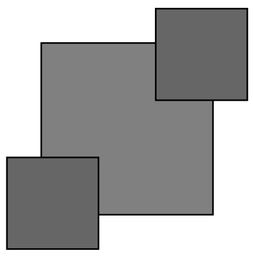
Large Scale Spatial Data Processing With User Defined Filters In BBoxDB

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

Our Software

BBoxDB ...

- is a distributed *key-bounding-box-value store*.
- can handle *n*-dimensional point and non-point big data.
- stores each value together with a *bounding box*. The bounding box determines the location of the value in the *n*-dimensional space.
- partitions the space dynamically and redistributes the data.
- is freely available and licensed under the *Apache 2.0* license.
- is a generic datastore; values are plain arrays of bytes. The semantics of the stored values are unknown.
- **performs operations (e.g., range queries or spatial joins) only on the bounding boxes of the data.**

The Most Important Operations

- **Store new data:**
  `put(table, key, hyperrectangle, value)`
- **Retrieve data:**
  `getByRange(table, hyperrectangle, udf, udf value)`
- **Execute a spatial join:**
  `join(table1, table2, hyperrectangle, udf, udf value)`

Partitioning the Space

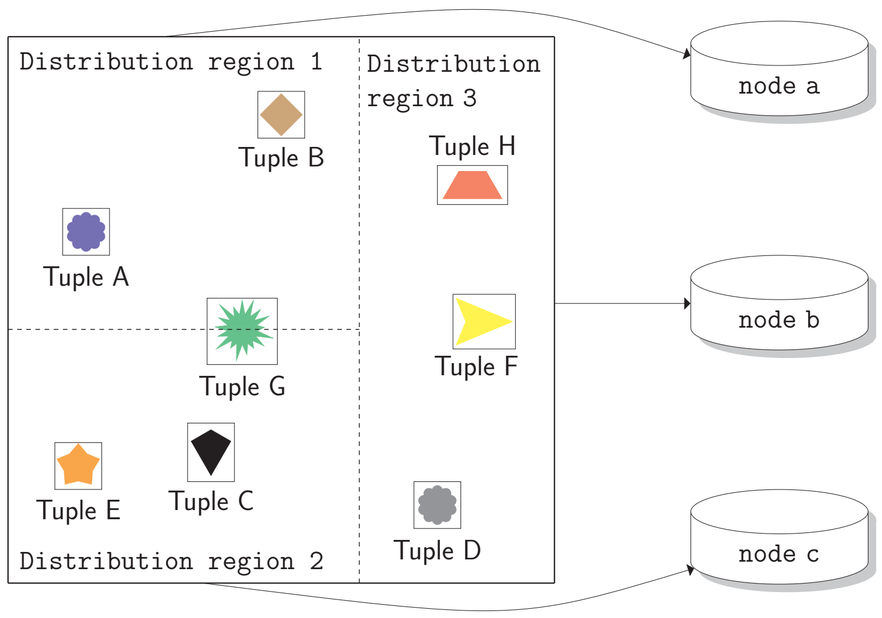


Figure: The space is partitioned into distribution regions. Each tuple is stored together with its bounding box. Tuples that belong to multiple regions are duplicated.

User Defined Filters (UDFs)

- Enhance the query processor so that the stored values can be decoded (e.g., GeoJSON encoded values).
- **Turn the generic data store into a specialized system for a specific data type (e.g., spatial joins on the real geometries of stored values become possible).**

