

Ride the Lightning

RMI – Safir application

Royal Meteorological Institute – Belgium
Geo Solutions – Belgium
Wouter Schaubroeck

Thx to Metallica for this title...

Summary

- Introduction
- Requirements
- Software stack
- Application Structure
- Usage of Geotools, and customisations
- Demo
- Questions??

Introduction...

- All Belgium (30,528 km²) is covered by a 30 by 30 km² grid of lightning sensors;
- These sensors provide the location (accuracy: 2 km) and the type of lightning



Display in real time the location and the type of lightning in an user friendly web environment

Requirements

- Open Source
- Open Standards
- Java
- User friendly (cfr Google)
- Upload own data (other than the lightnings)
- High Performance visualisation of all the lightnings in a period of time

Software Stack

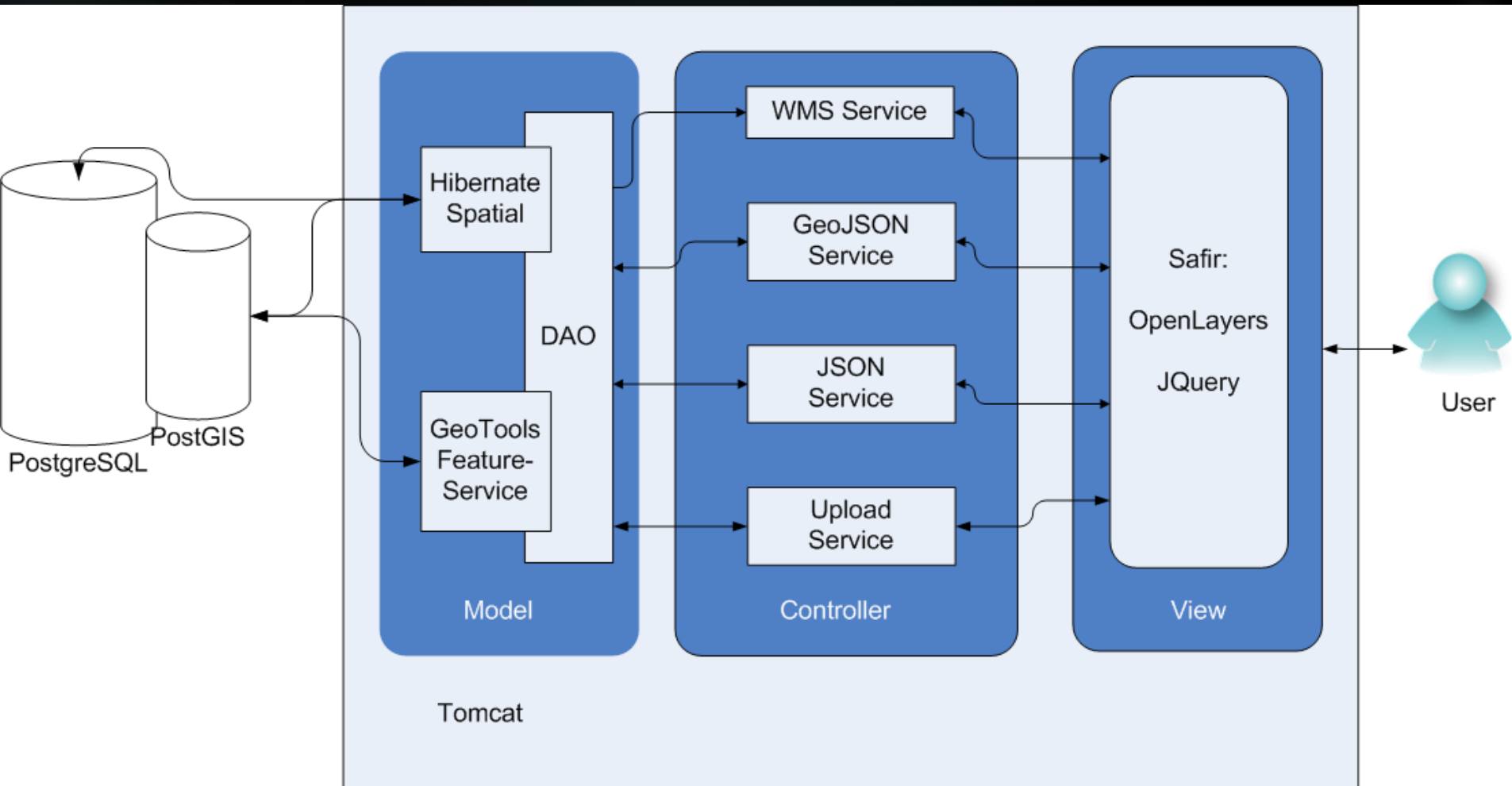
Server-side:

- PostgreSQL – PostGIS
- GeoTools
- Hibernate Spatial

Client-side

- JQuery
- OpenLayers

Application Structure



Application Structure

The application is glued together at the client-side:

- JavaScript is used to communicate with the different services, and to display the data.
- All the services are completely stateless.
- There are no dependencies between them.

Application Structure

Several advantages:

- every service can run on a dedicated machine.
- Easy scalable
- Easy maintainable

Application Structure

All the services are constructed as a chain of FeatureReaders:

- Each FeatureReader has a specific task
- Easily pluggable: always returning an Object-type
- Compatible with Hibernate Spatial and GeoTools

Application Structure

Examples of requests:

- Wms – getmap request (back-end GeoTools)
- GeoJSON: locations of lightnings (back-end: Hibernate Spatial)
- JSON: legend of the density map, usermanagement, ... (back-end: Hibernate Spatial)

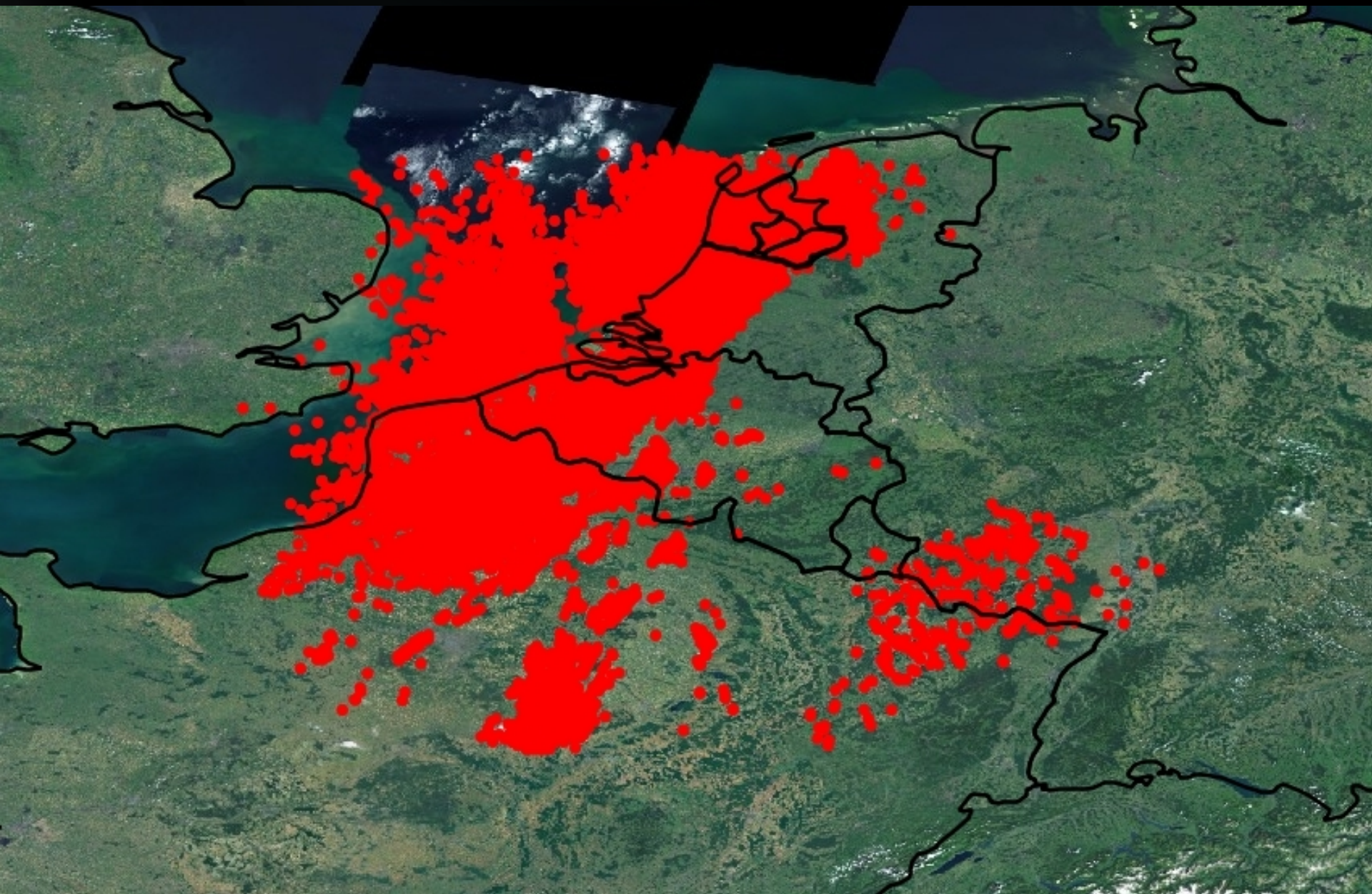
Usage of GeoTools

- WMS Rendering: only points
- On-The-Fly density grid: how to calculate this as fast as possible?

Usage of GeoTools

WMS Rendering:

Case: display 10000 points inside a small bounding box, on a large scale.



RMI - Safir application

Usage of GeoTools

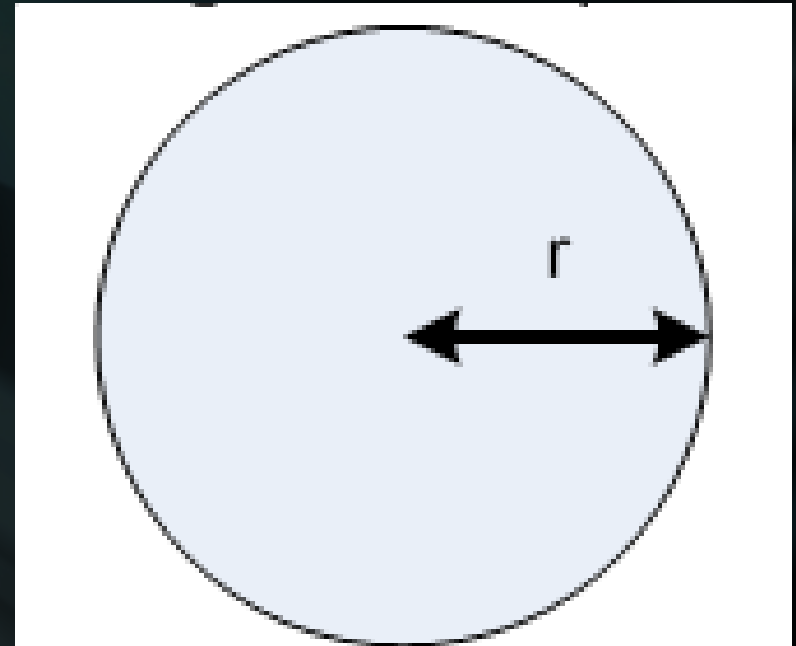
Solution:

Only draw what is visible...

