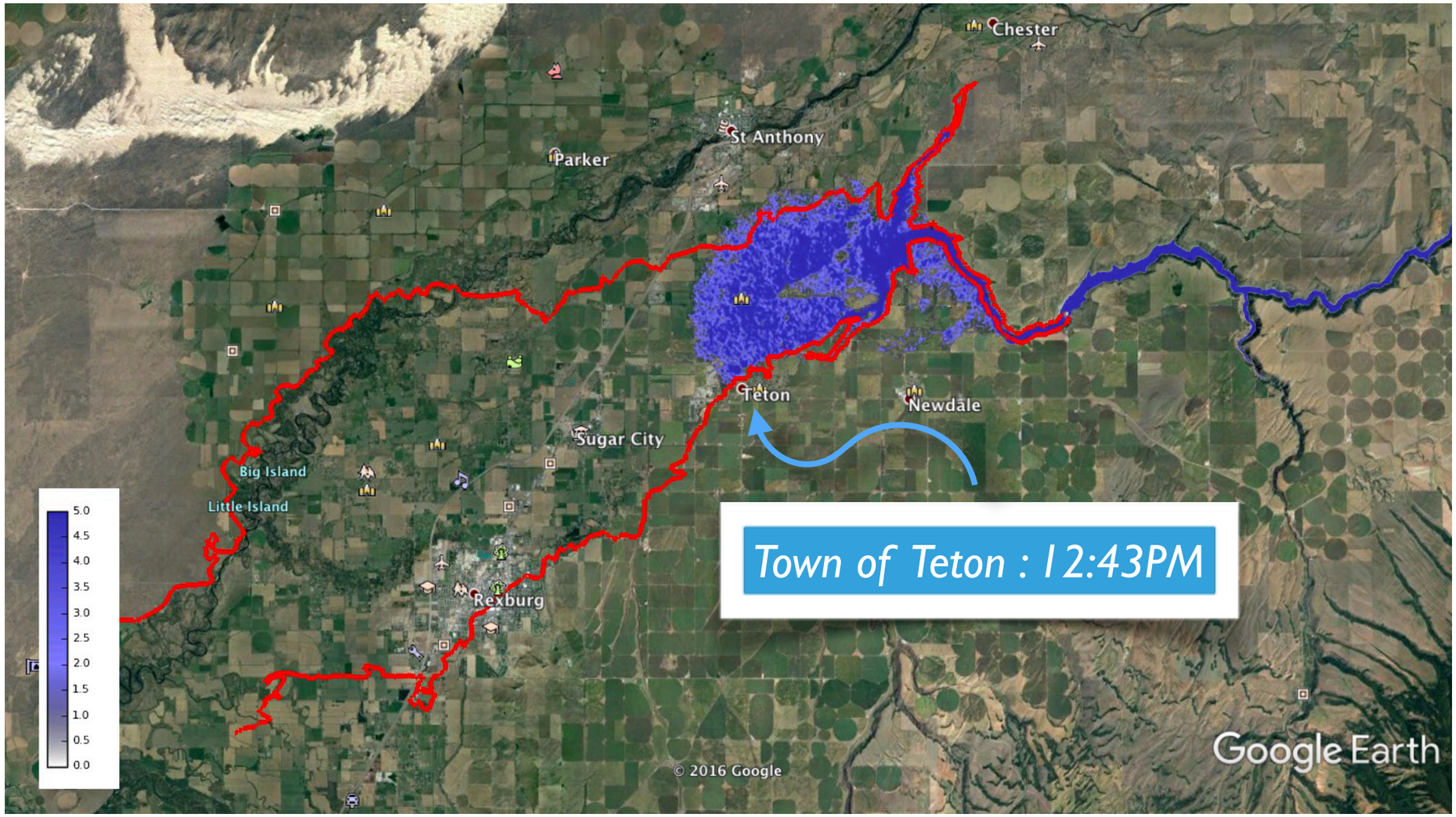


Google Earth

miles 10
km 20

Near mouth of Teton Canyon	5.0	12:20 p.m.	23 minutes		

Simulation results

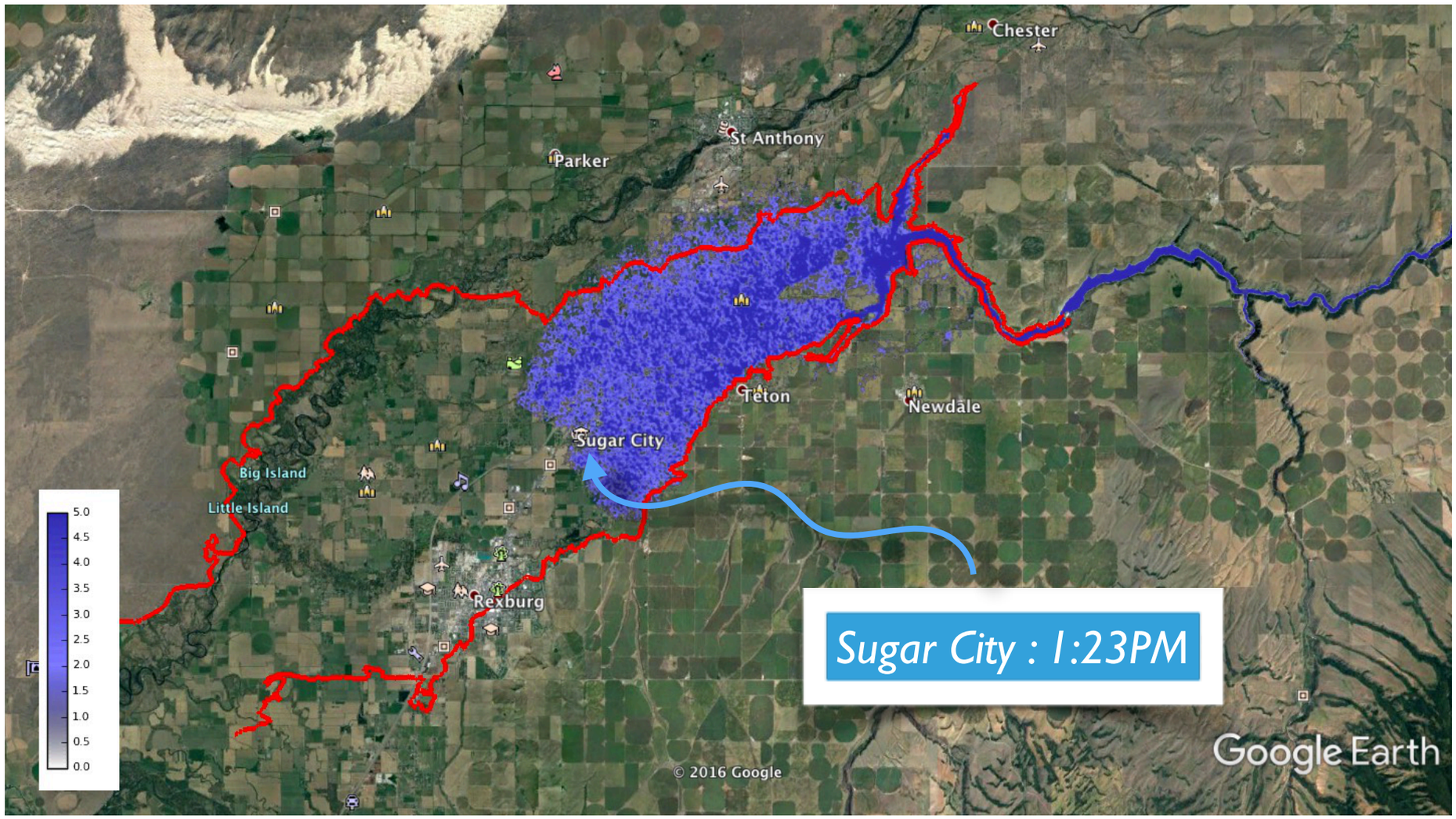


