Distributed Geometric Data Structures

Philip Levis
Stanford Platform Lab Review
Feb 9, 2017

Big Control

The Physical World

- Big control applications collect data on, and take action in, the physical world
  - There will be a lot of data: they need distributed data structures to store, query, and compute on it
- Big control applications have high locality (literally)
  - Physical world data is geometric (2D, 3D) in nature; it has much more complex data inter-dependencies than key-value stores
- Need new, distributed geometric data structures

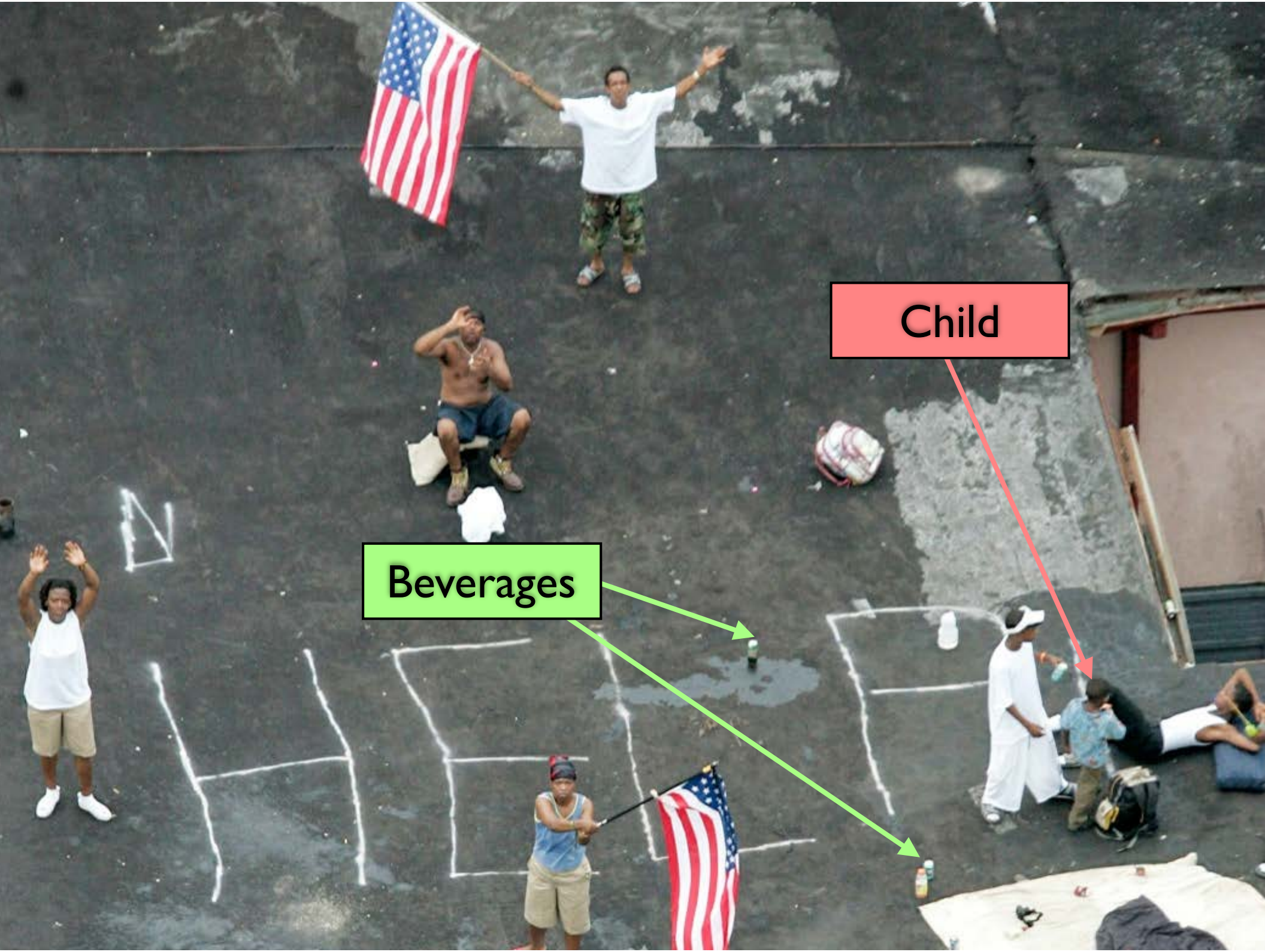
Outline

- Example big control applications
- Geometric data structures
- Distributing geometric data structures

Outline

- Example big control applications
- Geometric data structures
- Distributing geometric data structures


N
HERE


Child
Beverages

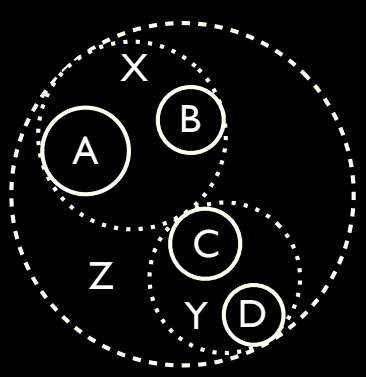
Data Requirements

- Altitude produces different data resolutions
  - Dynamically changing in response to application actions
- Data changes/decays over time: 4D
- Grid-based (e.g., temperature, pixels) as well as point-based data (people, objects, landmarks)