-> How to know what is visible?

Visible lightnings

Lightnings are displayed on the map as circles, with a certain radius (specified in the SLD)



>> These circles have a certain spatial extent

Visible Lightnings

Points (or lightnings) inside this spatial extent are not or partially visible.

>> Solution:

Draw only points that aren't covert by this spatial extent

Visible Lightnings

Type of spatial extent:

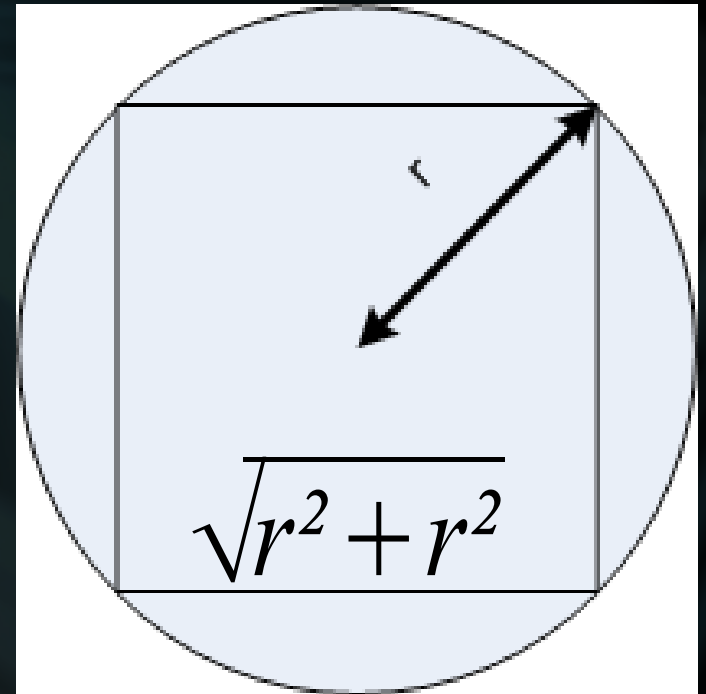
1.Circle: not favorable (what with points near the border?)

2.Inner Square: better, a little margin for points near the border, easy to work with inside the bounding box

Visible Lightnings

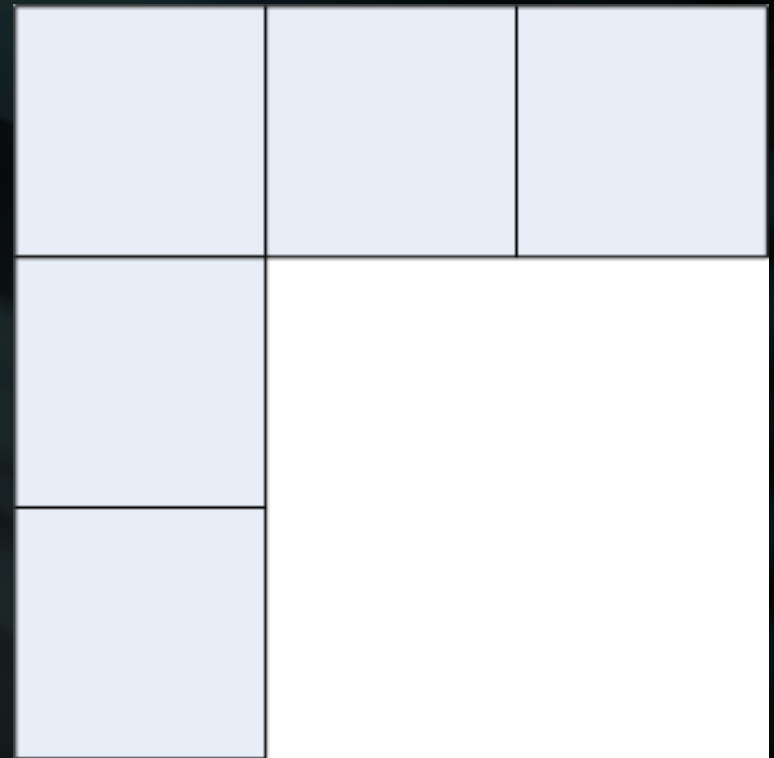
Create a grid with these inner squares that fits the bounding box