Google Earth

miles 10
km 20

Town of Teton	8.8	12:30 p.m.	33 minutes	1,060,000	Only tiny fraction flooded

Simulation results

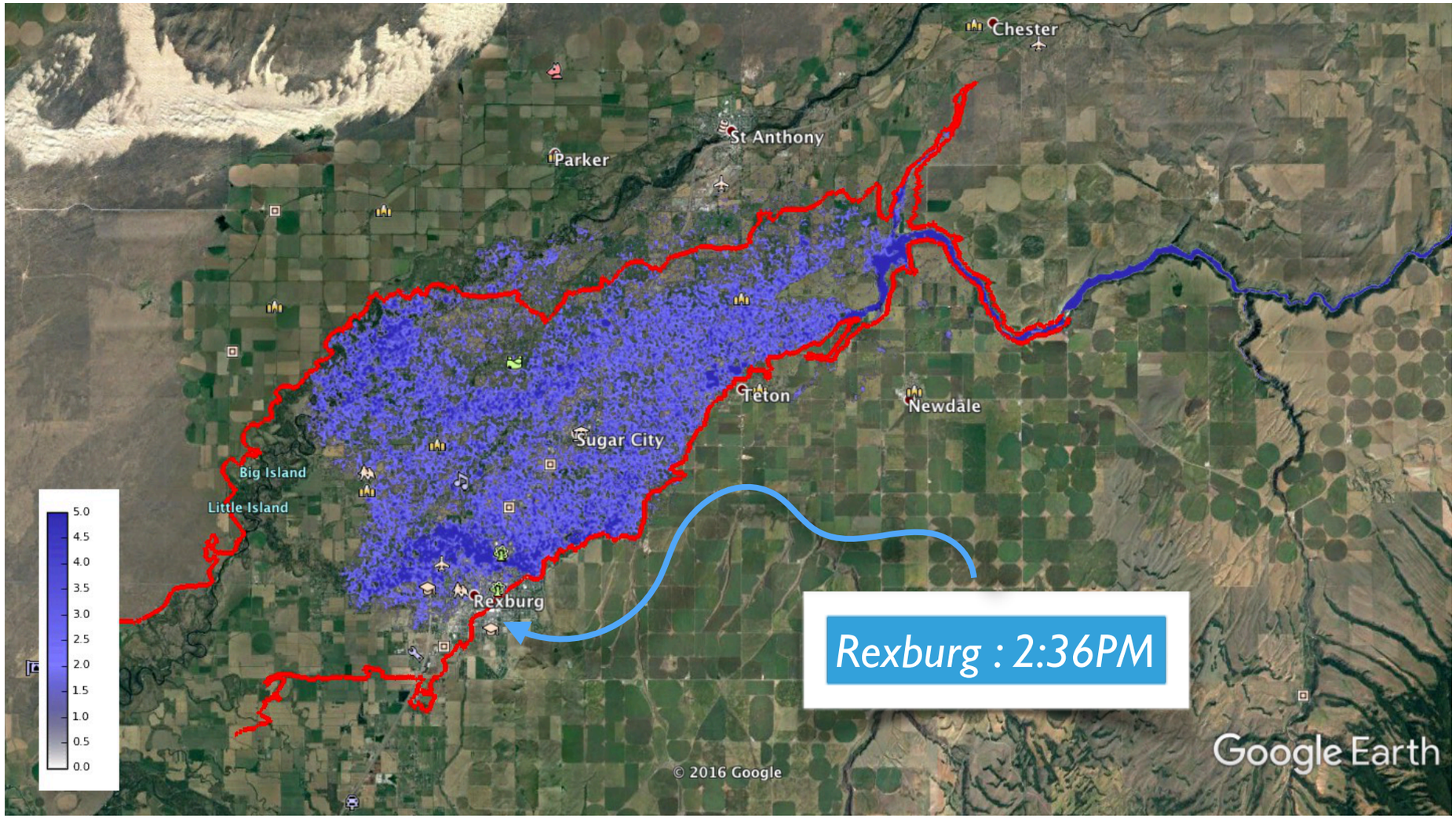


Google Earth

miles 10