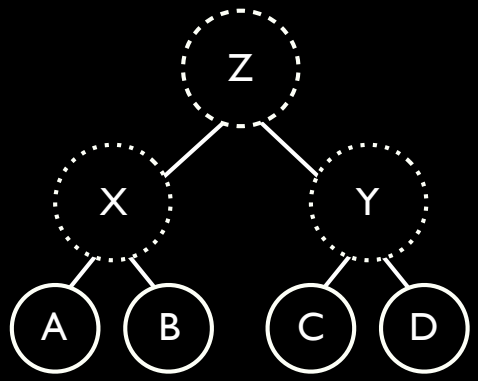
Outline

- Example big control applications
- Geometric data structures
- Distributing geometric data structures

Two Basic Approaches

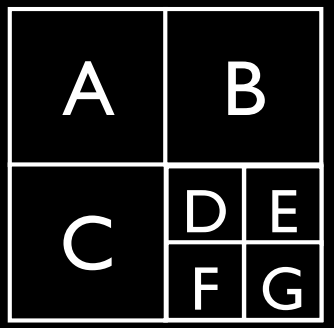


geometry

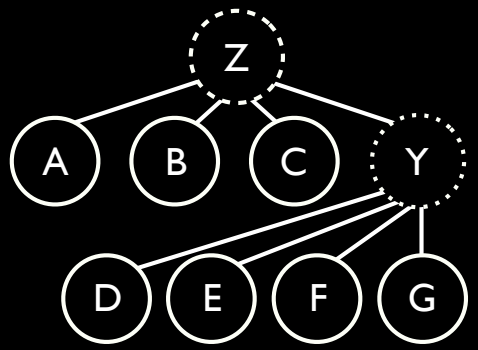


data structure

Bounding Volume Hierarchy (BVH)



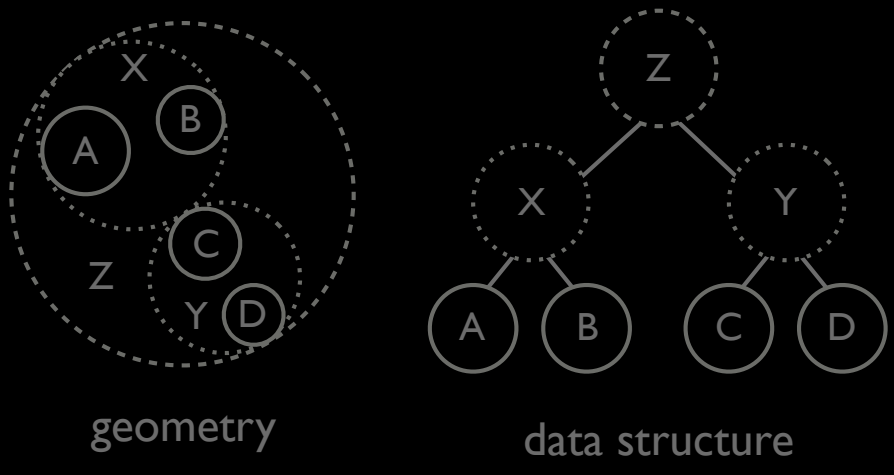
geometry



data structure

Spatial subdivision

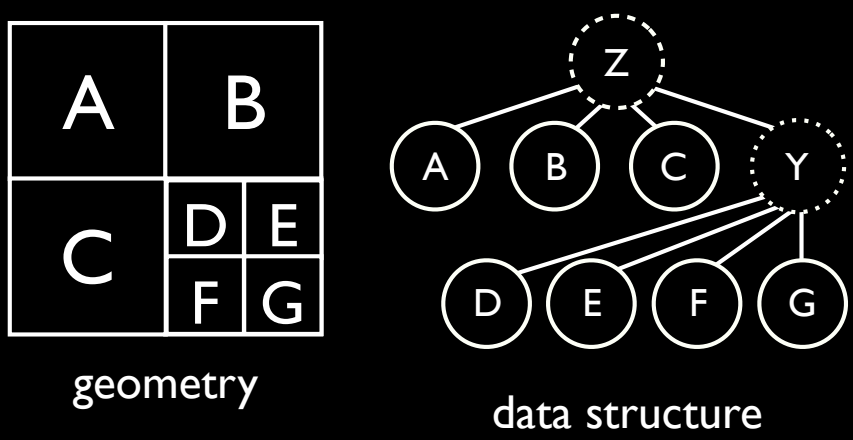
Two Basic Approaches



geometry

data structure

Bounding Volume Hierarchy (BVH)