The length of one side:
(Pythagoras)



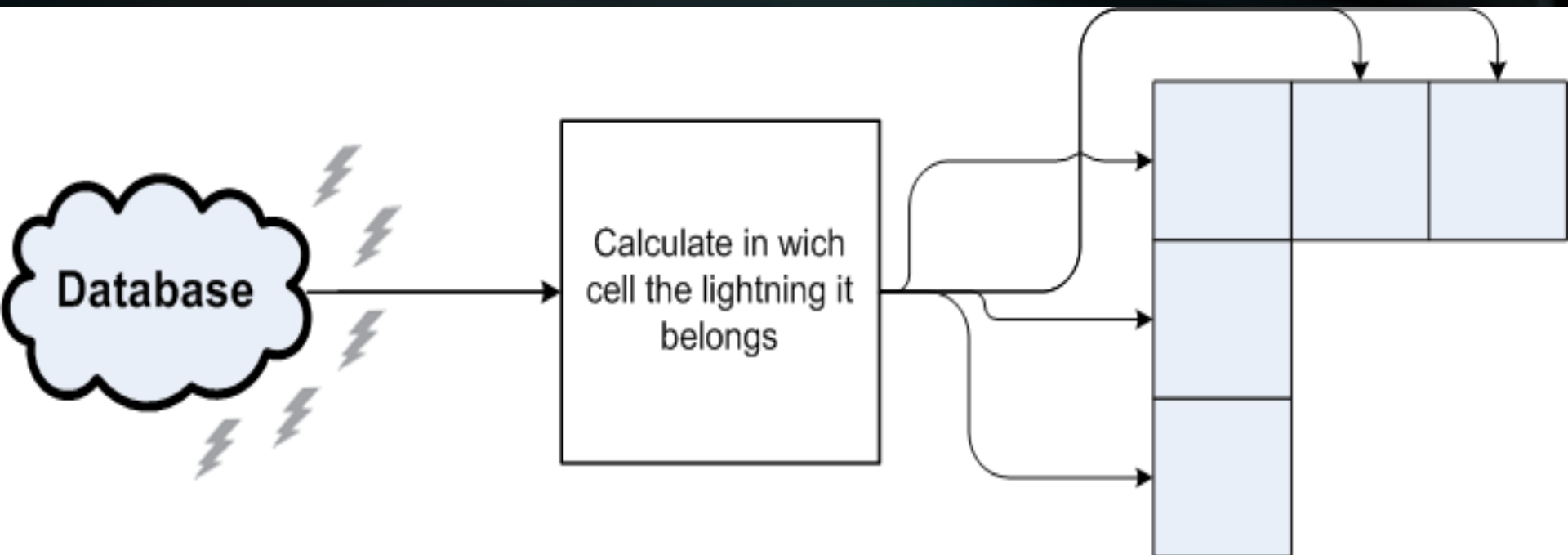
Visible Lightnings

Resulting grid:



Visible Lightnings

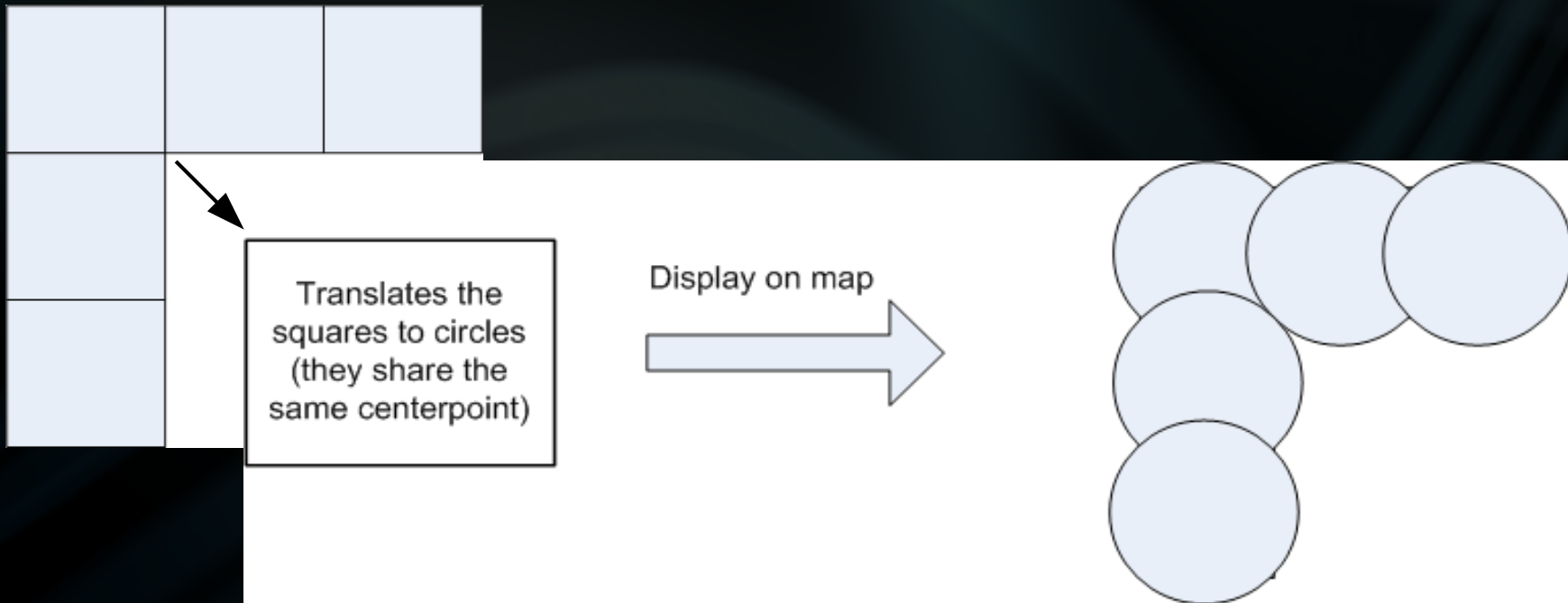
Loop over all the lightnings in the database, and calculate in which cell the lightning belongs



