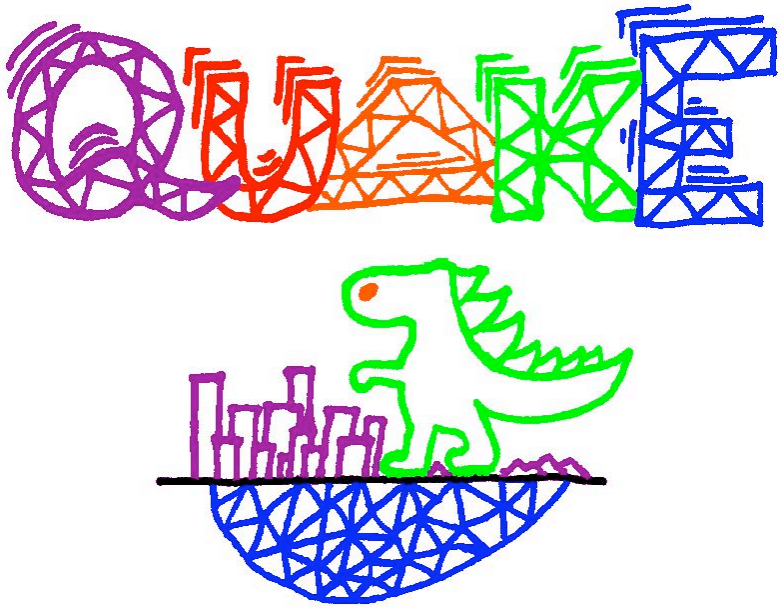
Octrees in Computational Seismology

David O'Hallaron

Director, Intel Research Pittsburgh
Associate Prof, CS and ECE, Carnegie Mellon University

CIG/SPICE meeting, October 2007



www.cs.cmu.edu/~quake

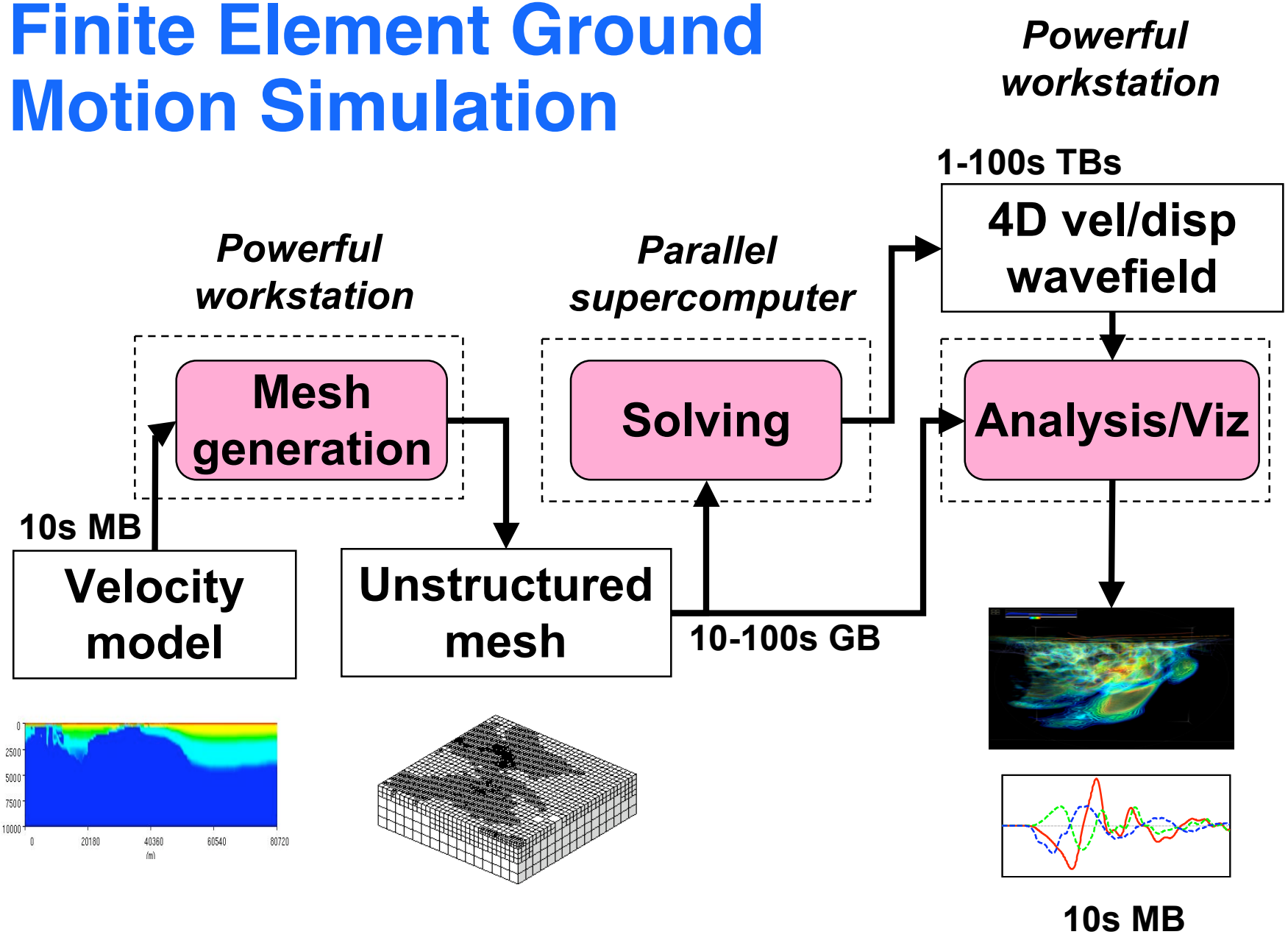
Quake Group
Carnegie Mellon University

Jacobo Bielak (CEE)
Aysegul Askan (CEE)
Leonardo Ramirez-Guzman (CEE)
Ricardo Taborda-Rios (CEE)

David O'Hallaron (CS & ECE, Intel Research Pittsburgh)
Julio Lopez (CS & PDL)
Tiankai Tu (CS, DE Shaw)

Carnegie Mellon

Finite Element Ground Motion Simulation



Hercules FE Ground Motion Code

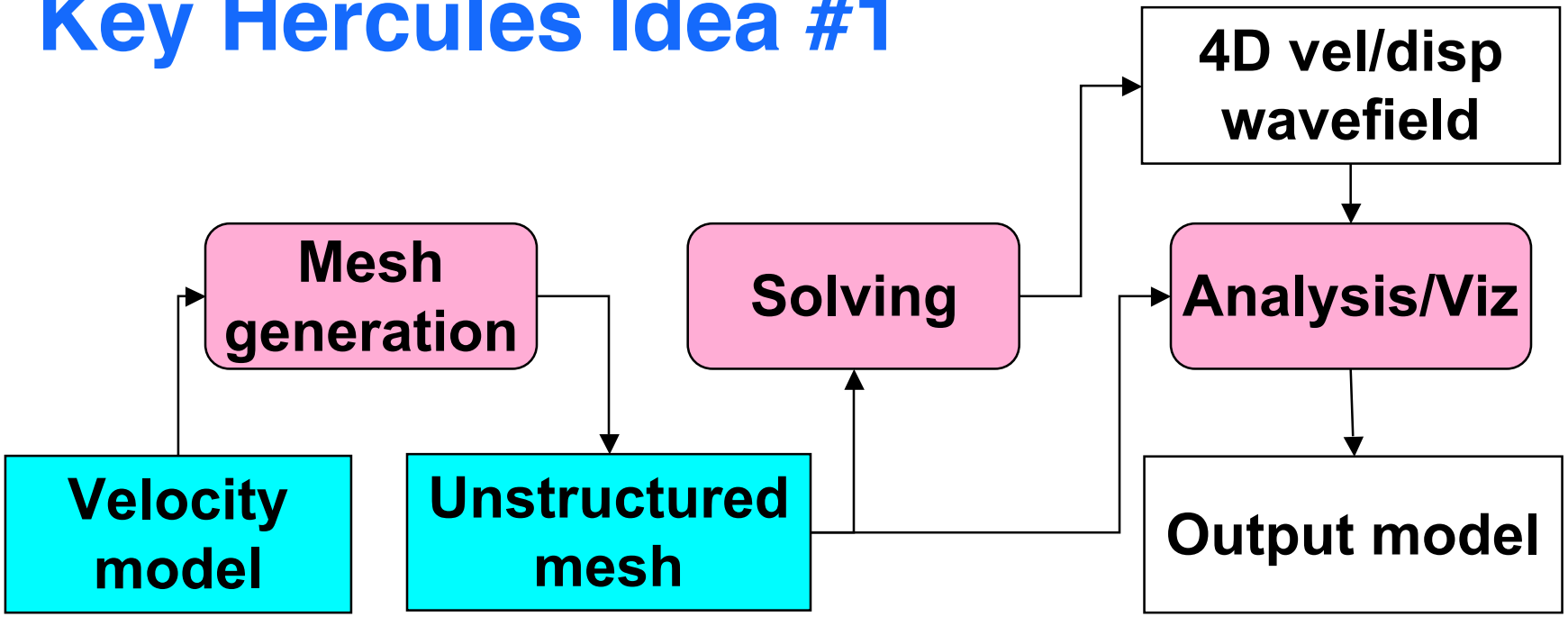
- **An *end-to-end* approach for parallel simulation**
  - **Parallel mesh generation, solving, and viz**
  - **Production ground motion code for the CMU Quake group**

- **Why end-to-end?**
  - **Eliminate intermediate files.**
  - **Fast turnaround on simulation/analysis cycle (e.g., 10 -> 8 nodes/wave)**
  - **Enables runtime steering of visualizations**

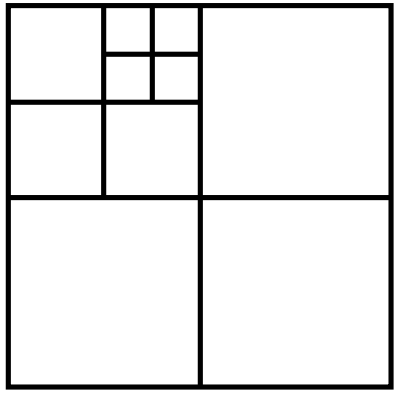
Culmination of long term collaboration among Carnegie Mellon computer scientists and civil engineers, and domain scientists at Southern California Earthquake Center (SCEC)

Tiankai Tu, Hongfeng Yu, Leonardo Ramirez-Guzman, Jacobo Bielak, Omar Ghattas, Kwan-Liu Ma, David R. O'Hallaron, *From Mesh Generation to Scientific Visualization: An End-to-End Approach to Parallel Supercomputing,* Proceedings of SC06, Tampa, FL, November, 2006.

