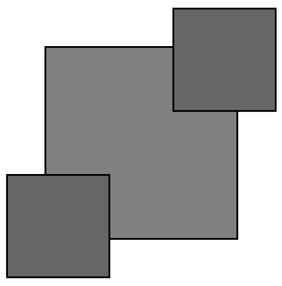
BBoxDB - A Scalable Key-Bounding-Box-Value Store for Multi-Dimensional Big Data

Jan Kristof Nidzwetzki and Ralf Hartmut Güting

Faculty of Mathematics and Computer Science, FernUniversität in Hagen, Germany

BBoxDB



BBoxDB

A Key-Bounding-Box-Value Store

Key-Value Stores

Key-Value Stores . . .

- are a popular type of datastore.
- use a simple key-value data model.
- can be implemented as a distributed system for large amounts of data.
- provide at at least the operations `put(table, key, value)` and `get(table, key)`.
- can not handle multi-dimensional data or non-point data well.

The Problem

```
{
  "1234": {
    "customer_id":1234,
    "firstname":"John",
    "lastname":"Doe",
    "email":"jd@domain.tld"
  }
}
```

Key → 1234

Figure: Determining the key for one-dimensional point data (e.g., a JSON encoded customer).

Figure: Determining the key for two-dimensional non-point data (e.g., a road).

Our Solution

BBoxDB . . .

- is a *key-bounding-box-value store*.
- stores each value together with an axis-parallel *bounding box*.
- can handle n-dimensional point and non-point data.
- splits the space by using a *space partitioner* (e.g., *K-D Tree*, *Quad-Tree*).
- redistributes uneven data distributions dynamically in the background.
- provides a two-level index structure. The *global index* (space → nodes) is stored in ZooKeeper and the *local index* (space → tuples) is stored on each node. The local index is implemented by an *R-Tree*.
- stores data in *string sorted tables* (SSTables).

Multi-Dimensional Shards

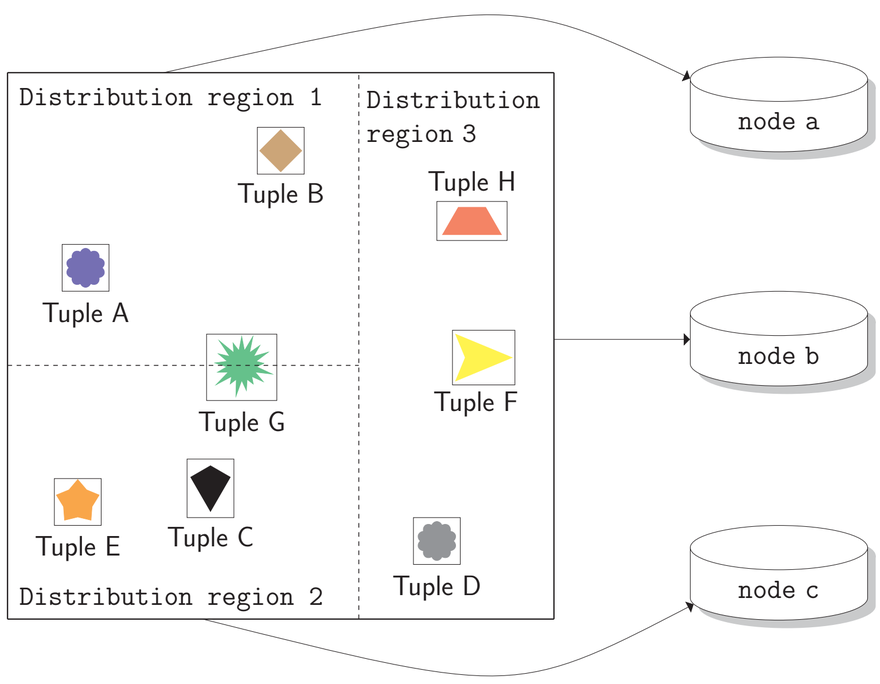


Figure: The space is partitioned into distribution regions. Each tuple is stored together with its bounding box. Tuples that belong to multiple regions are duplicated.