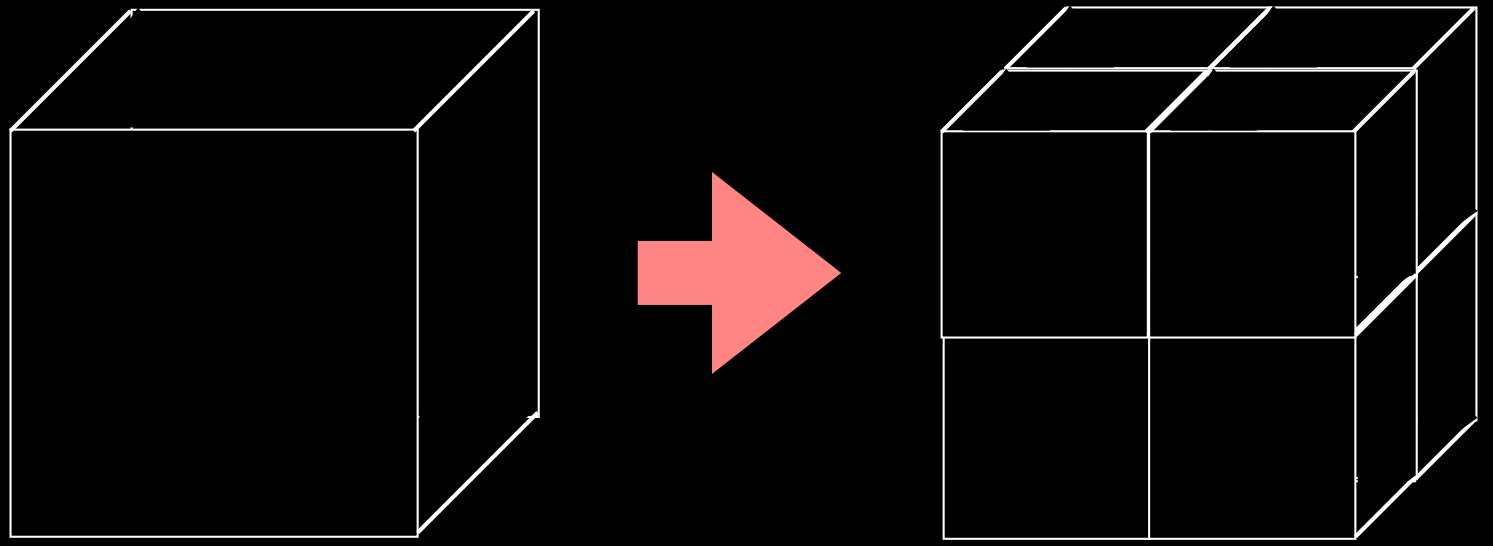
geometry

data structure

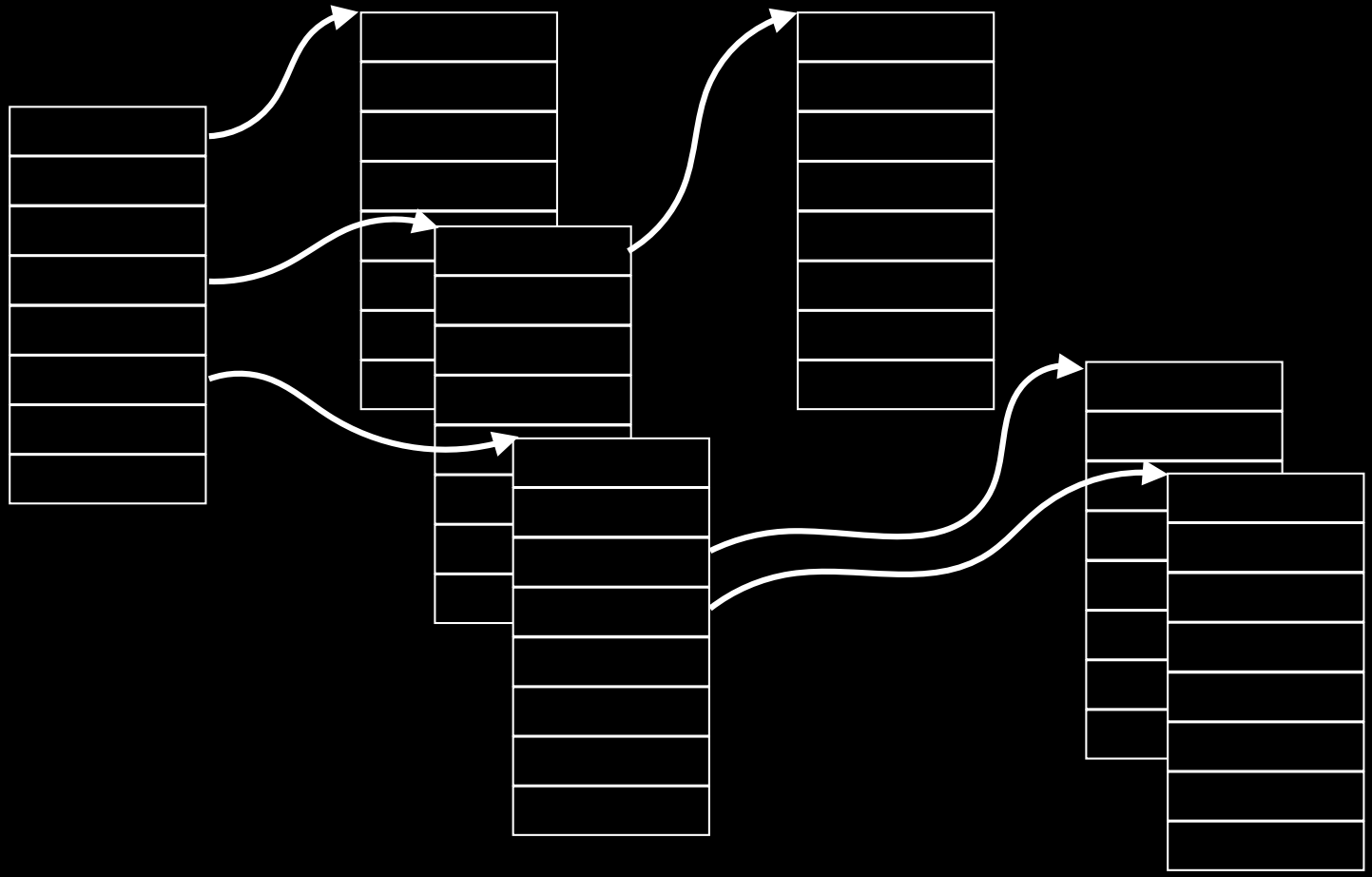
Spatial subdivision

Spatial subdivision

- Many variants: quad/oct-trees, kd-trees, binary space partitioning
- Oct-tree: subdivide each axis evenly