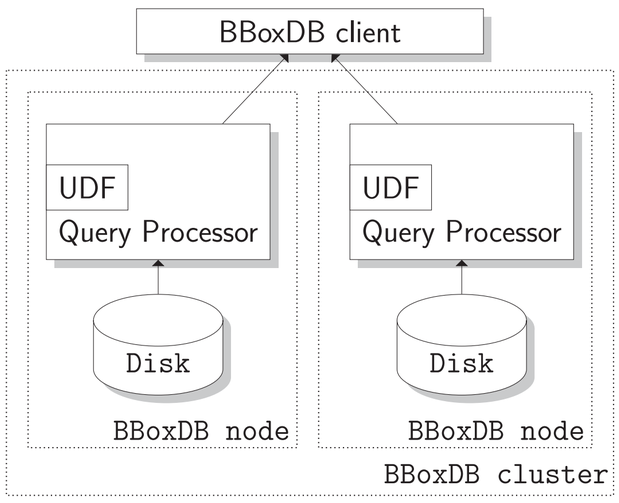


Figure: Using a UDF in a BBoxDB cluster with two nodes. The UDF is loaded into the query processor on each node. Only the tuples that pass the UDF are sent to the client.

Technical Details of the UDFs

- The UDF acts as a filter and refines the output of the query processor.
- The query processor loads the UDF dynamically at runtime.
- The creation of a new UDF is simple: only the two methods of the interface `UserDefinedFilter` need to be implemented.
- The method `filterTuple` refines range queries, the method `filterJoinCandidate` refines join queries.
- Existing Java libraries can be used (e.g., the *Esri Geometry API for Java*).
- The UDF is compiled into *Java bytecode*, placed into a special directory and distributed automatically to all nodes of the cluster.

A UDF for GeoJSON Data

```
public class UserDefinedGeoJSONFilter implements
    UserDefinedFilter {

    public boolean filterTuple(Tuple tuple, byte[] udfValue) {
      OGCGeometry geo1 = toGeometry(udfValue);
      OGCGeometry geo2 = extractGeometry(tuple);
      return geo1.intersects(geo2);
    }

    public boolean filterJoinCandidate(Tuple tuple1,
       Tuple tuple2, byte[] udfValue) {
      OGCGeometry geo1 = extractGeometry(tuple1);
      OGCGeometry geo2 = extractGeometry(tuple2);
      return geo1.intersects(geo2);
    }
}
```

