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
- Visualisation and usage of GeoTools
- Demo
- Questions??

Introduction…

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

Real-time display the location and the type of lightning in an user friendly web environment

Requirements

- Open Source
- Open Standards
- WMS – Service
- Java
- User friendly (cfr Google)
- High Performance visualisation of all the lightnings in a period of time

Software Stack

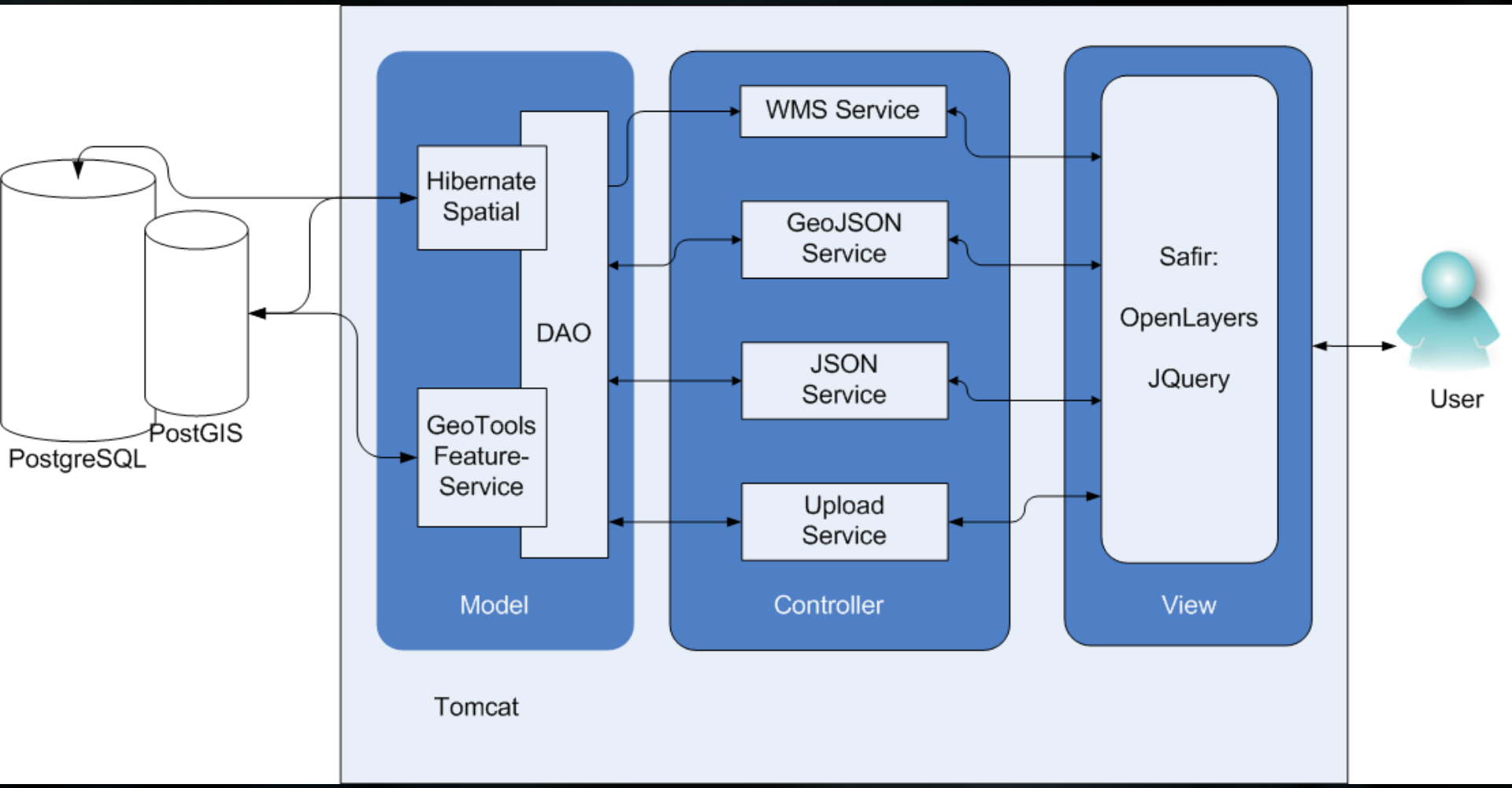
Server-side:

- PostgreSQL – PostGIS
- GeoTools
- Hibernate Spatial

Client-side

- Jquery (JavaScript library)
- OpenLayers

Application Structure



RMI - Safir application

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

Visualisation

WMS (map) Rendering:

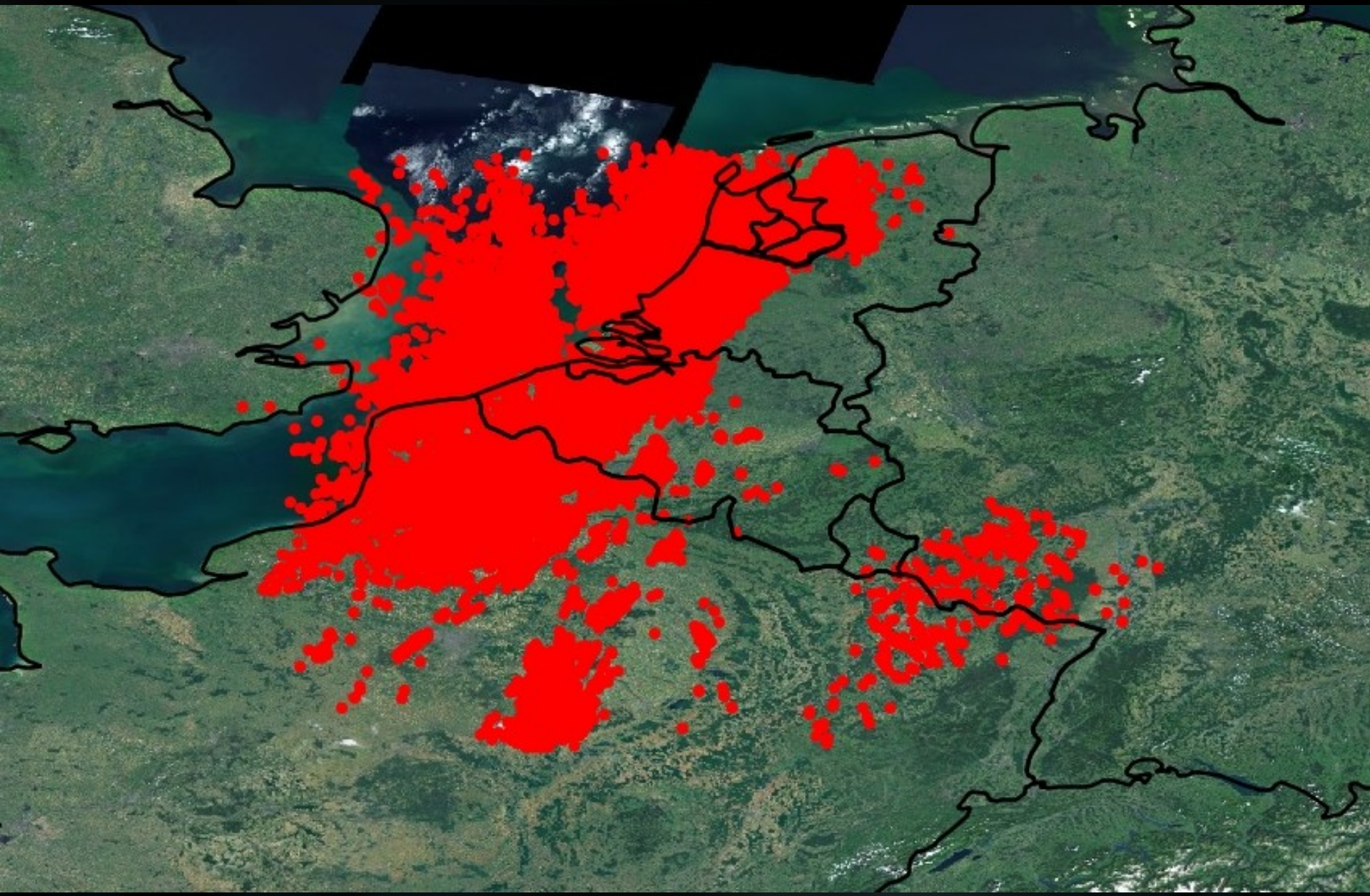
- point locations
- density grid (calculated in real time)

→ how to calculate this as fast as possible?