km 20

Sugar City	12.3	About 1:30 p.m.	1.5 hours		15-foot wall-of-water

Simulation results

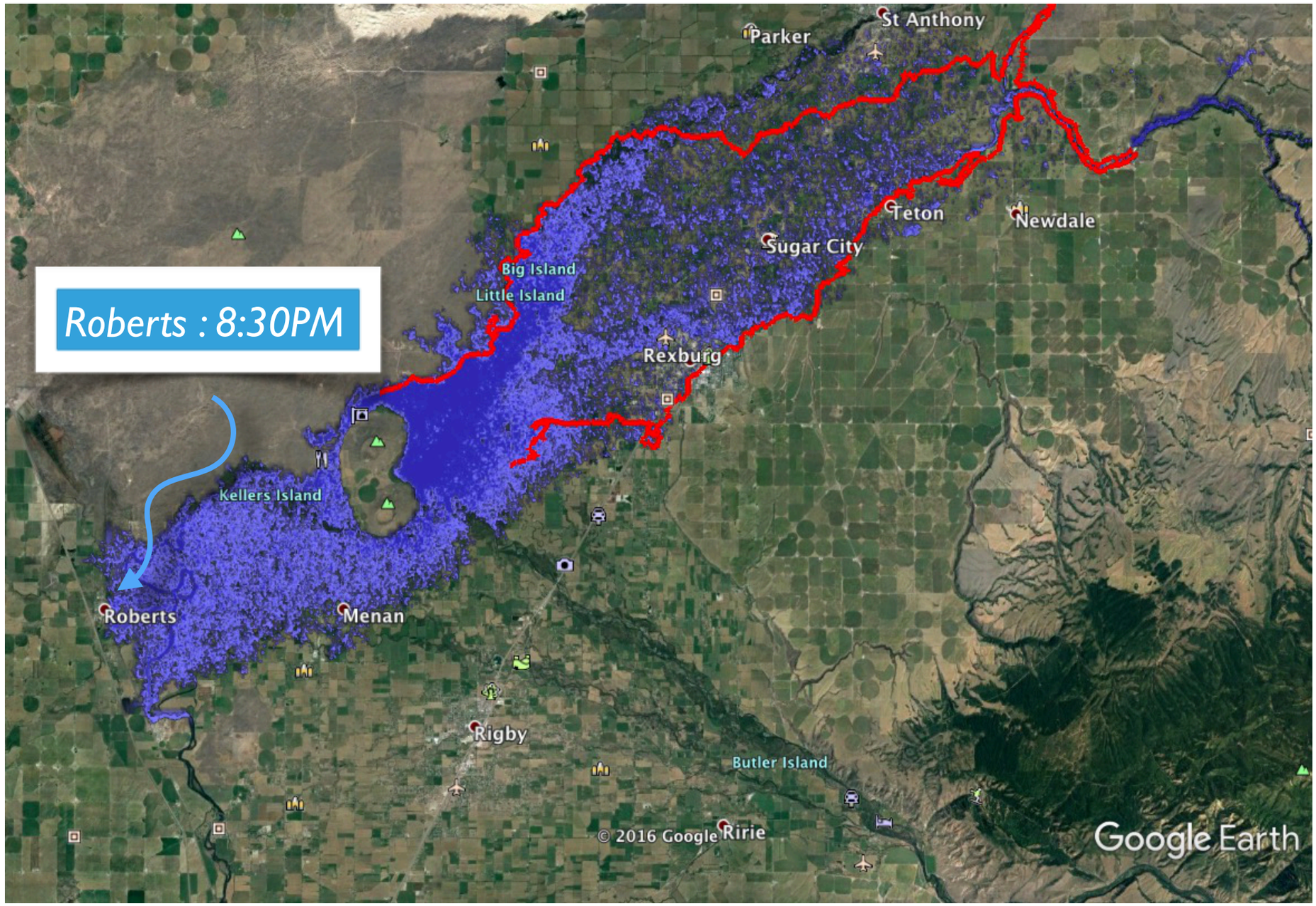


Google Earth

miles 10
km 20

