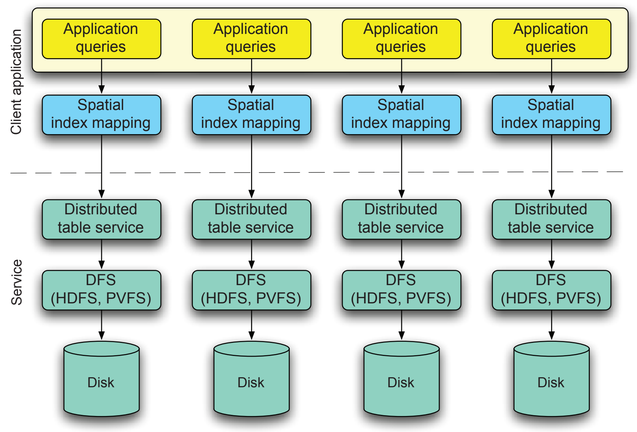
Scalable Spatial Indexing for Data-Intensive Science Applications

Julio López, Bin Fu, Eugene Fink, Garth Gibson

Overview

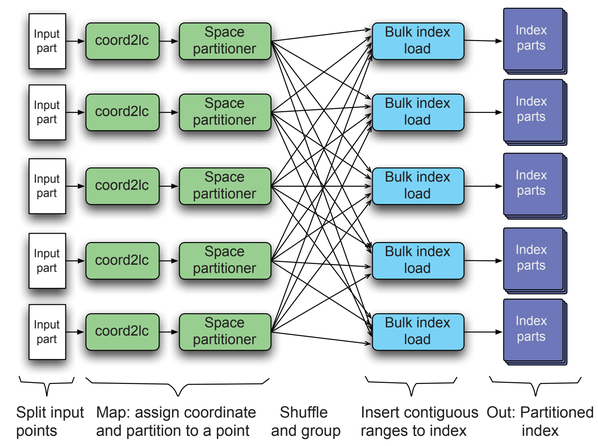
Many data-intensive problems in science rely on spatial data structures to obtain a solution. Our approach is to extend table software, such as HBase, to provide distributed scalable spatial data structures. Our initial implementation linearizes octrees using Morton codes to create a Z-order fill.



Distributed spatial indexing architecture

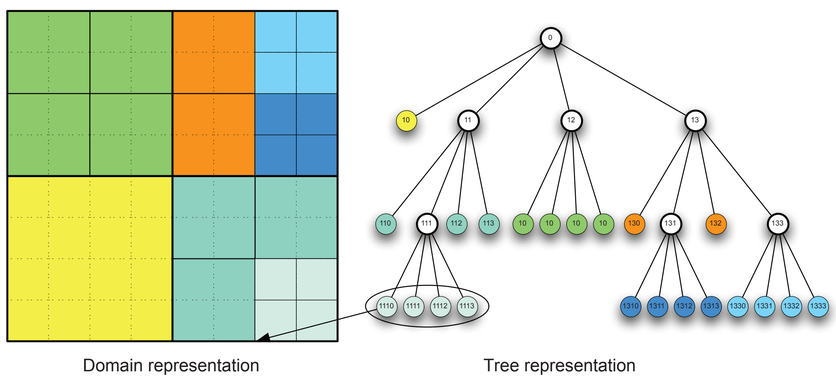
Approach

- Input: point records with location and other attributes
- Use an octree to partition and index the records
- Linearize octree and store in a distributed sorted table
  - Key linearization using Morton codes
  - Z-order space filling curve
  - Sorted table: HBase, Cassandra, Hypertable
- Initial implementation as a client-side library
- Tools for bulk-loading index creation

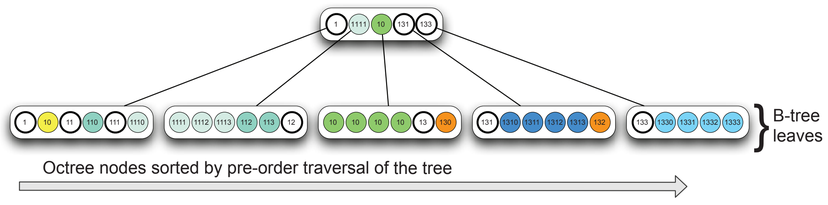


Using Map-Reduce, input points are grouped in spatial buckets and then continuous ranges can be bulk loaded into the index partitions.

Decomposition of the space using an octree. The spatial domain representation (left) has an equivalent tree representation (right). Nodes in the tree are numbered using a pre-fix scheme equivalent to the pre-order traversal of the tree.



The nodes in the octree are mapped to a sorted array that is indexed with a B-tree.



Applications

- Astrophysics:
  - Group-finding (galaxy clustering)
  - Correlation functions
  - Domain partitioning
  - Density distribution function
- Seismology
  - 3D ground model datasets for simulation
  - Indexing of 4D wavefields

Research Questions

- Approaches for index construction:
  - A1: Insert unsorted point records
  - A2: Sort first, then partition and bulk load
- What functionality should be moved to the server?
- New distributed algorithms for data-analytics at scale.
- How can these algorithms leverage the distributed spatial indices?

Carnegie Mellon

Parallel Data Laboratory

R09