Visualisation

WMS Rendering point locations:

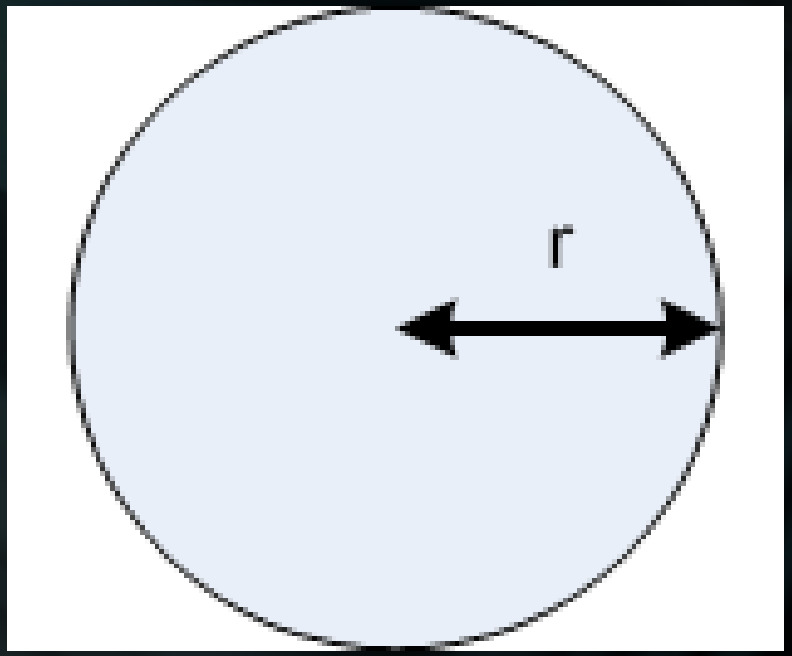
Case: display 100000 points inside a small bounding box, on a large scale.



RMI - Safir application

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.

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

Visible Lightnings

Type of spatial extent:

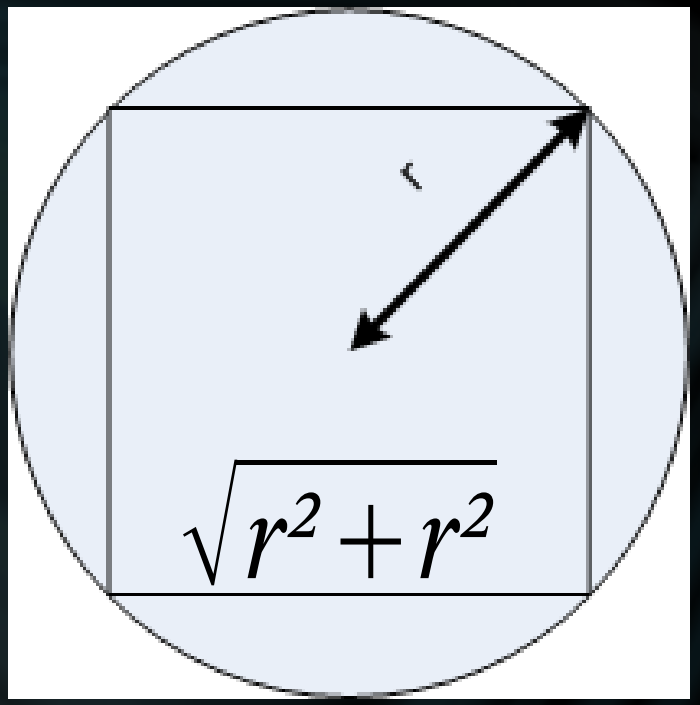
1. Circle: not favorable (what with points near the border?)