Rexburg	15.3	About 2:30 p.m.	2.5 hours		6 to 8 feet in a few minutes

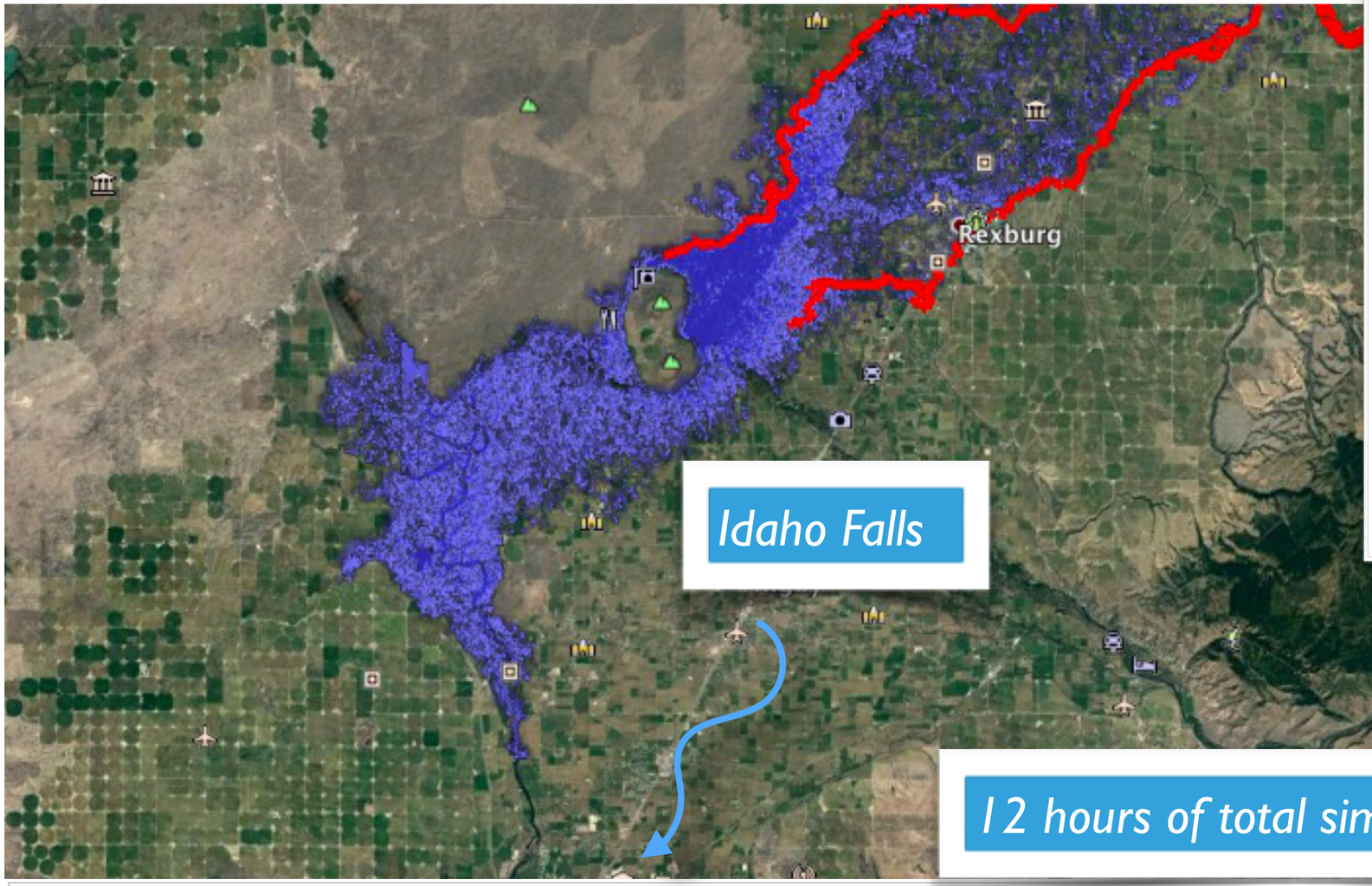
Simulation results



	miles			10	
Roberts	43.1	9:00 p.m.	9 hours		
Idaho Falls	63.0	1 a.m.	13 hours	90,500	