Key Hercules Idea #1



Key Idea #1: Use *octrees* for the input datasets.



Structured Meshes

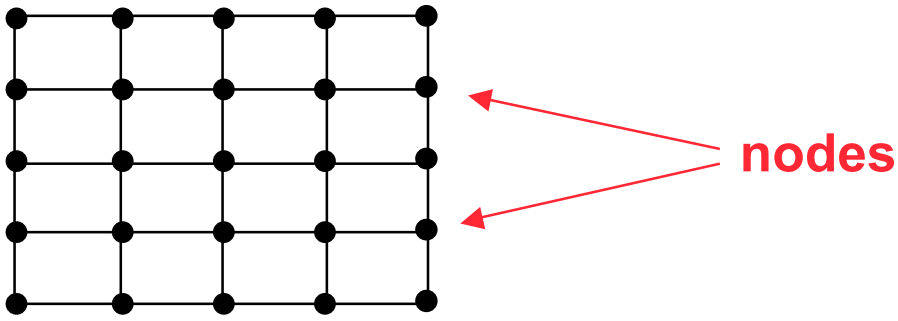
Structured mesh: **Collection of equally-sized rectangles/bricks**

+ Simple to build

+ No need to store topology info

- Not wavelength adaptive

- Cannot resolve arbitrary geometry



Unstructured Meshes

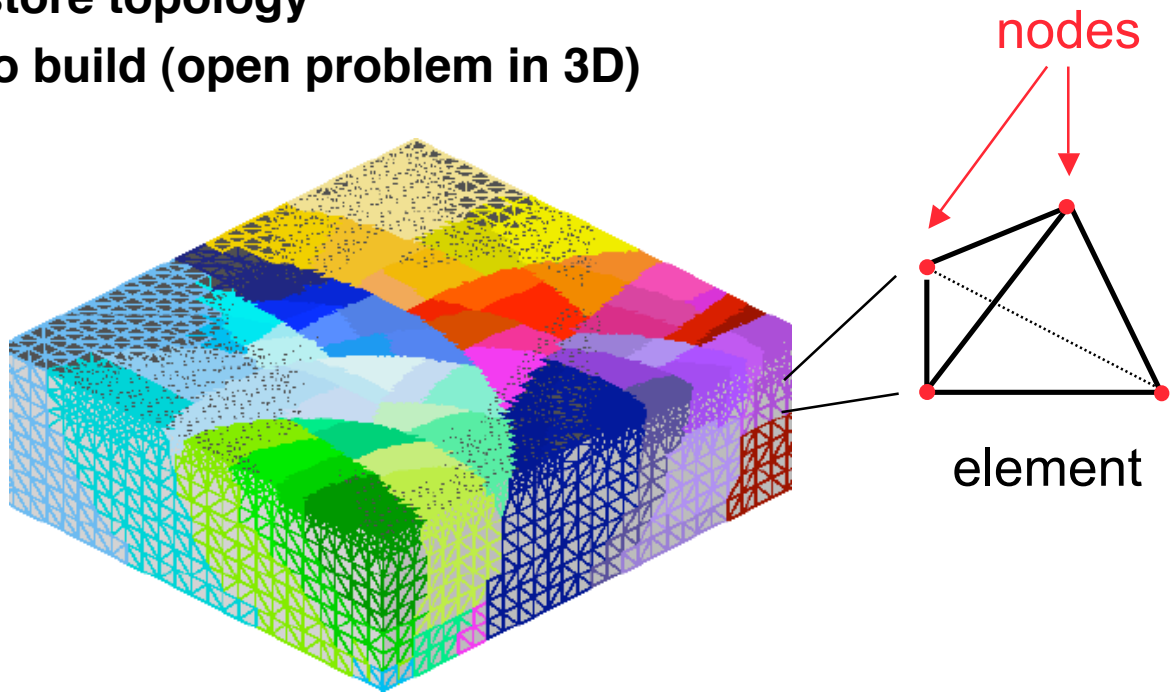
Unstructured mesh: **Collection of different sized triangles/tetrahedra**

+ Wavelength adaptive

+ Resolves arbitrary geometry

- Must store topology

- Hard to build (open problem in 3D)



Octree Meshes

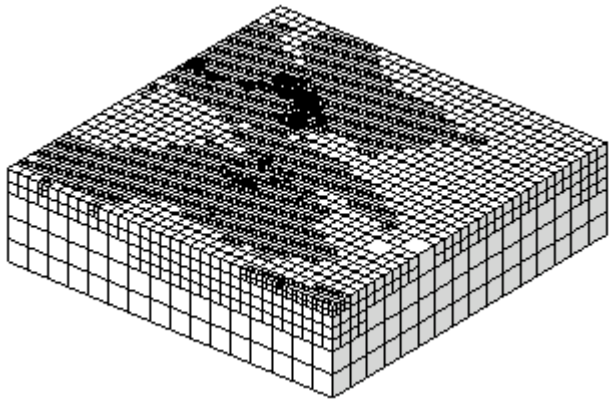
Octree mesh: **Hierarchical collection of different sized squares/cubes**

Interesting hybrid of structured & unstructured meshes

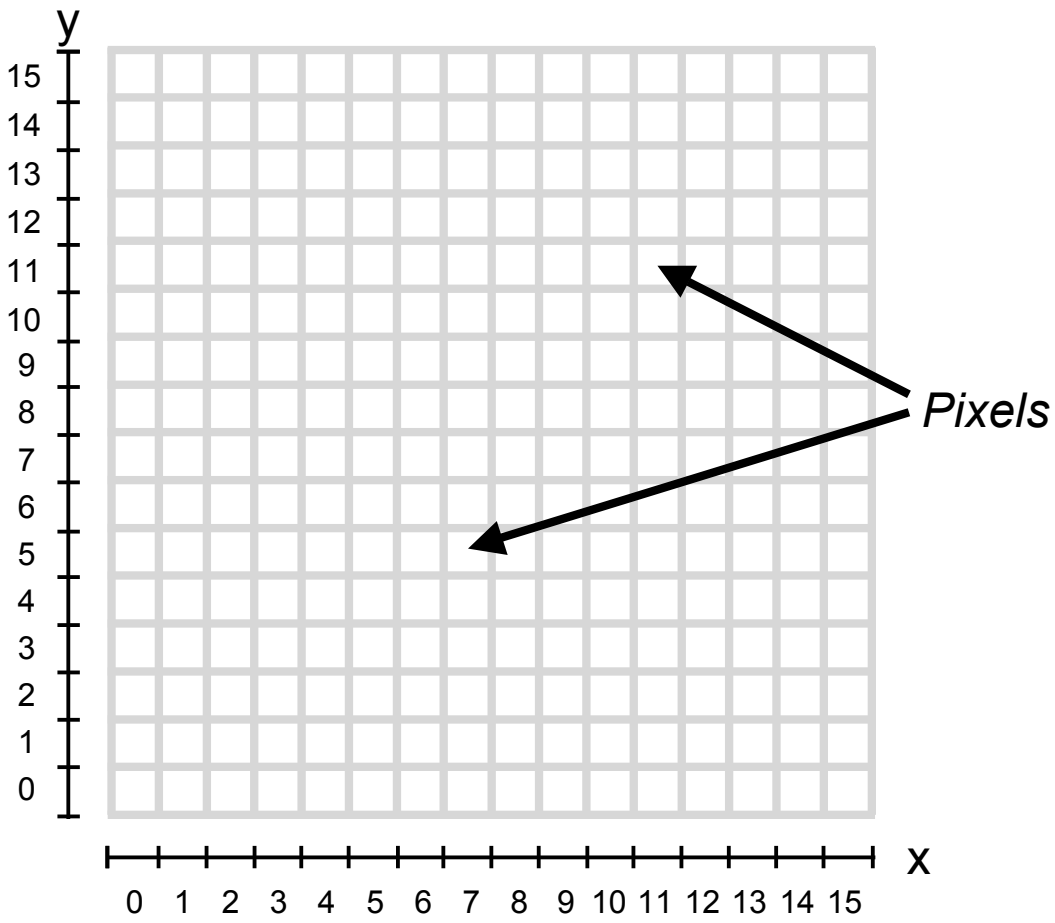
+ Simple to build
+ No need to store topology
+ Possess some beautiful mathematical properties
+ Efficient *etree* database representation enables both
- **Fast parallel mesh generation (fast velocity model queries)**
- **Fast 4D wavefield access (fast mesh queries)**