Result: Grid with empty cells where there are no lightnings

Visible Lightnings

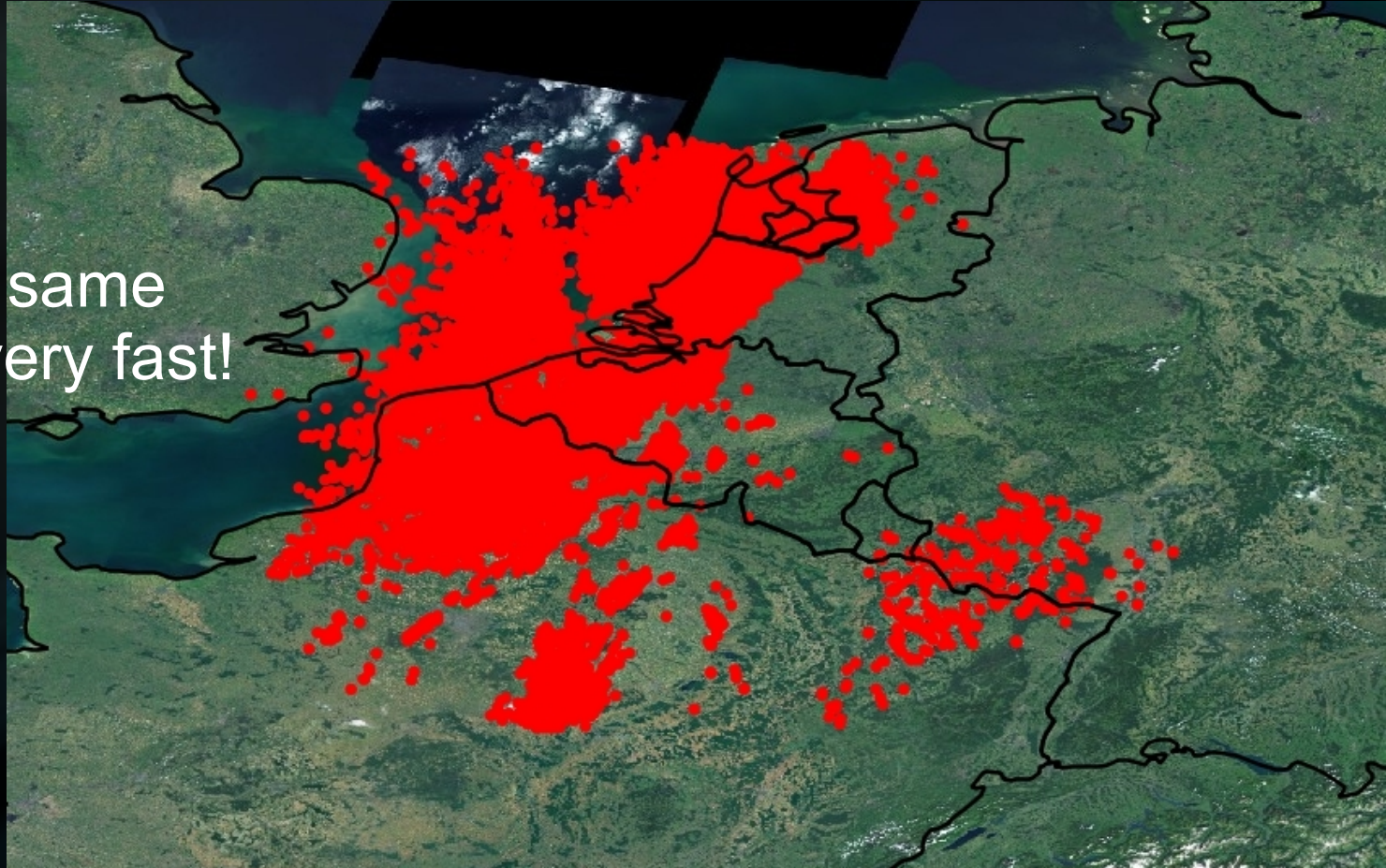
Convert the grid (only the filled cells)



Visible Lightnings

Result:

Looks the same
Renders very fast!