Simulation results

Google Earth



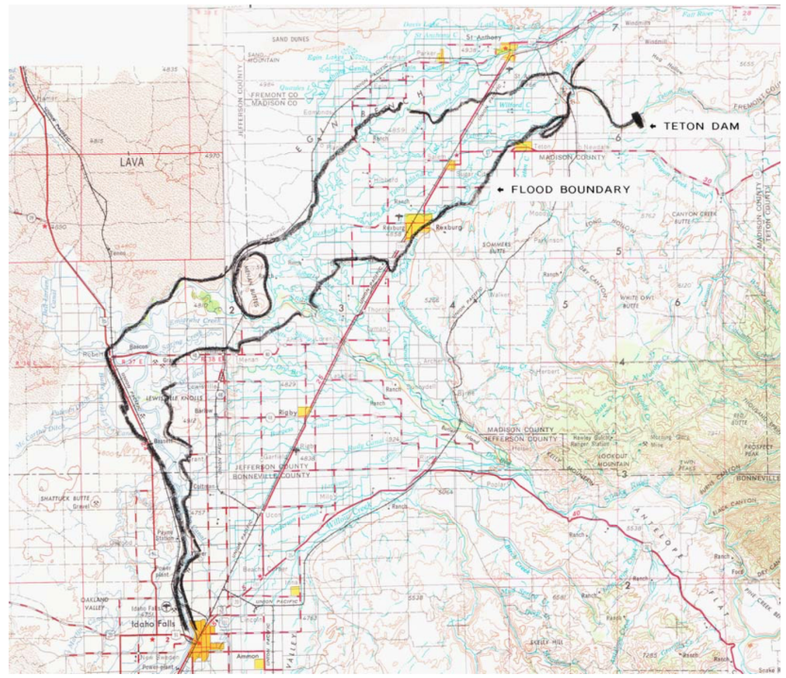


Figure 2 – Teton Dam Failure Inundation Map from Teton Dam to Idaho Falls

Level 0

Roberts	43.1	9:00 p.m.	9 hours		
Idaho Falls	63.0	1 a.m.	13 hours	90,500	

Parallel/AMR Efficiency

~ 10m resolution (8192 x 4096)

6.5 hours vs. 30 minutes

Procs	14	28	56	112	224
Wall (s)	23601.9	12510.6	6626.7	3499.7	1872.9
Speed-up	1.00	1.89	3.56	6.74	12.60
Efficiency	**100%**	**94%**	**89%**	**84%**	**79%**
Grids per processor	**670**	**334**	**167**	**83**	**41**

Procs	Wall	Advance	(%)	Ghost Comm	(%)	Ghost fill	(%)	Regrid	(%)	Speed-up	Par. eff.
14	23601.9	17706.4	**75%**	4500.4	**19%**	1343.3	**6%**	28.5	**0%**	1.0	**100%**
28	12510.6	8863.0	**71%**	2838.0	**23%**	772.4	**6%**	17.0	**0%**	1.9	**94%**
56	6626.7	4453.7	**67%**	1714.5	**26%**	432.6	**7%**	9.1	**0%**	3.6	**89%**
112	3499.7	2229.0	**64%**	1002.8	**29%**	248.1	**7%**	5.3	**0%**	6.7	**84%**
224	1872.9	1114.1	**59%**	602.8	**32%**	138.6	**7%**	3.3	**0%**	12.6	**79%**

Outstanding issues

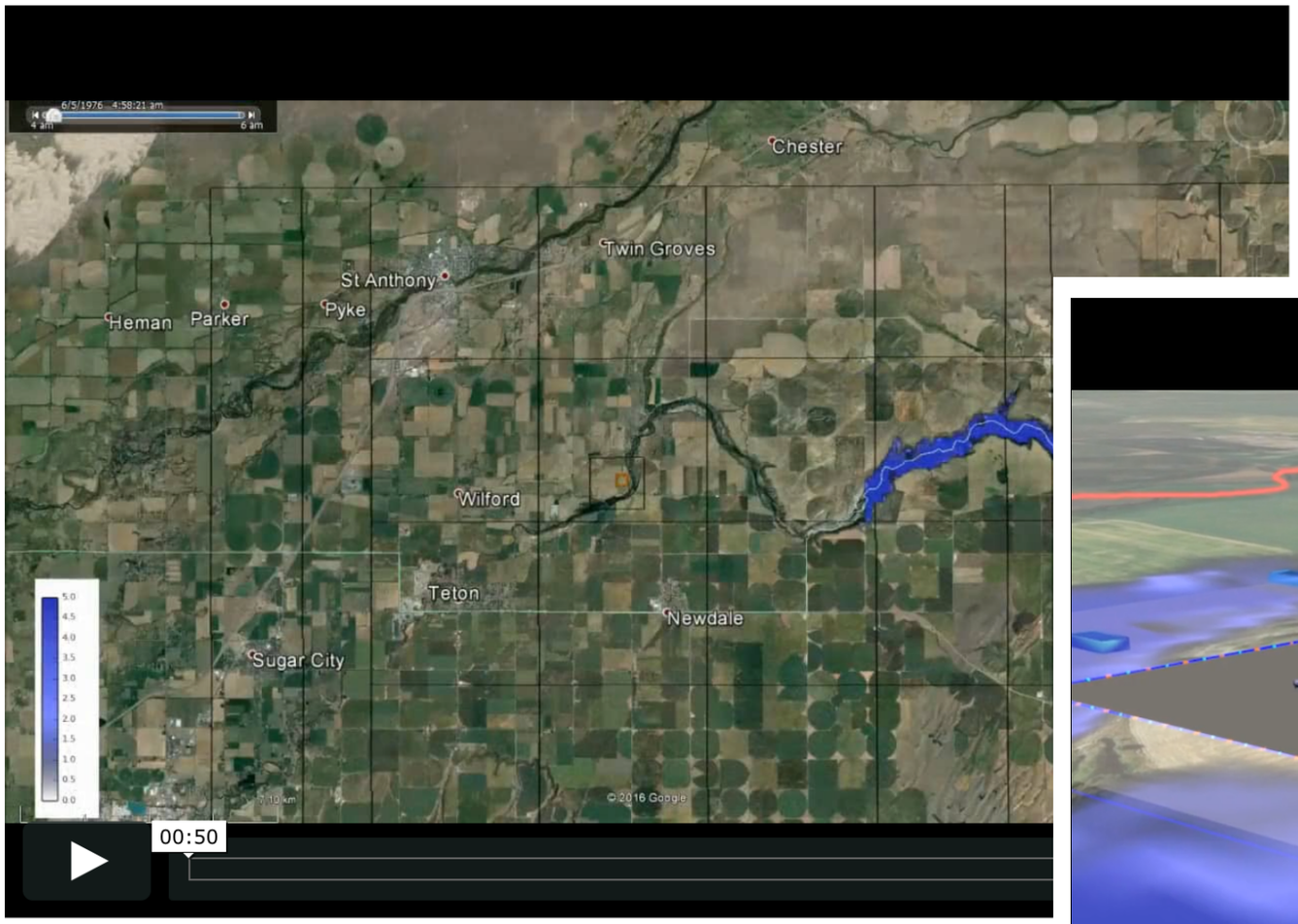
What governs good parallel performance?