+ Wavelength adaptive

- Cannot resolve arbitrary geometries

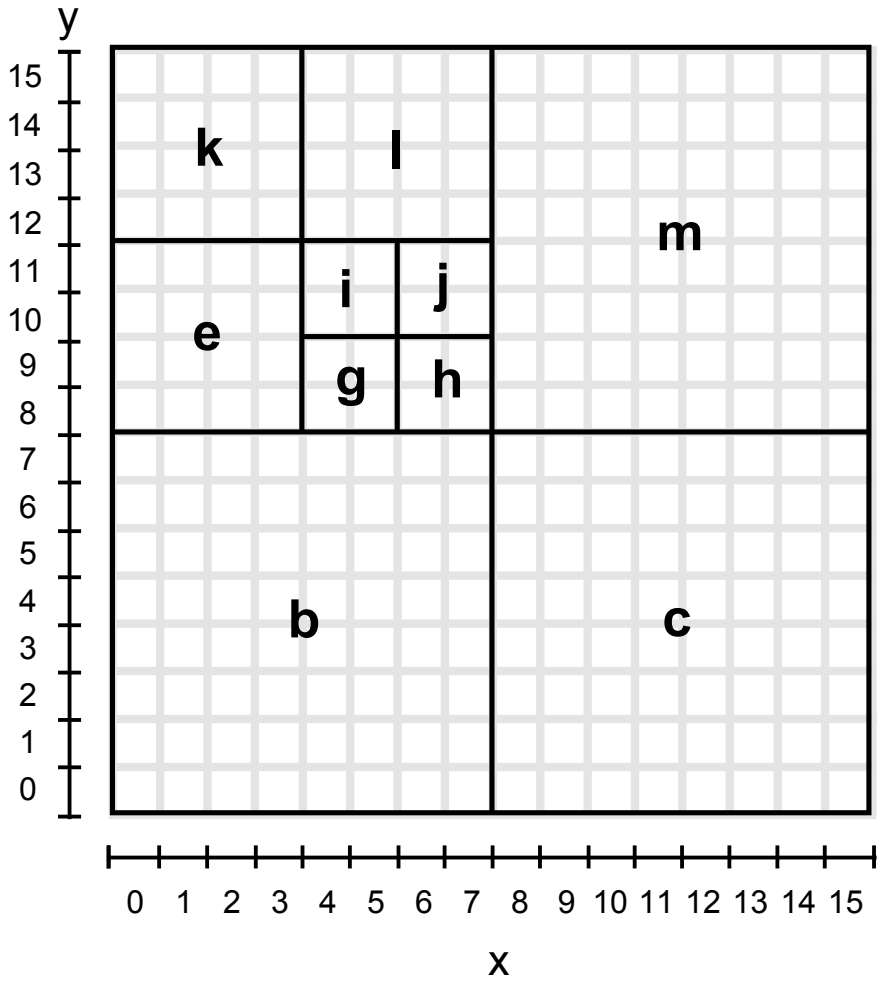


Octree Basics: An Octree Domain

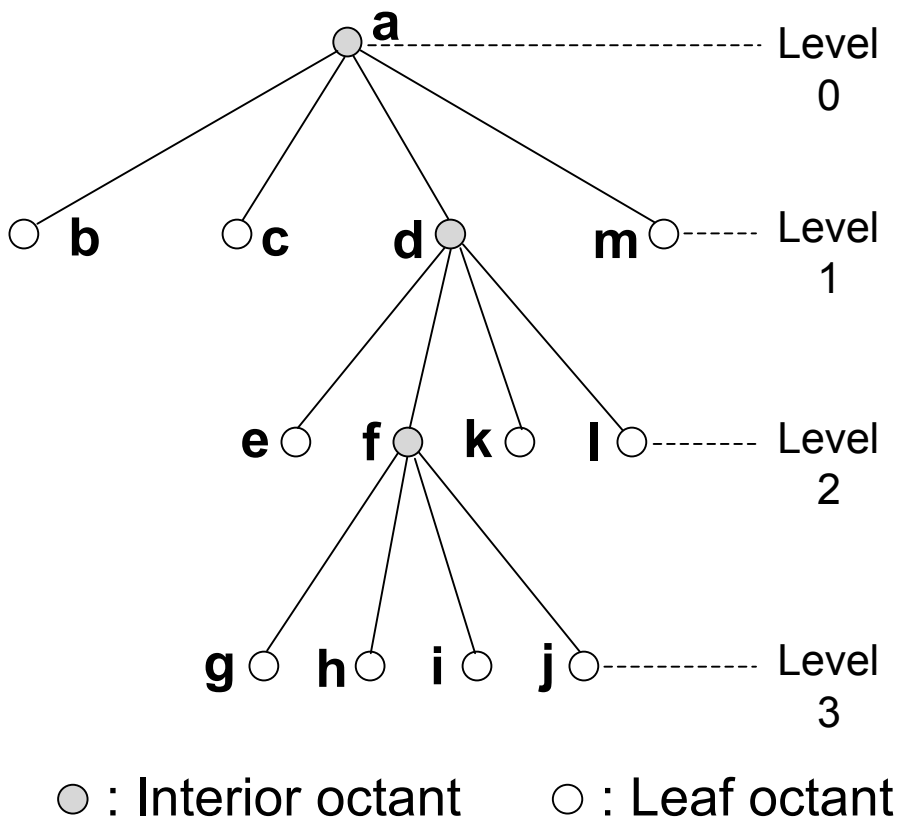


Equivalent Octree Representations

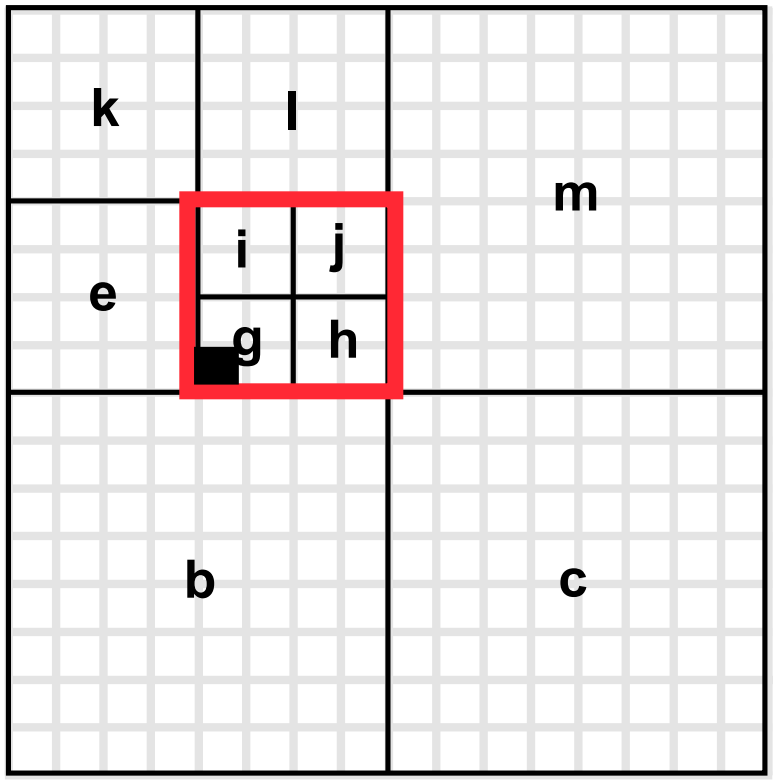
Domain Representation



Tree Representation



Computing Addresses of Octants



Morton code: **Maps n-dimensional points to one-dimensional scalars**

Locational code: **Appends an octant's *level* to the Morton code of its left-lower corner**

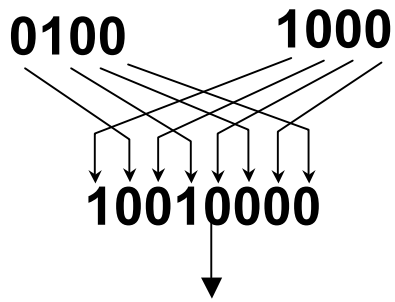
Octant *f*'s left lower corner (4, 8)

↓

Binary form (0100, 1000)