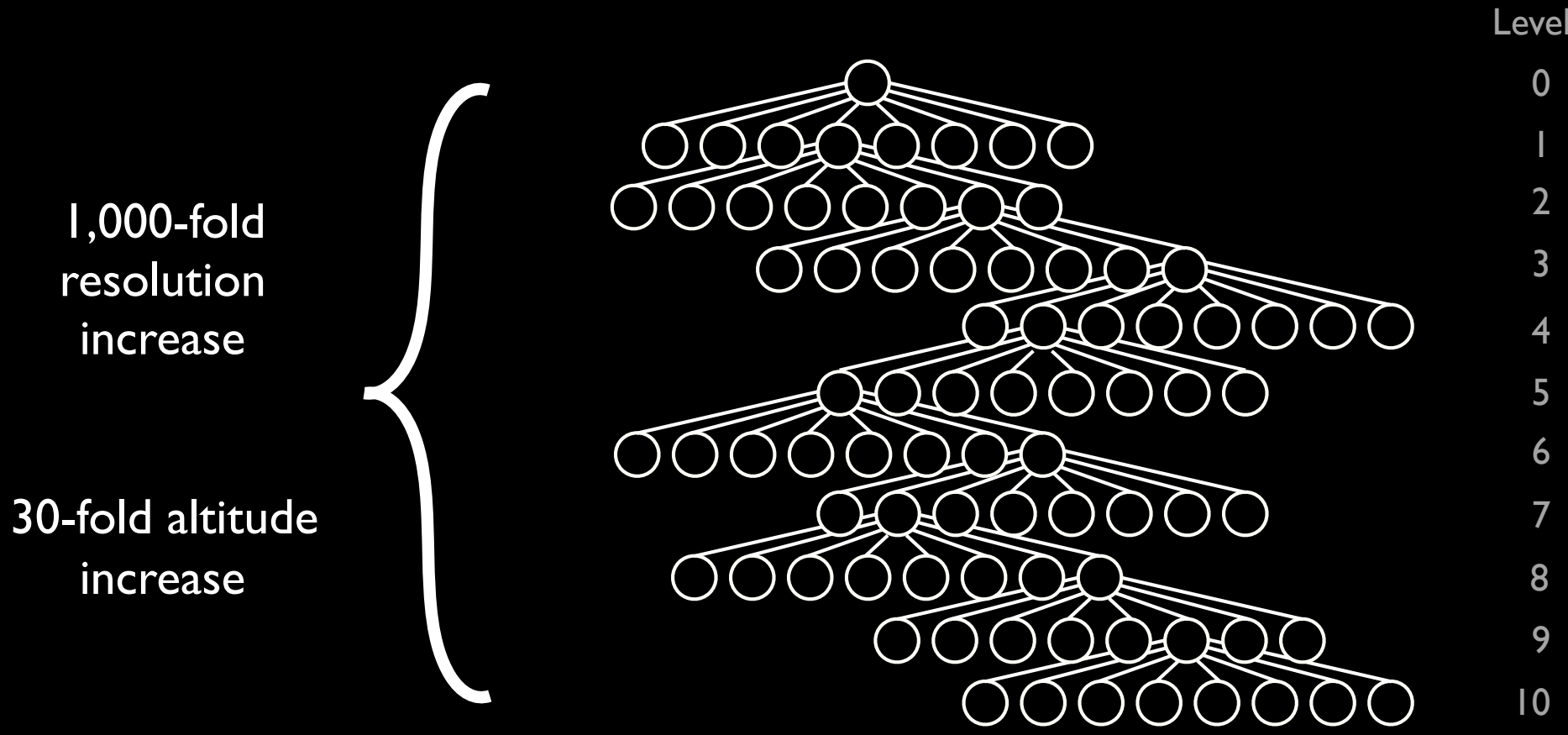


Problem with Oct-trees



Sparse, pointer structure: low locality, cache-poor

Another Problem

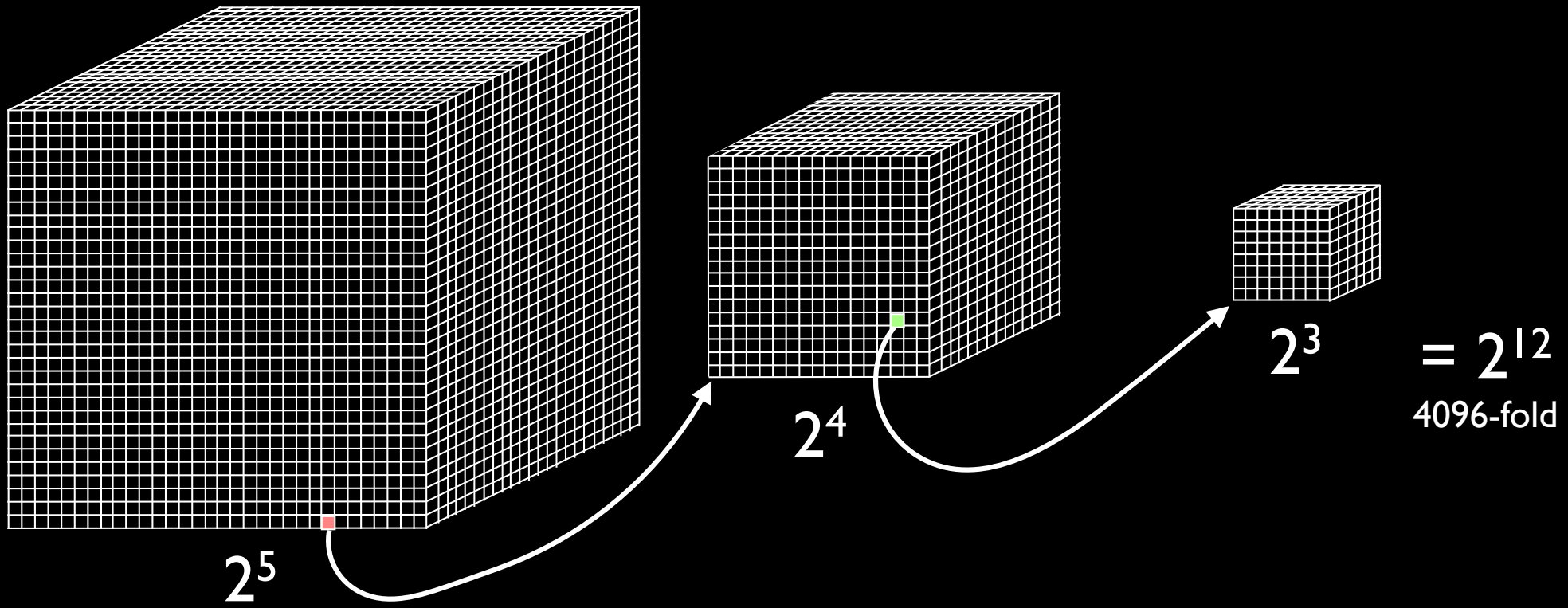


Many levels for large variations in resolution