The Most Important Operations

- **Create a distribution group:**
  `createdgroup(group, dimensions, replicas, space partitioner)`
- **Store new data:**
  `put(table, key, hyperrectangle, value)`
- **Retrieve data:**
  `getByHyperrectangle(table, hyperrectangle)`
- **Execute a spatial join:**
  `join(table1, table2, hyperrectangle)`

Spatial Joins on Co-Partitioned Data

- BBoxDB stores all tables of a distribution group *co-partitioned*.
- The data of these tables is distributed in the same manner.
- A spatial join can be executed without any data shuffling between nodes only on locally stored data.

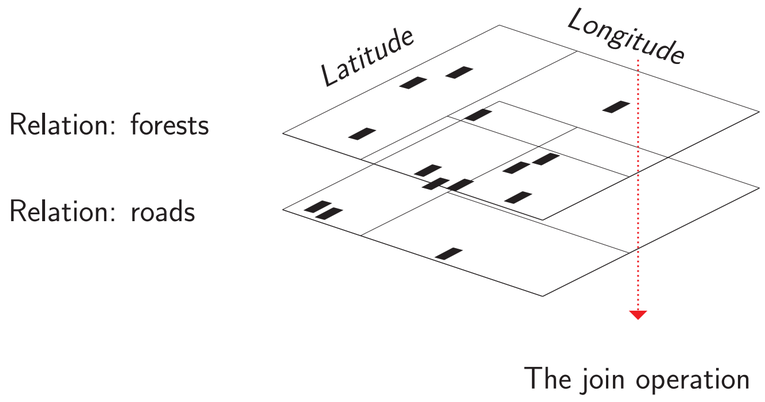


Figure: Executing a spatial join on co-partitioned two-dimensional data.