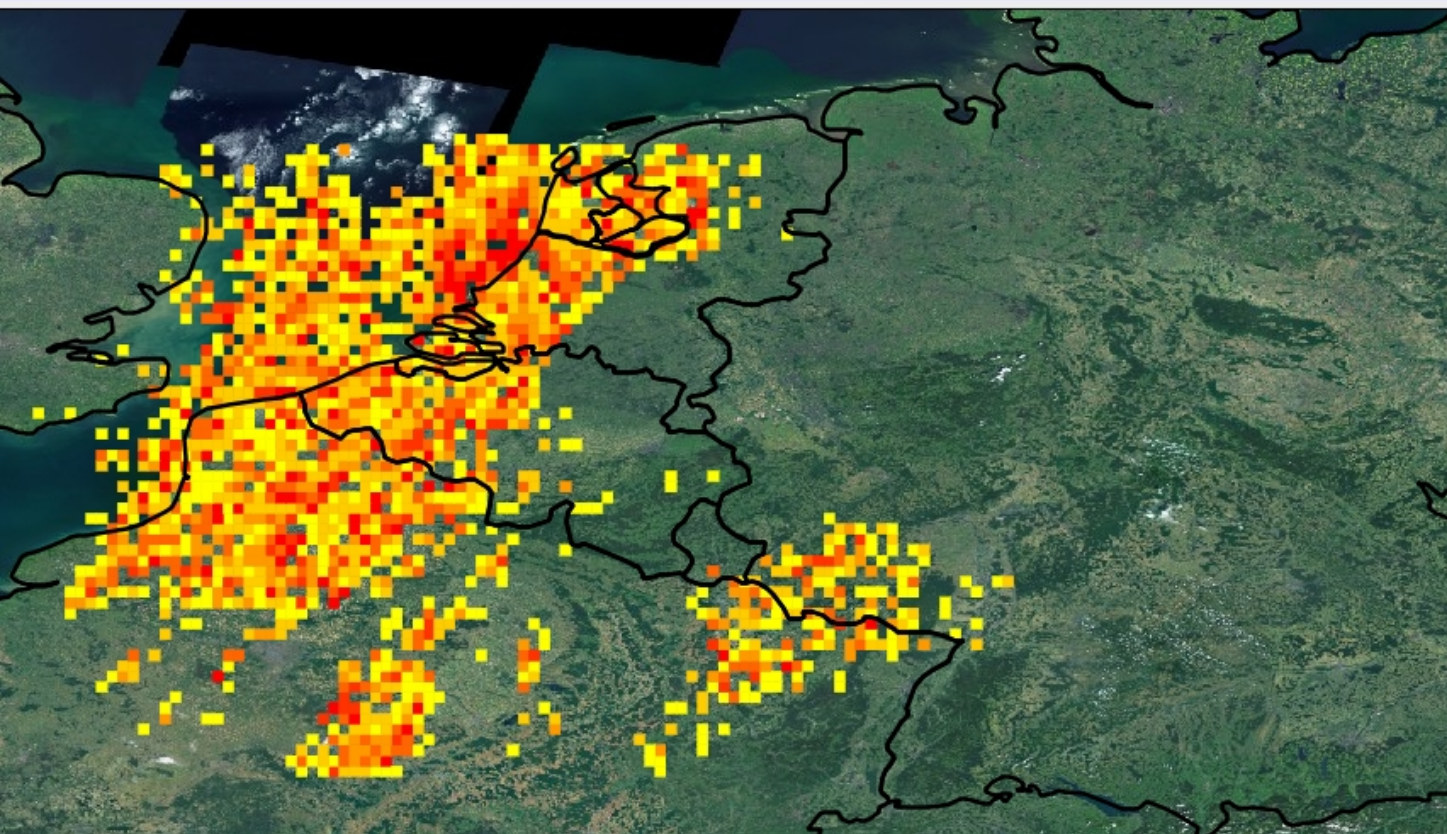


RMI - Safir application

On the fly density grid

- Same algorithm as for the visible lightnings
Except: it counts how many lightnings per cell
- User-defined grid-size
- Displayed as squares, in 6 different categories

On the fly density grid



map

legend

info

▼ legend



Laatste geregistreerde inslag: Tijdstip:
15/08/2007 - 11:08:59 - Opgewraagd:
9:52AM

< Click on the map for information
KMI © 2007 | [Disclaimer](#) | [Contact](#)

Usage of GeoTools

Some remarks:

- This method works only for points
- Not so accurate, accuracy depends on size points on map

Usage of GeoTools

This principle has already been implemented in the code of uDig (which is based on GeoTools), and will be implemented in the code of GeoTools.

Usage of GeoTools

The JSON and GeoJSON Service both rely on Hibernate Spatial to retrieve their data.

Hibernate Spatial is the perfect combination to retrieve objects, combined with a (complex) spatial query on the database.

Demo & Questions?