VDB

(Museth, ACM TOG 2013 Vol 32, 3:27)

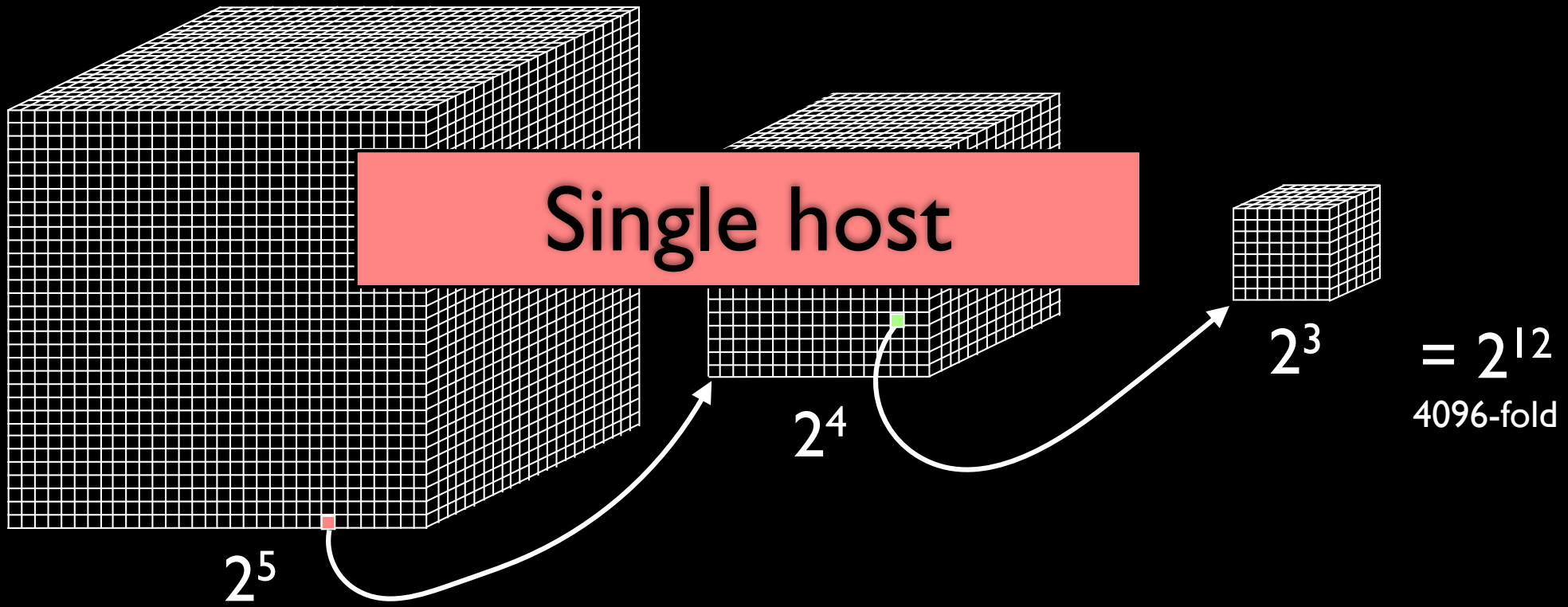
- Hierarchical data structure for the efficient representation of sparse, time-varying volumetric data discretized on a 3D grid