↓

Interleave the bits to obtain non-unique Morton code

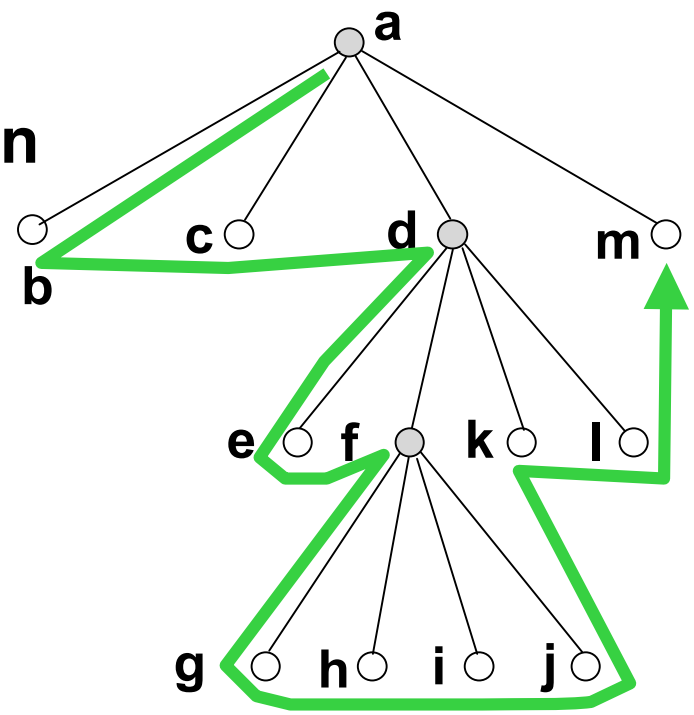


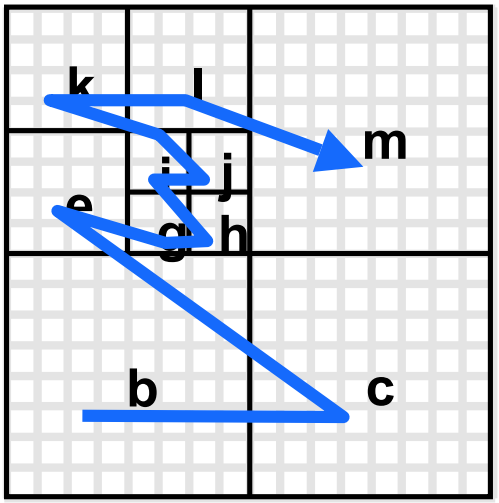
Append level of *f* to obtain *unique* locational code

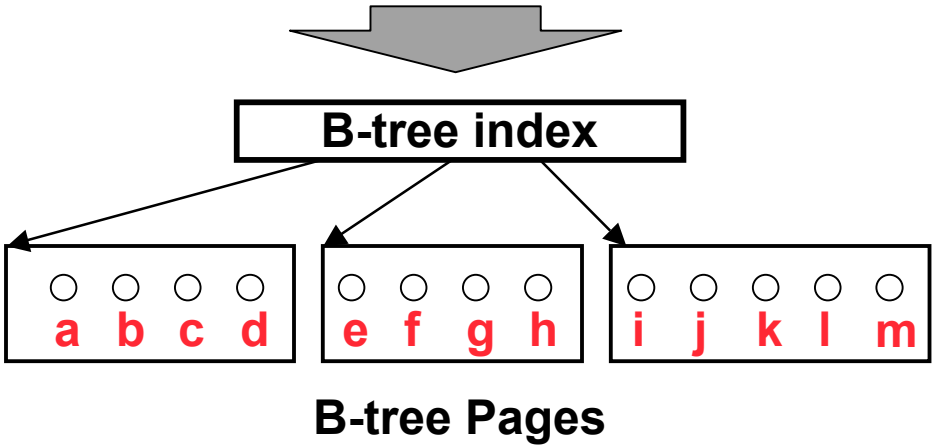
10010000_010

Linear Octrees

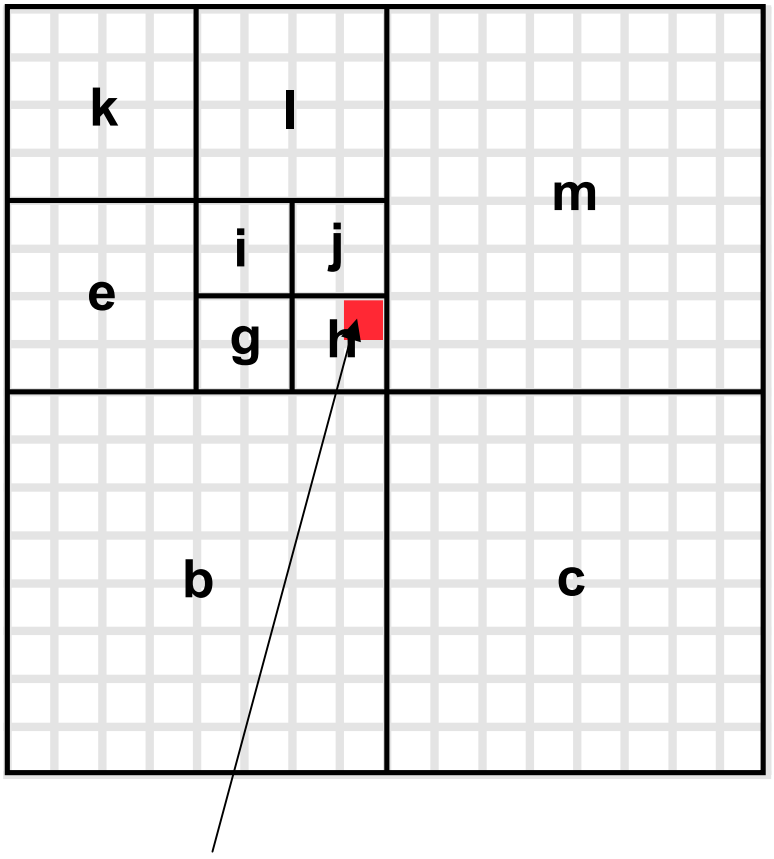
- **Octants are stored on disk in locational-code order**
  - Interior nodes are optional
- **Cool fact: the following are equivalent:**
  - Sorted order on disk
  - Space filling Z-order
  - Preorder tree traversal

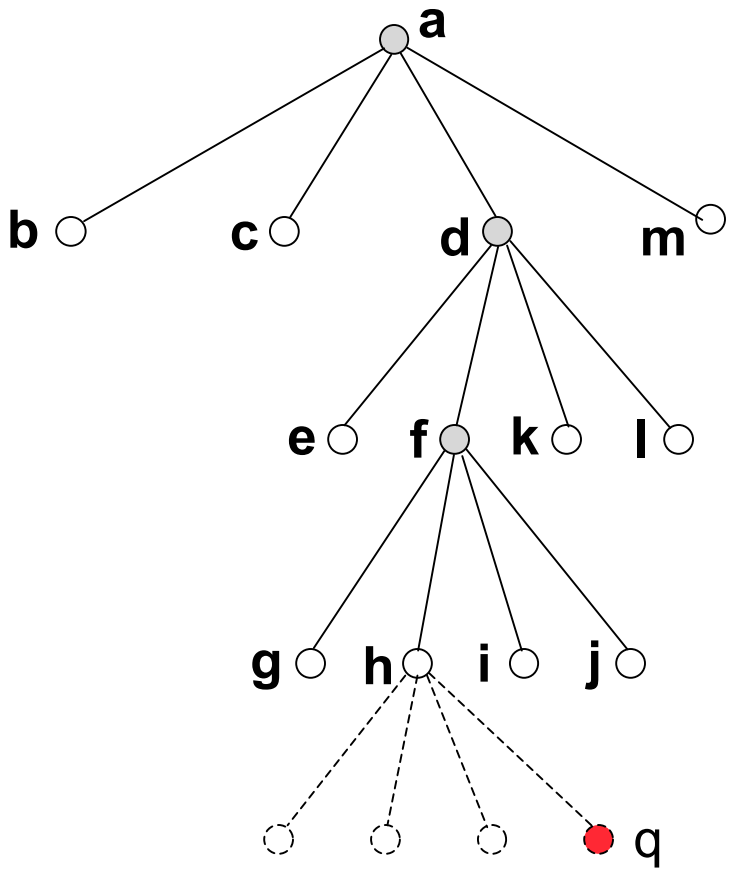






Key Idea: Fast Enclosing Octant Queries

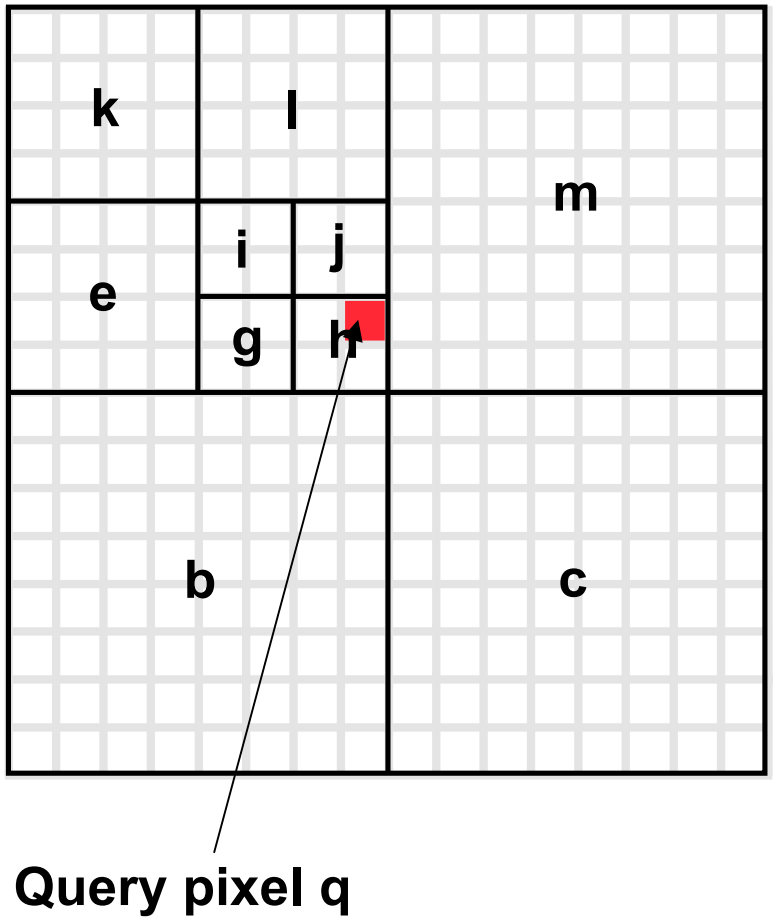




The following powerful fact derives from the preorder property:

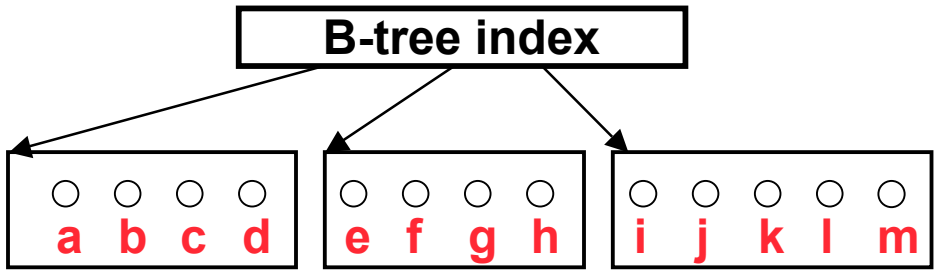
In the linear ordering on disk, h < q < i

Key Idea: Fast Enclosing Octant Queries



Preorder property enables the following fast lookup algorithm for the octant enclosing q

1) **Use B-tree index to find page in log time.**
2) **Binary search for q in page:**
   a) **If found return q**
   b) **If not found return largest p such that p < q**



B-tree Pages

Using Octrees for CVMs

Basic idea: Each octant represents a region of the earth with uniform material properties (plus/minus some epsilon)

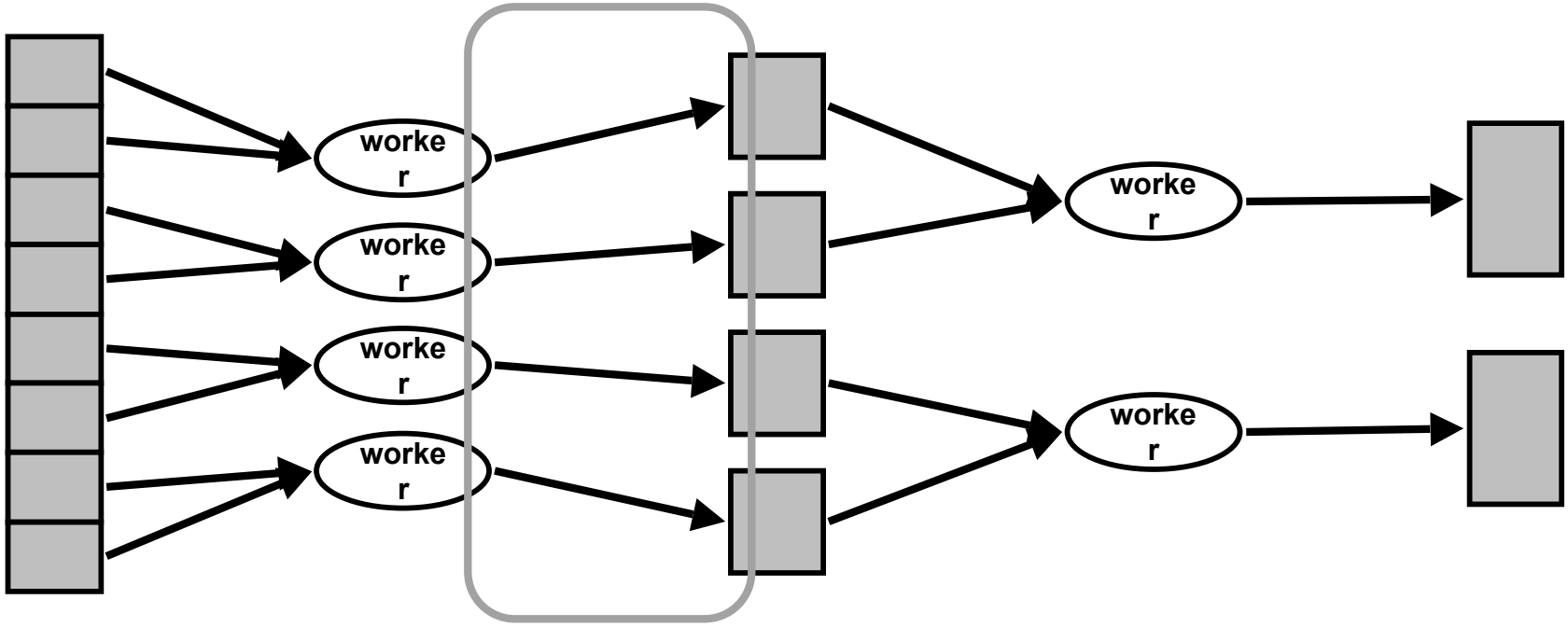
Construction outline:

1. **Sample entire region at desired spatial resolution (e.g. 200m)**
   - **Creates a complete octree (regular grid)**
2. **Iteratively aggregate octants bottom up, one later at a time.**

Key Issue: Initial regular grid before aggregation could be terabytes or petabytes

Solution: *GroundMR:* Parallel cluster algorithm based on Open source Hadoop version of Google's map-reduce framework

Map/Reduce Primer



Input tuples (k, v) | ***Map* phase** | **Intermediate tuples (k', list(v'))** | ***Reduce* phase** | **Output (k'', v'')**

Map tasks
input tuple

Intermediate tuples are partitioned into buckets by an optional *Partition* function, usually a hash