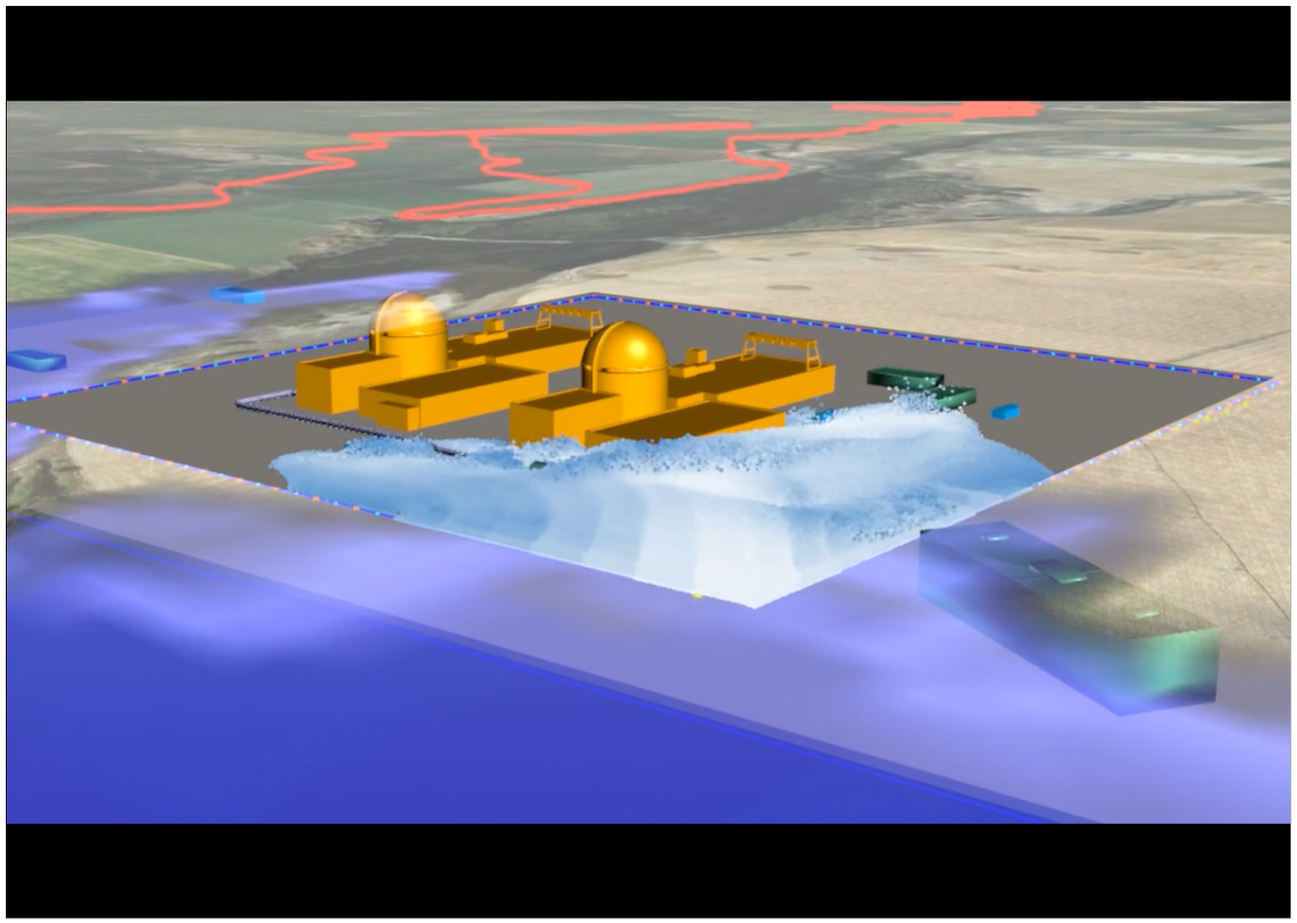
- Sufficient number of grids per processor is key to good parallel efficiency
- Grid size (8x8, 16x16, 32x32 and so on) is also important,
- Ghost cell communication involves not only the solution but also bathymetry, needed to average fine grid values onto the coarse grid ghost cells

Outstanding issues

- Sub-cycling (local time stepping, multi-rate time stepping) in time for SWE?
- Handling console IO and user interface to Python routines
- Push the code to fine resolutions? (~5m, ~1m ….)
- Maximum flooding at a prescribed set of values (“fixed grid solutions”).