VDB

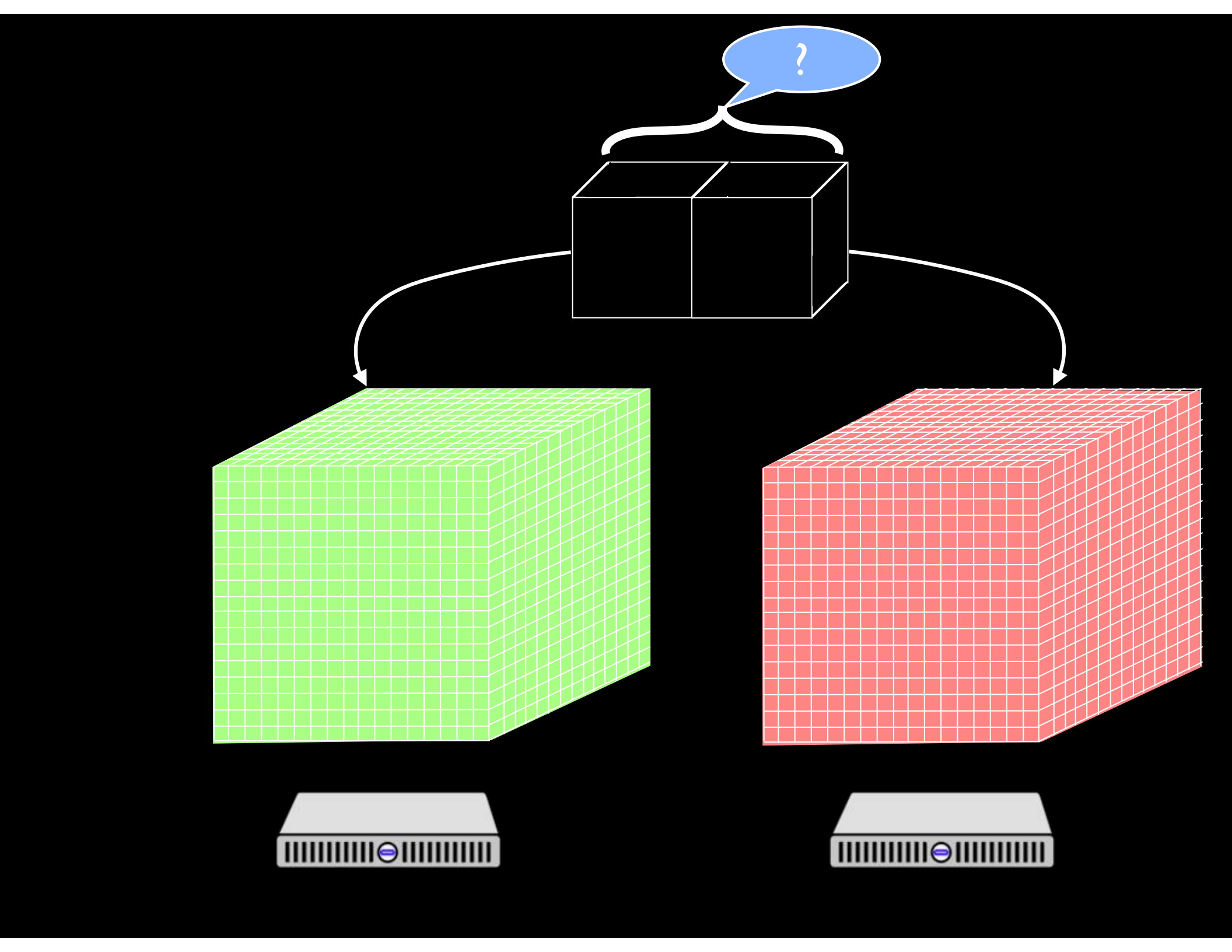
(Museth, ACM TOG 2013 Vol 32, 3:27)

- Hierarchical data structure for the efficient representation of sparse, time-varying volumetric data discretized on a 3D grid



Outline

- Example big control applications
- Geometric data structures
- Distributing geometric data structures

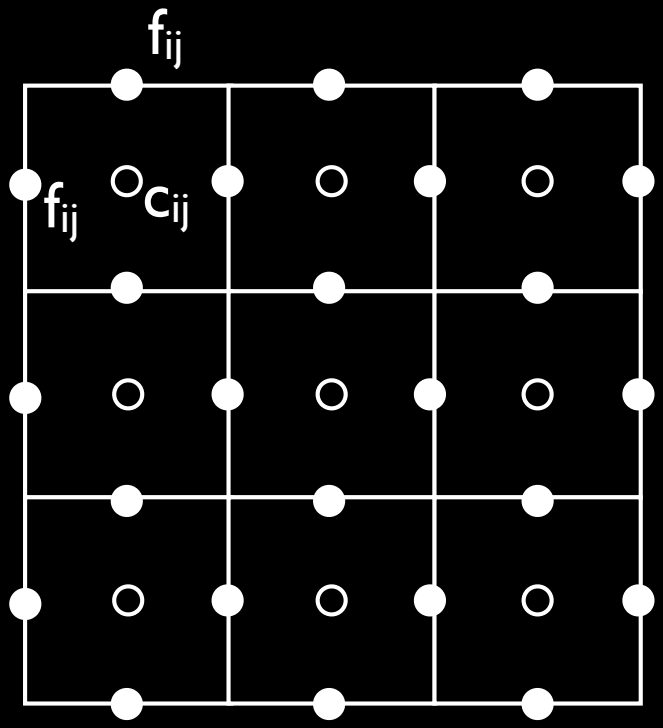


Problem 1: Distribution

- Where should the system place each tree node?
- Option 1: Random/spray placement
  - Improves potential read bandwidth
  - Balances load easily
- Option 2: Locality-based placement
  - Better when computations pushed to data (granular computing)
  - Load balancing is an open problem
  - Strawman: balance data size (assumes uniform computation)