The Graphical User Interface

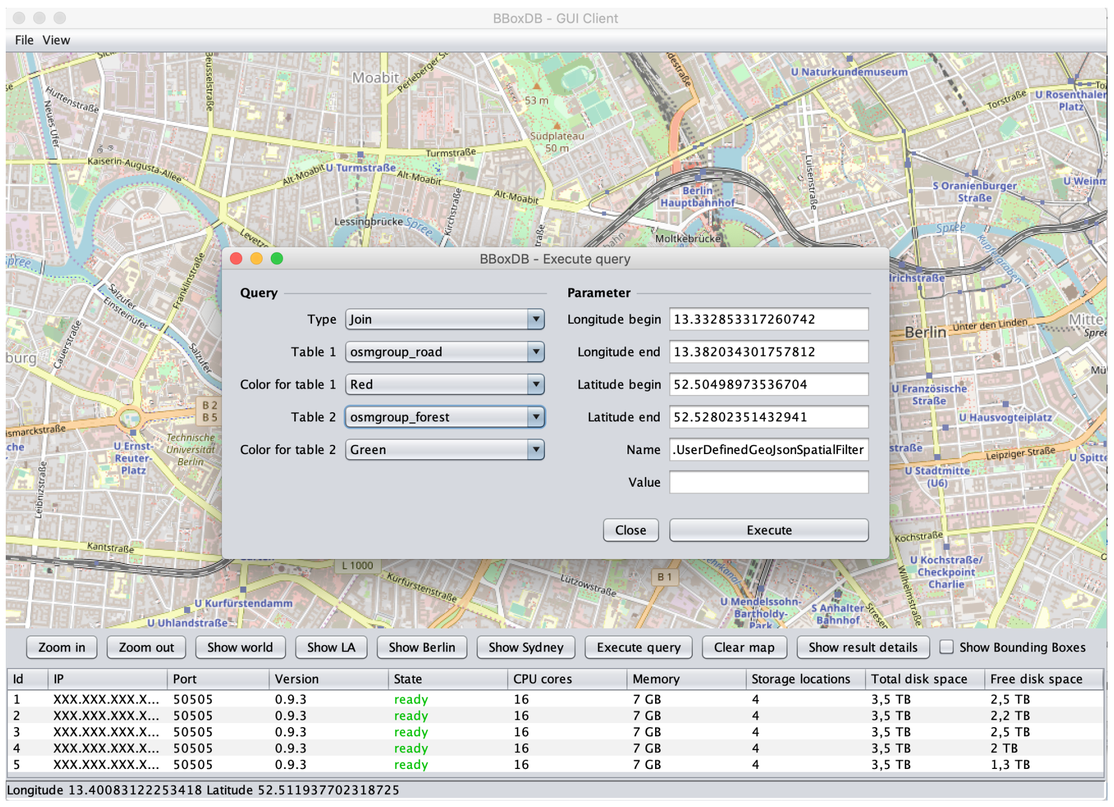


Figure: The GUI of BBoxDB allows the execution of queries. The query range can be interactively selected using the mouse.

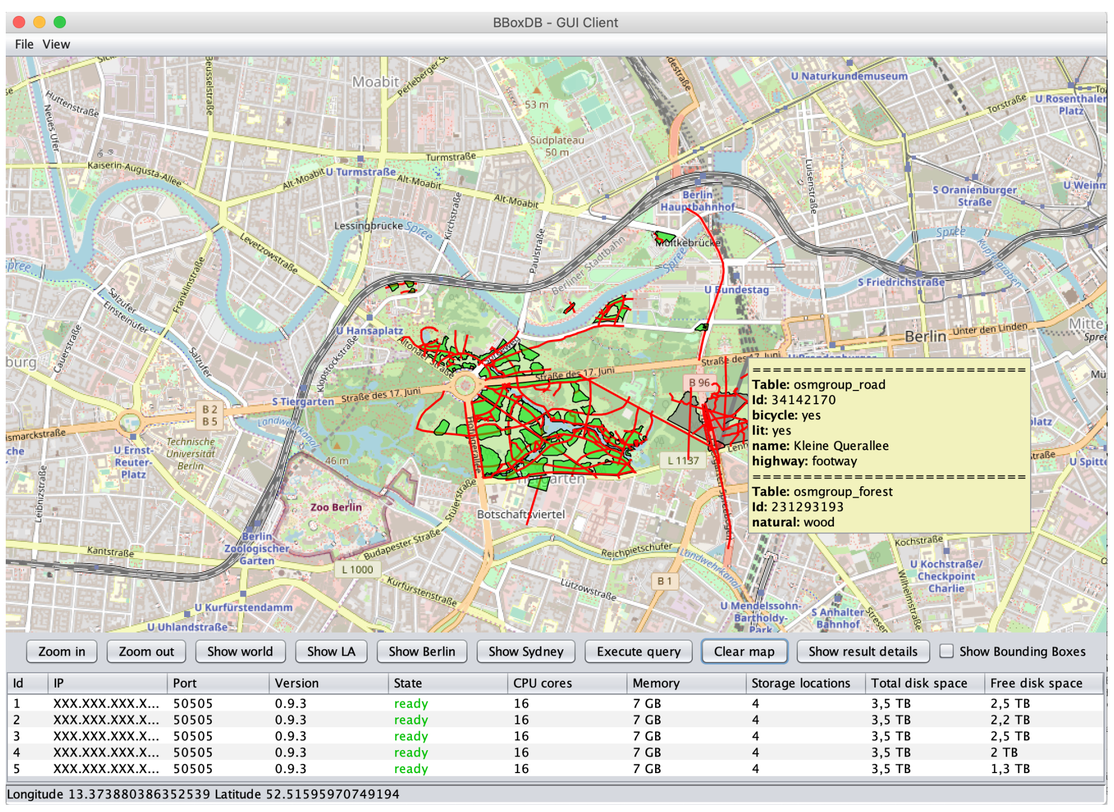


Figure: The result of a spatial join between roads and forests. The data can be explored interactively and information about the elements are shown in a tool-tip.