Teton Dam Failure, June 5, 1976





Ram Sampath, Centroid Lab, Los Angeles, CA

http://neutrinodynamics.com//portfolio-riverflood.html

Future plans

- Description of parallel ghost-filling algorithm

  D. Calhoun, C. Burstedde, *ForestClaw : A parallel algorithm for patch-based adaptive mesh refinement on a forest of quadtrees*, (submitted), 2017. (arXiv:1703.03116)

- Multi-rate time stepping for Runge-Kutta-Chebyshev time stepping for parabolic equations (T. Mirzakhanian, BSU)
- Elliptic solvers using PETSc (J. Brown, Univ. of Colorado)

ForestClaw extension to volcanic ash cloud modeling

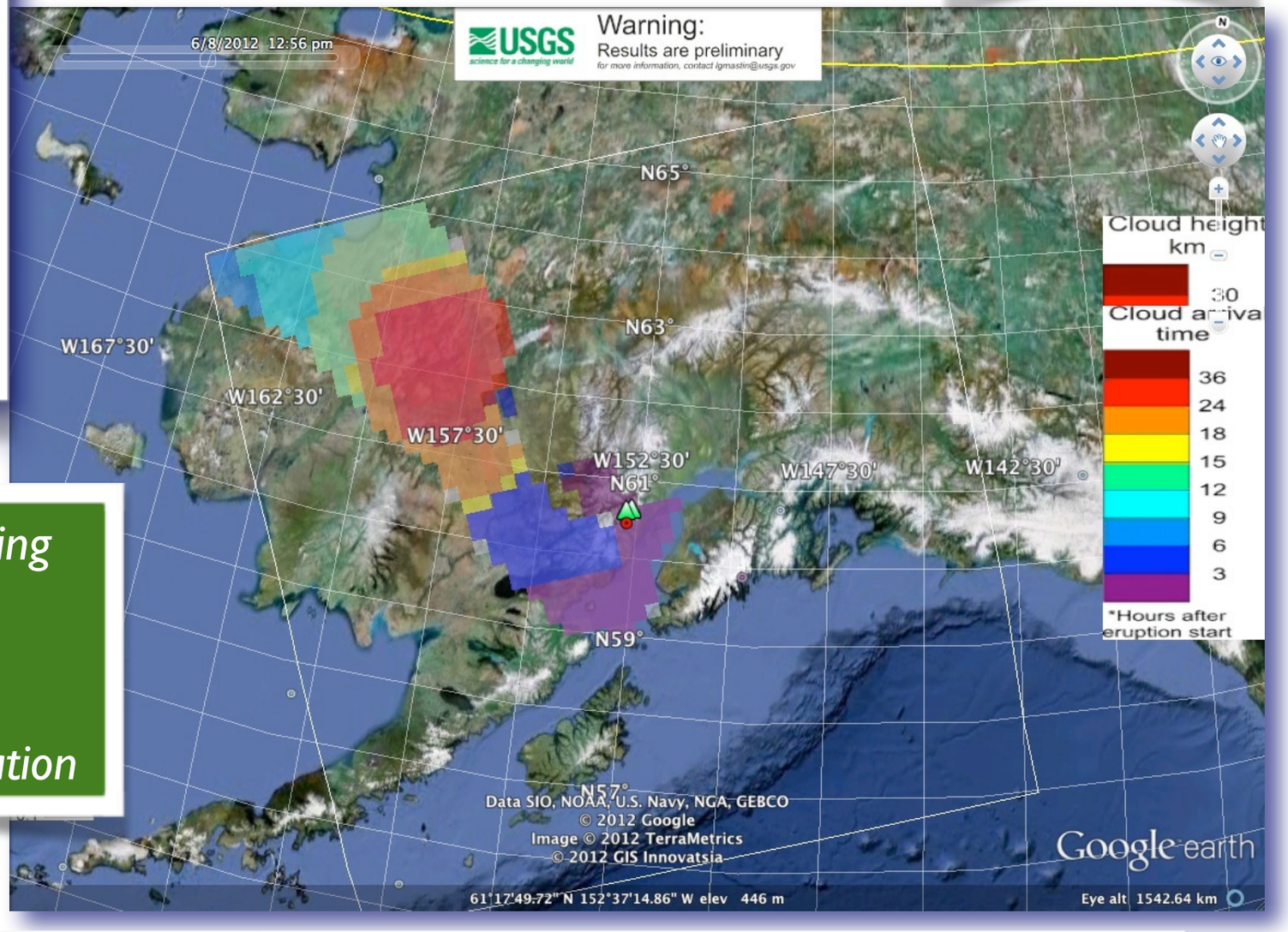
- Collaboration with the USGS (Vancouver, WA) to add parallelism and adaptivity to Ash3d, a transport code for modeling dispersion of volcanic ash.