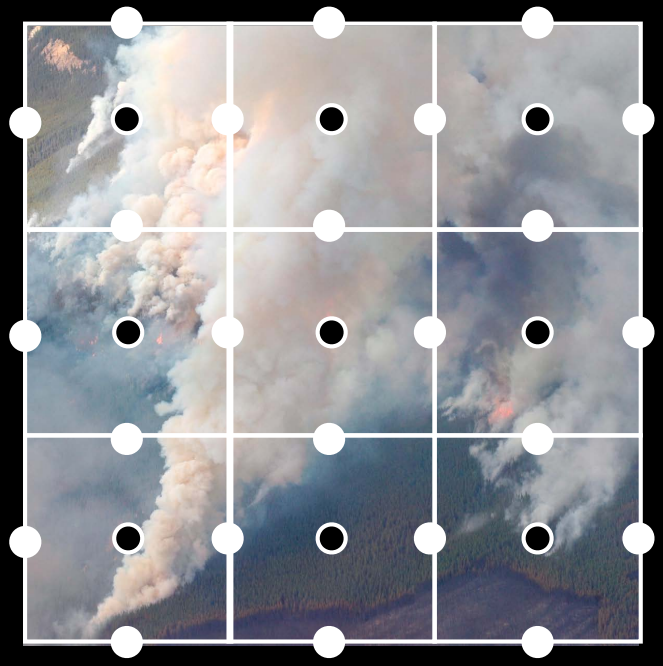
Problem 2: Not So Simple

- Staggered (MAC) grids store data on faces as well as in cells: important to represent flow



Problem 2: Not So Simple

- Staggered (MAC) grids store data on faces as well as in cells: important to represent flow

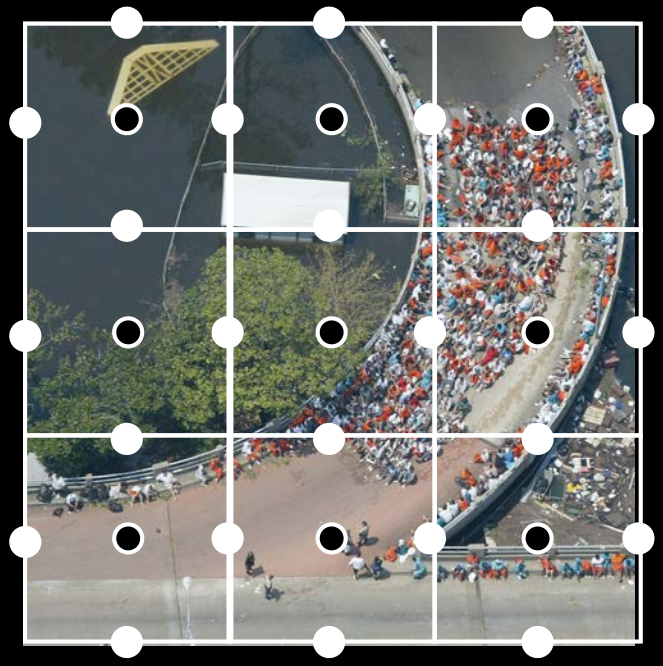


c_{ij} -temperature, burn state, etc.

f_{ij} - wind

Problem 2: Not So Simple

- Staggered (MAC) grids store data on faces as well as in cells: important to represent flow