The Graphical User Interface

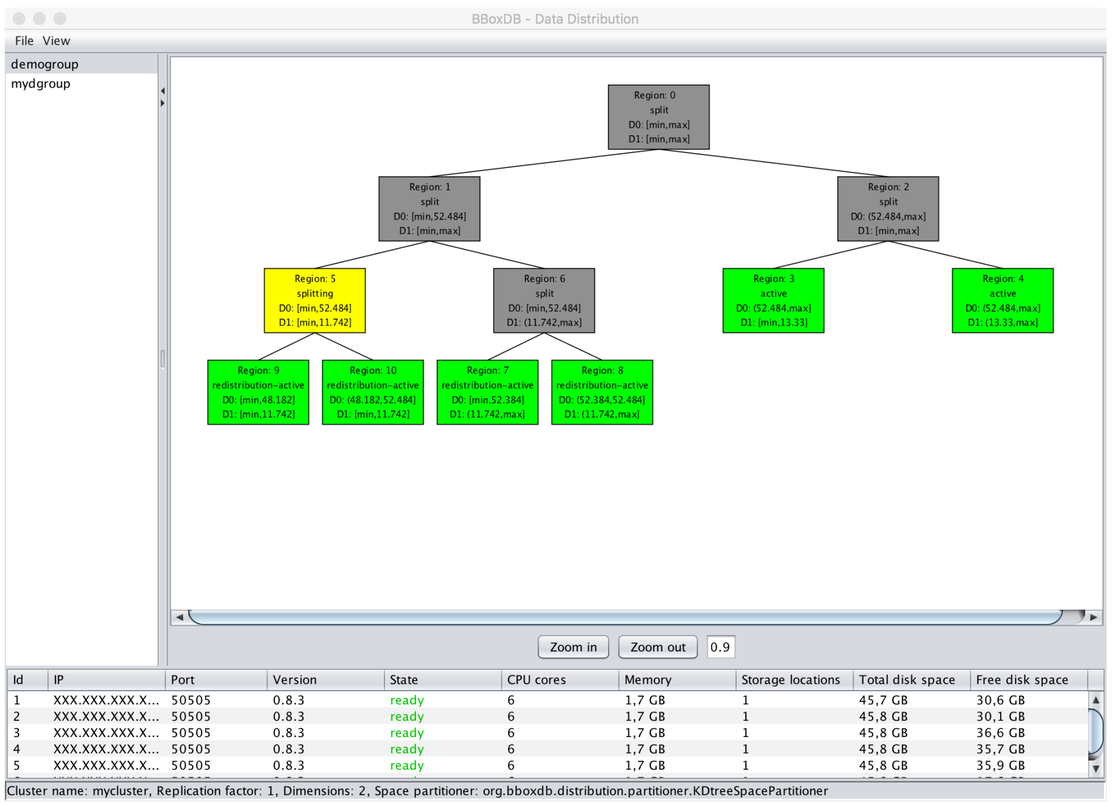


Figure: The GUI of BBoxDB shows the cluster and the global index.

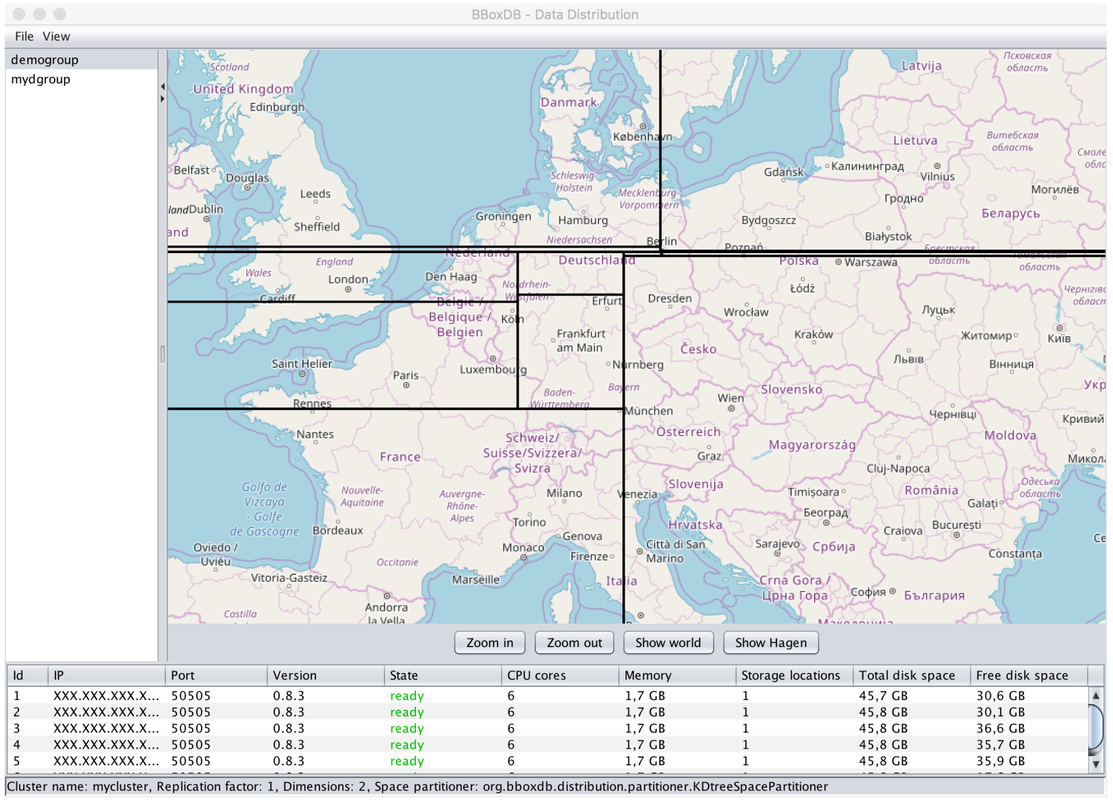


Figure: The global index visualized as Open Street Map overlay.