Thanks to the NSF for supporting this project!

www.forestclaw.org

Ash cloud modeling

Ash3d

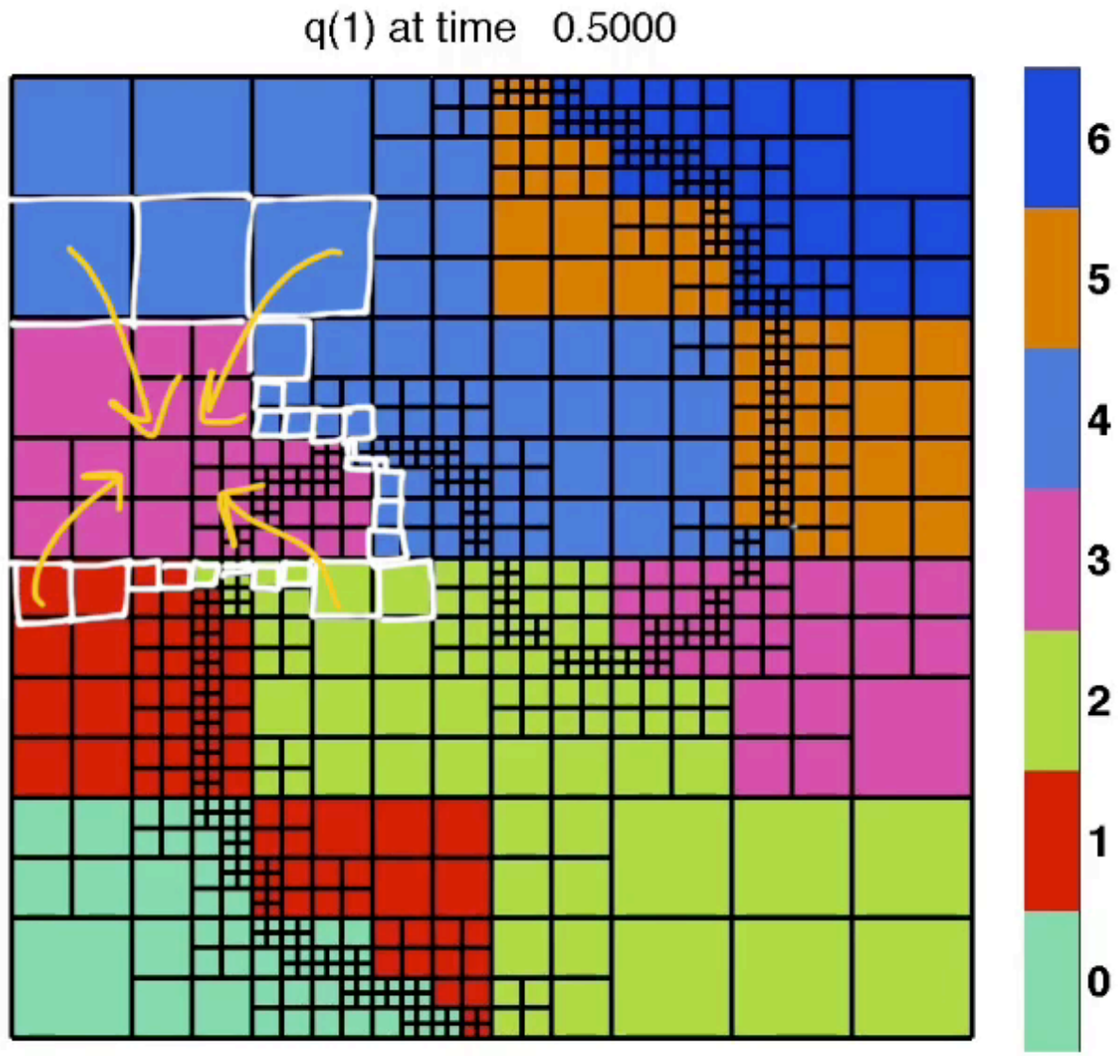


- *Split horizontal, vertical time stepping*
- *Fully conservative,*
- *Eulerian, finite volume*
- *Algorithms based on wave propagation*

Ash3d :A finite-volume, conservative numerical model for ash transport and tephra deposition, Schwaiger, Denlinger, Mastin, JGR (2012)

Parallel implementation

- Grids are ordered and load balanced using *Morton ordering (z-ordering)*
- *Grids at processor boundaries are exchanged*



Why are AMR codes difficult to write?

- Heterogeneous data structures associated with storing hierarchy of grids,
- Dynamically creating and destroying grids :

  Need a “factory” paradigm to create new grids and any auxiliary data arrays (material properties, metric terms, bathymetry, etc) that go with each new grid,
- Parallel load balancing
- Coupling of multiple solvers (e.g. advection + diffusion + elliptic solvers + source terms),
- Multi-physics,
- Data visualization and post-processing
- Computing diagnostics on a nested grid hierarchy,
- Error estimation, tuning for efficient use of grids, ...

What makes AMR codes difficult to use?

- Time stepping beyond just single step explicit schemes (IMEX, SSP, RK, ...)
- Understanding how overall time stepping interacts with dynamic grid creation, destruction and management.
- Implicit solvers,
- Coupling of multiple solvers (e.g. advection + diffusion + elliptic solvers + source terms),
- Multi-physics,
- Data visualization and post-processing
- Computing diagnostics on a nested grid hierarchy,
- Error estimation, tuning for efficient use of grids, ...