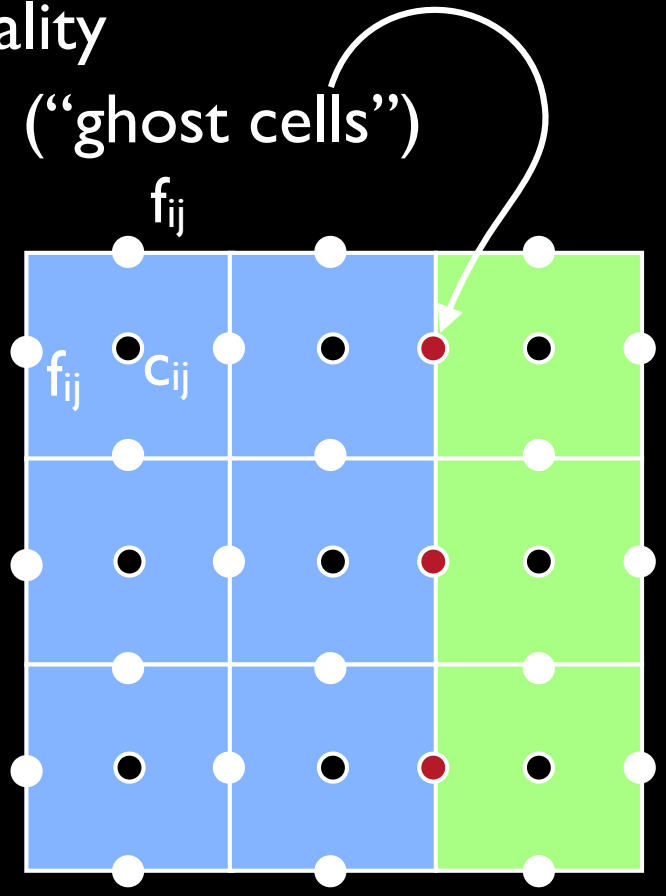


c_{ij} - # of people

f_{ij} - movement of people

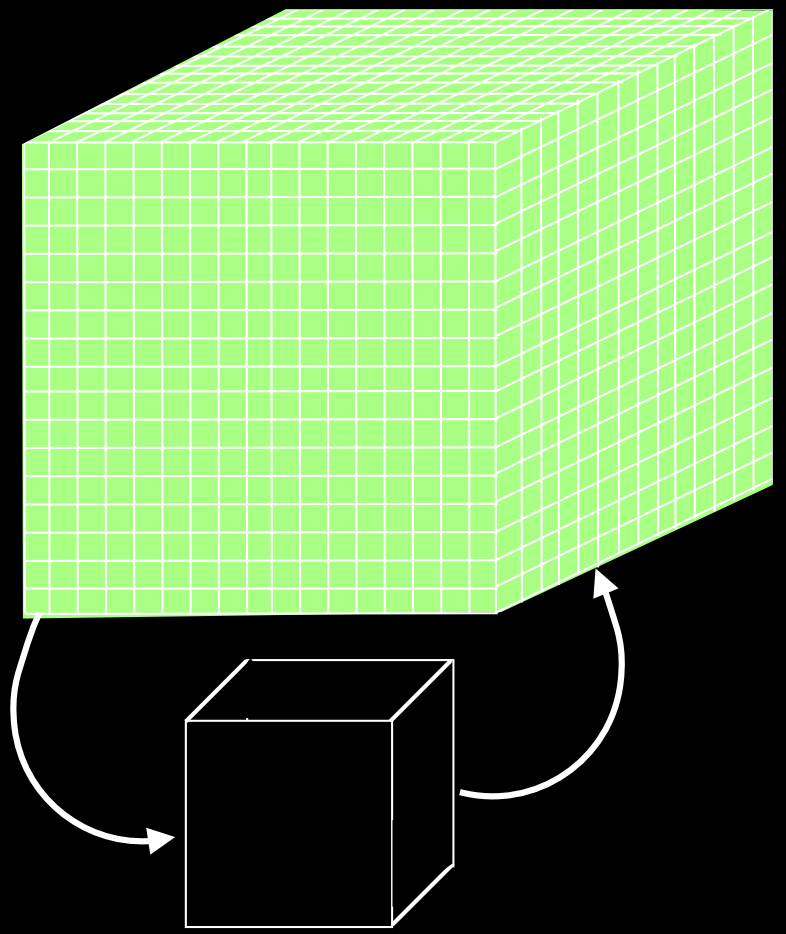
Problem 2: Not So Simple

- Cell and face values should have locality
- Distributed grids require replication ("ghost cells")
- Minimizing surface area of volumes minimizes communication, but complicates load balancing



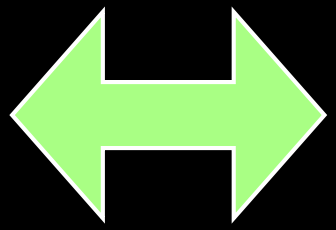
Problem 3: Dynamic Updates

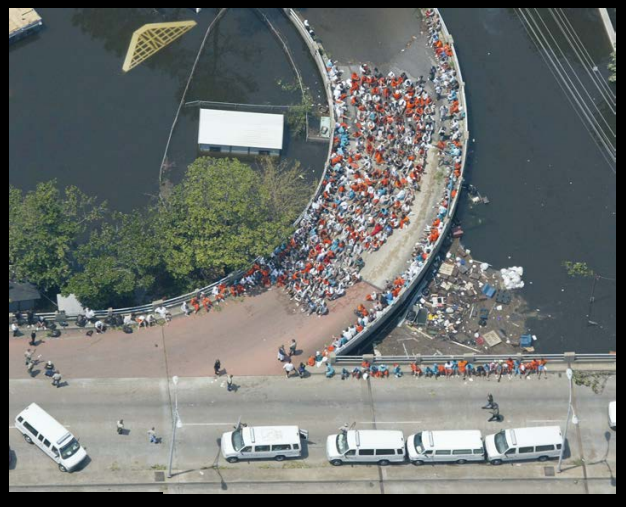
- Applications will dynamically subdivide and coarsen the data structure
- Operations may trigger load rebalancing: need to mask latency from application (asynchrony/ replication)



Problem 4: Time

- Big control applications require being able to look backwards in time
  - Where did those people needing rescue go?
  - Where did the fire jump the fire break?
  - What is traffic downtown like in 15 minutes (at 5:30PM)?
- Complicates load balancing: historical data should be close to current data





Current Status

- Understanding bottlenecks/performance issues requires workloads (computations, hierarchy structure)
  - Have implemented multi-resolution FLIP simulation
  - Next step: simulator for drone exploration
- Integrating replication/ghost cells for distribution

Conclusion

- Big control applications use geometric data structures
- Dynamically distributing these data structures is an open problem
- We're starting with space partitioning (have some prior results on BVHs)

Chinmayee Shah

Hilbert Helix