ion to
final output

GroundMR strategy

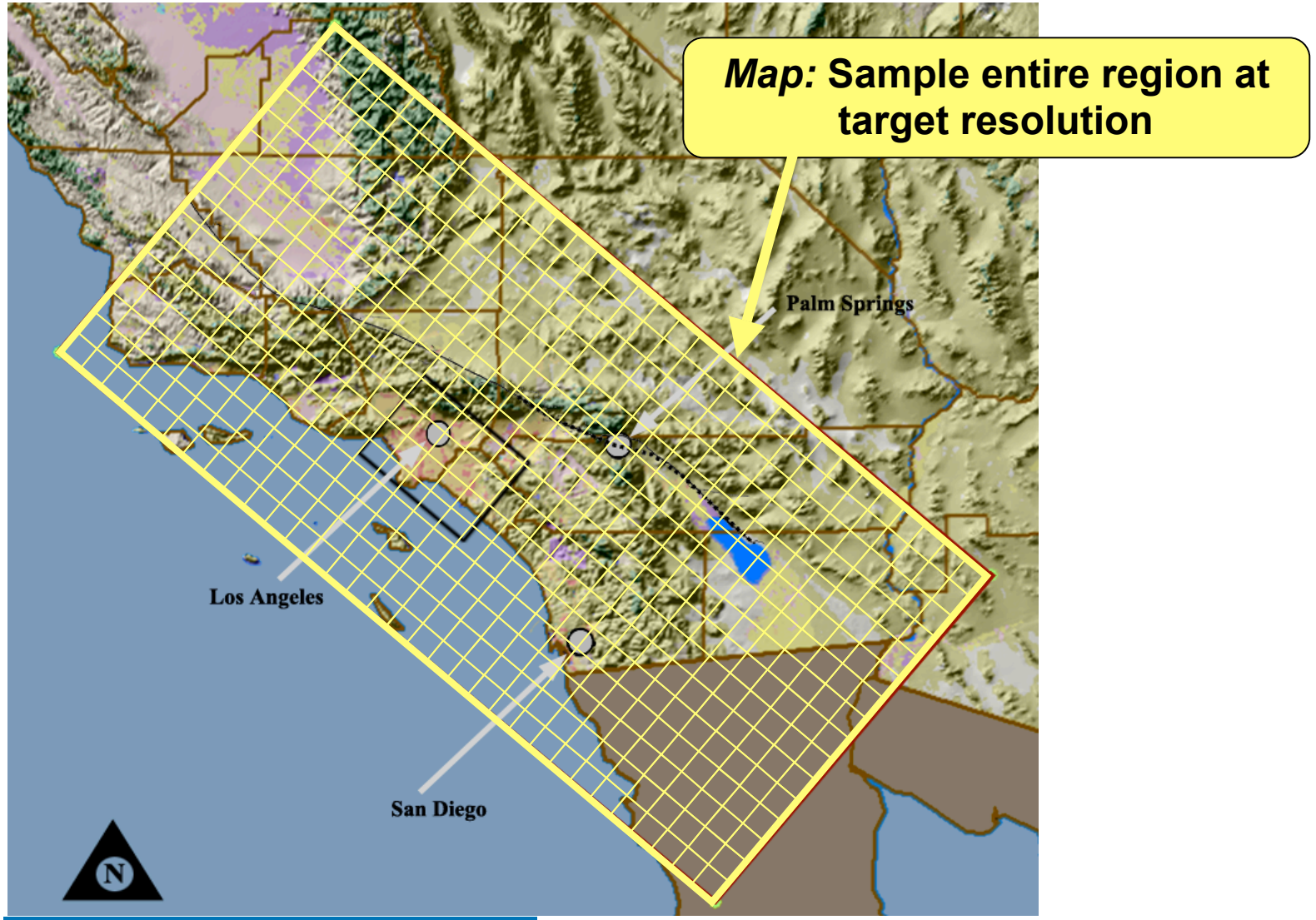


Image credit: Amit Chourasia, Visualization Services, SDS

GroundMR strategy

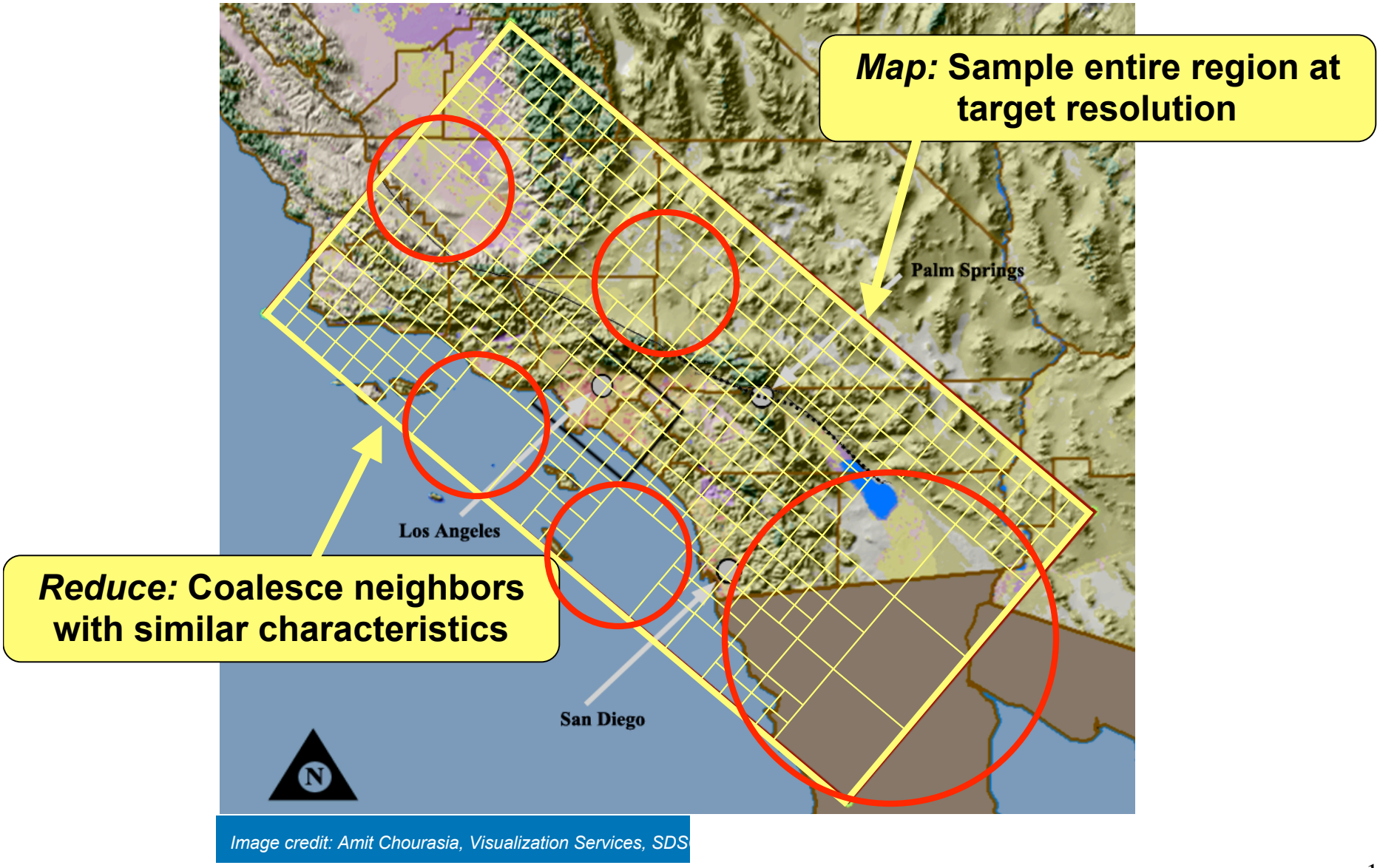


Image credit: Amit Chourasia, Visualization Services, SDS

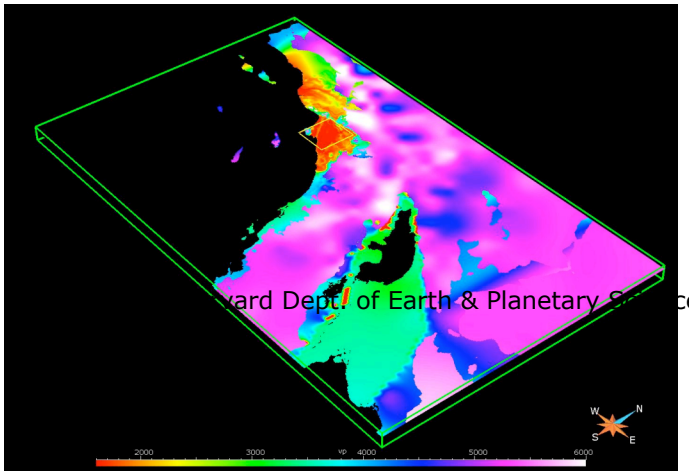
GroundMR Performance

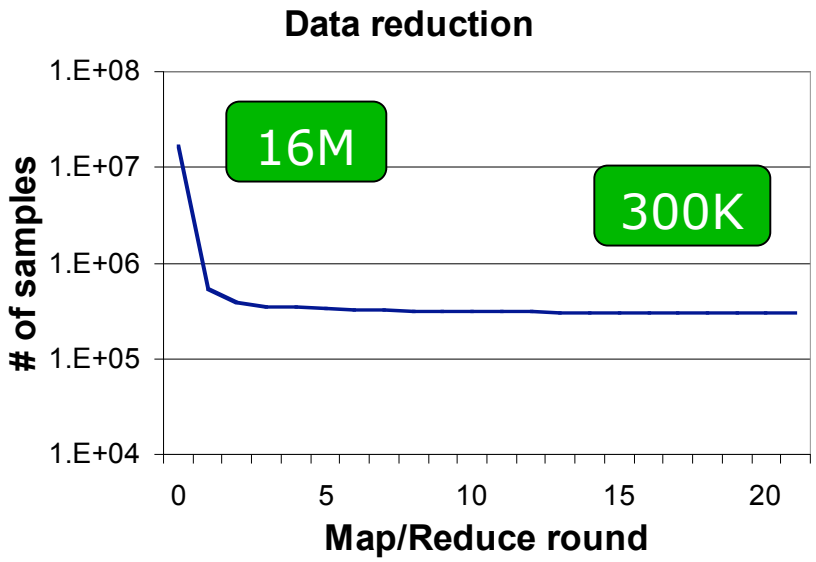
100 x 100 x 100km at 100m spacing

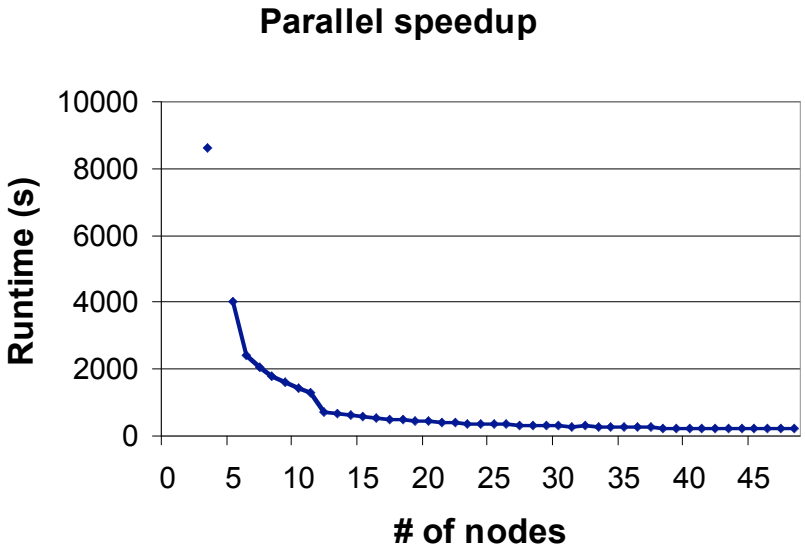
- **~10 minutes on 48 nodes**

Well-suited for Map/Reduce

- **>50X data reduction after coalescing**
- **Linear parallel speedup**







Using Octrees for FEM Meshes

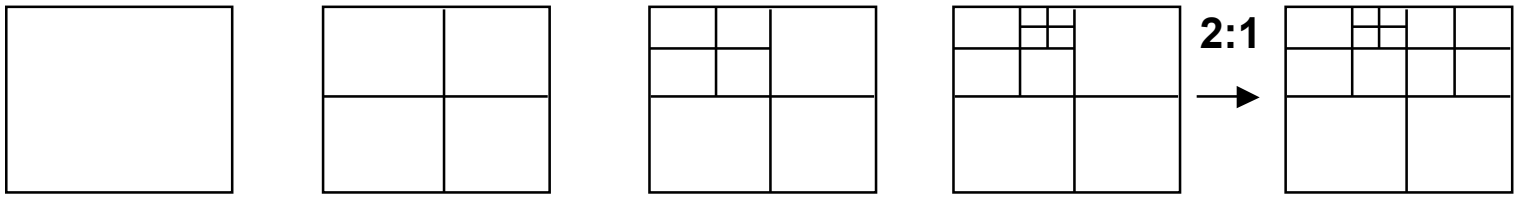
Basic idea: Each octant represents one element

Construction outline:

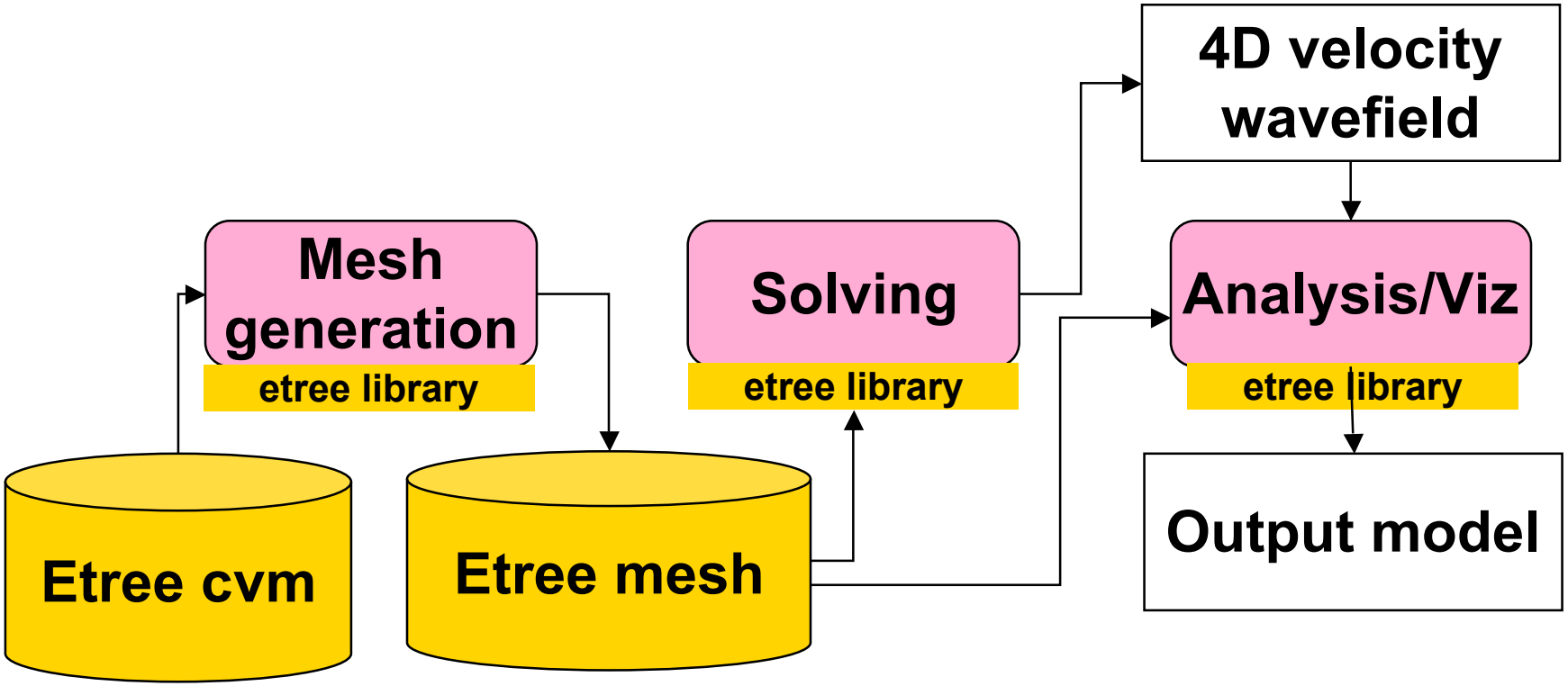
1. **Sample material properties at the corners of octant**
2. **If octant diagonal too small, then subdivide**
3. **Adjust neighboring octants to maintain 2:1 constraint**
4. **Recursively apply 1-4**

A key issue: Need fast queries of material model

Solution: indexed *etree* database structures



Key Hercules Idea #2



Key Idea #2: Each octree input data set is an *etree* database

The Etree Library

- C library with 22 functions for manipulating large linear octree files (called *etrees*) on disk
  - Opening and closing etrees
  - Inserting and deleting octants and subtrees
  - Support for aggregate queries on internal nodes (Brad Aagaard)
  - Searching for octants
  - Traversing octants in Z-order (preorder traversal)
  - Loading and storing application metadata
- Supports 3D or 4D spatial data sets
- Portable file format
  - Standard header characterizes byte ordering and data sizes
  - Text trailer contains arbitrary application-level metadata
- Code and documentation available on the Web
  - `www.cs.cmu.edu/~euclid`

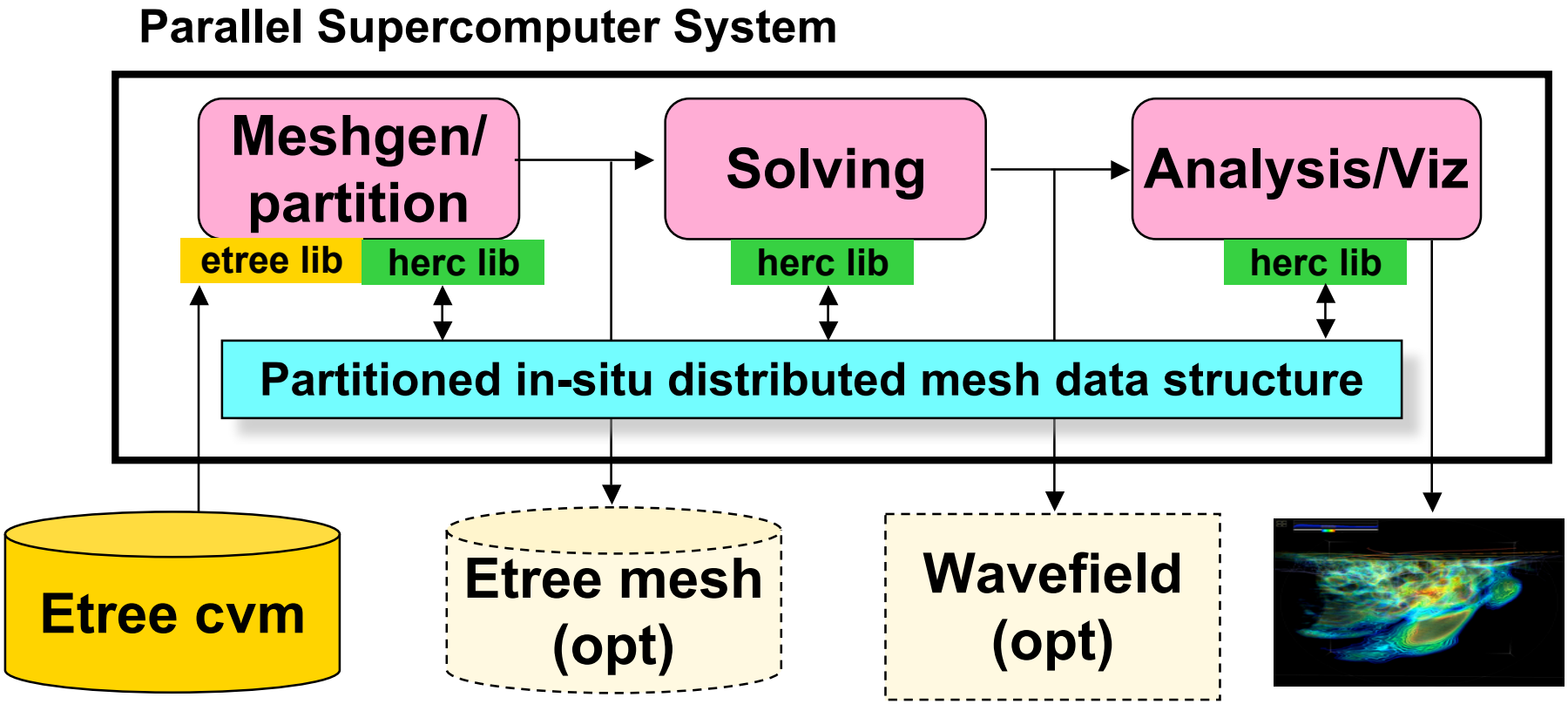
Community Applications of Etrees

Brad Aagaard (USGS)

- Wrote a query package (based on the open source PROJ4 library) that can query etree velocity models in either lat/lon/depth or UTM/depth. Used by SF06 simulation project:
- PROJ4 library:
  - http://proj.maptools.org
- Brad's etree query library and an etree version of the Central California Velocity Model (cencalcvm):
  - ftp://ehzftp.wr.usgs.gov/baagaard/cencalvm/
- Additional info and manual pages for cencalcvm available at:
  - http://www.sf06simulation.org/geology/velocitymodel/querydoc/index.html

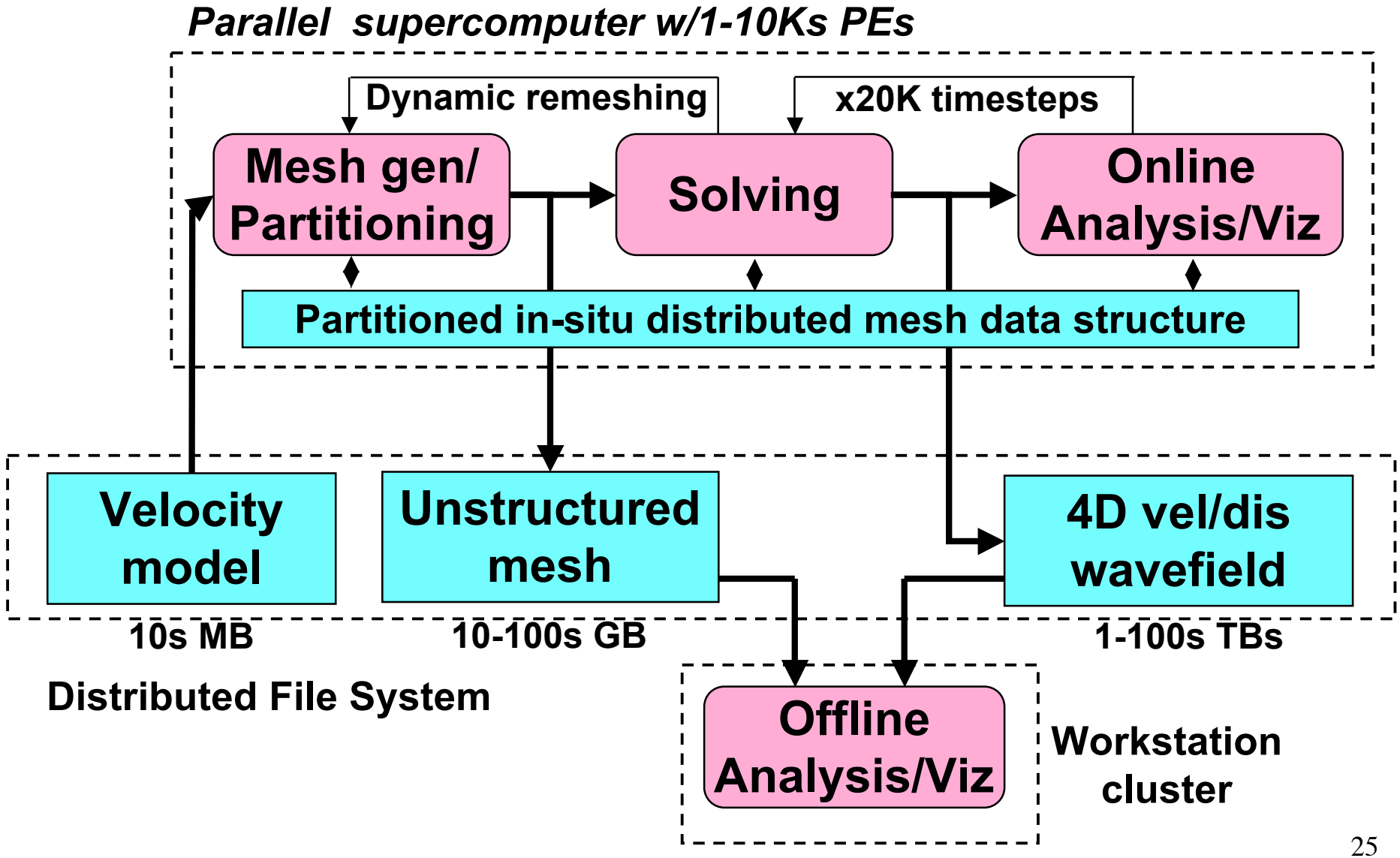
Artie Rodgers (LLNL)

Key Hercules Idea #3



Key Idea #3: Run entire finite element simulation (mesh generation, solving, and visualizing), in place and in parallel (*end-to-end simulation*) [Tu, O'Hallaron, Ghattas, SC04, Tu et al, SC06, Analytics Challenge Winner, Tu et al, SC06]

End-to-end Simulation



Hercules Performance (Tu et al, SC06)

***Scenario*: LA Basin (100km x 100km x 37.5 km), SCEC CVM Version 2.0; minimum shear wave velocity 100m/s, Lemieux at PSC**

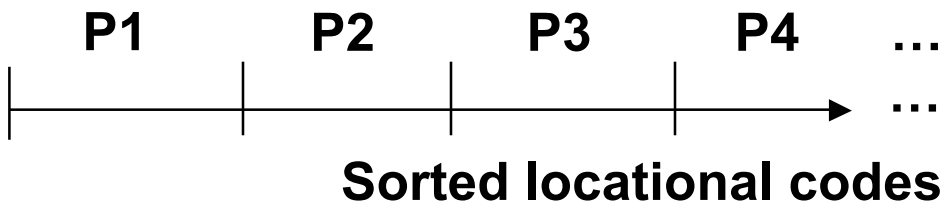
Processors	**1**	**16**	**52**	**184**	**748**	**2000**	
Frequency (Hz)	0.23	0.5	0.75	1	1.5	2	
Elements	6.61E+5	9.92E+6	3.13E+7	1.14E+8	4.62E+8	**1.22E+9**	1
Nodes	8.11E+5	1.13E+7	3.57E+7	1.34E+8	5.34E+8	1.37E+9	
Anchored	6.48E+5	9.87E+6	3.12E+7	1.14E+8	4.61E+8	1.22E+9	
Hanging	1.63E+5	1.44E+6	4.57+6	2.03E+7	7.32E+7	1.48+8	
Max leaf level	11	13	13	14	14	15	
Min leaf level	6	7	8	8	9	9	
Elements/PE	6.61E+5	6.20E+5	6.02E+5	6.20E+5	6.18E+5	6.12E+5	
Time steps	2000	4000	10000	8000	2500	2500	
E2E time (s)	12911	19804	38165	48668	13033	16709	
Replication(s)	22	71	85	94	187	251	
Meshing(s)	20	75	128	150	303	**333**	1
Solver(s)	8381	16060	31781	42892	11960	16097	
Vis(s)	4488	3596	6169	5528	558	*	3
E2E t/ts/e/PE(µs)	**9.77**	**7.98**	**7.93**	**7.86**	**8.44**	**10.92**	2
Sol t/ts/e/PE (µs)	**6.34**	**6.48**	**6.60**	**6.92**	**7.74**	**10.52**	
Mflops/sec/PE	569	638	653	655	*	*	

An example of how octrees simplify the design of a parallel end-to-end simulation

Key parallel computer problem: Given a global element id E, which processor P owns E?

Octree solution:

- **Sort elements by locational code order (space filling z-order).**
- **Assign each processor a contiguous chunk of sorted elements.**
- **Compute a small interval table with min and max locational codes on each processor.**
- **Mapping from element to processor performed by binary search of interval table.**



Future Directions: Standard Velocity Model Representations

Proposal: **Adopt etree database as the standard queryable representation for velocity models and meshes**

- Queryable CVM required for any unstructured FE code
- Queryable mesh required to index and query FE output wavefield.

Proposal: **Develop parallel tool for automatic generation of sampled etree fields**

- Quantize and aggregate given some global error constraint
- Standalone program
- Arbitrary field function dynamically linked from shared library (dll)

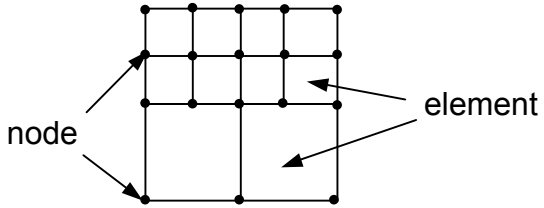
Conclusions

1. Octree meshes are an intriguing compromise between structured and unstructured meshes

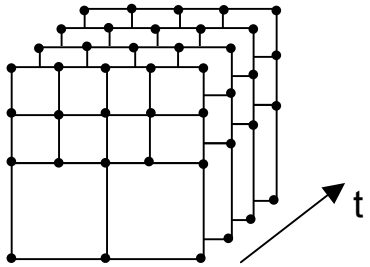
2. We should be using time and space efficient *queryable* database structures such as etrees for our velocity models and meshes:
   - Domain sizes and spatial resolutions are increasing
   - Fast queryable CVMs are required for FEM mesh generation
   - Fast queryable meshes are required to index and access FEM output wavefield.

Unused slides

The Physical Simulation Process



mesh



simulation results

Galerkin discretization

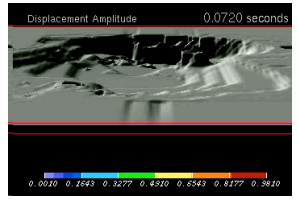
Numerical Model

FEM solver

Computer Model

Animation

Visualization Model

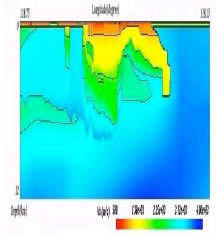


Physical System

Earthquake ground motion in San Fernando valley

Physical Model

Velocity model



Mathematical Model

Wave propagation equation

$$\nabla \cdot \left[\mu \left(\nabla \vec{u} + \nabla \vec{u}^{T} \right) + \lambda \left(\nabla \cdot \vec{u} \right) \vec{I} \right] = \rho \frac{\partial^2 \vec{u}}{\partial t^2}$$

Future Directions: "Big Data"

Claim #1: In the future, the most interesting parallel applications will involve manipulating and analyzing big data.

Examples:

- **Generating unstructured hexahedral and tetrahedral meshes**
- **Understanding 4D fields generated by physical simulations**
- **Verifying and validating physical simulations**
- **Processing medical images**
- **Real time sensor inferencing**
- **Mining and processing crawled web data**

Future Directions in Software: Computational Database Systems

Claim #2: Big data will be manipulated by *computational database systems* *(Cods)* that couple database queries with complex operations.

Properties of Cods:

- **All data stored in queryable indexed databases**
  - **"Flat files don't cut it anymore!"**
- **Manipulate big data by querying and updating databases**
- **Queries produce derived databases**
- **Data grows and evolves**

Etree cvms and meshes are just a start

- **We need a similar capability for wavefields**

Future Directions in Machines: Interactive Superclusters

Claim #3: Big data will be hosted and manipulated on interactive data-intensive superclusters, rather than traditional batch supercomputers.

Implications:

- **Superclusters are essentially storage devices with massive parallel computational capability**
- **Emphasis on response time rather than throughput**

Recent Google/IBM announcement on "cloud computing" for academic research a good start.