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

Visible Lightnings

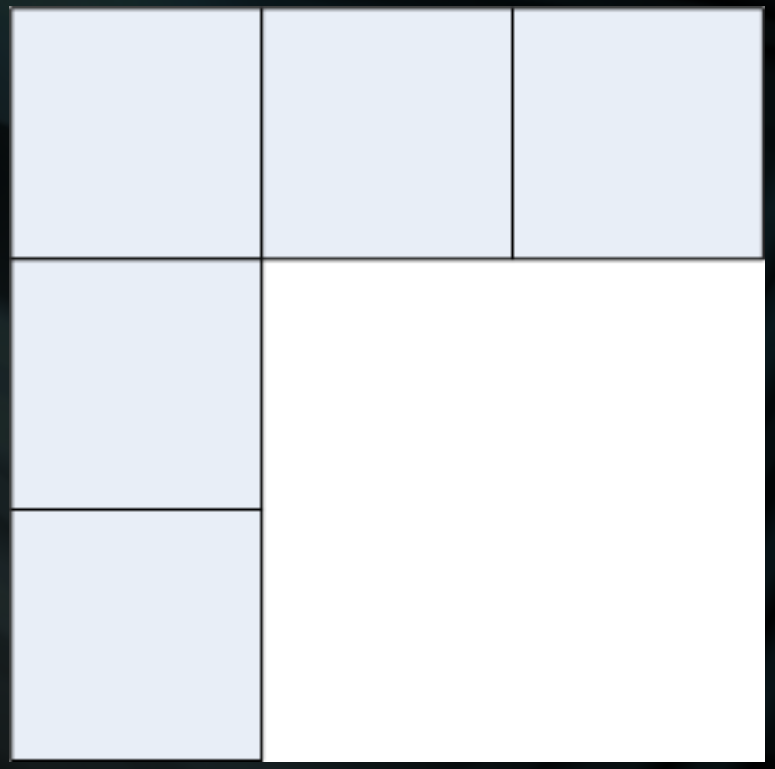
Create a grid with these inner squares that fits the bounding box (of the image)

The length of one side: (Pythagoras)



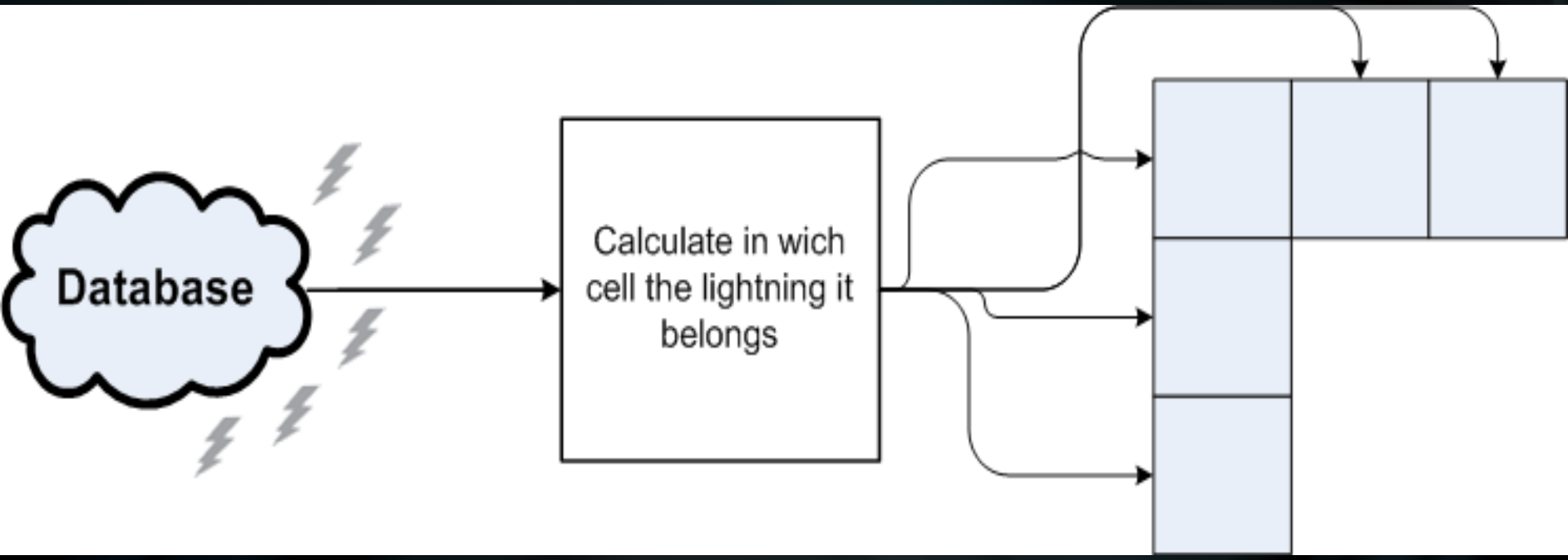
Visible Lightnings

Resulting grid:



Visible Lightnings

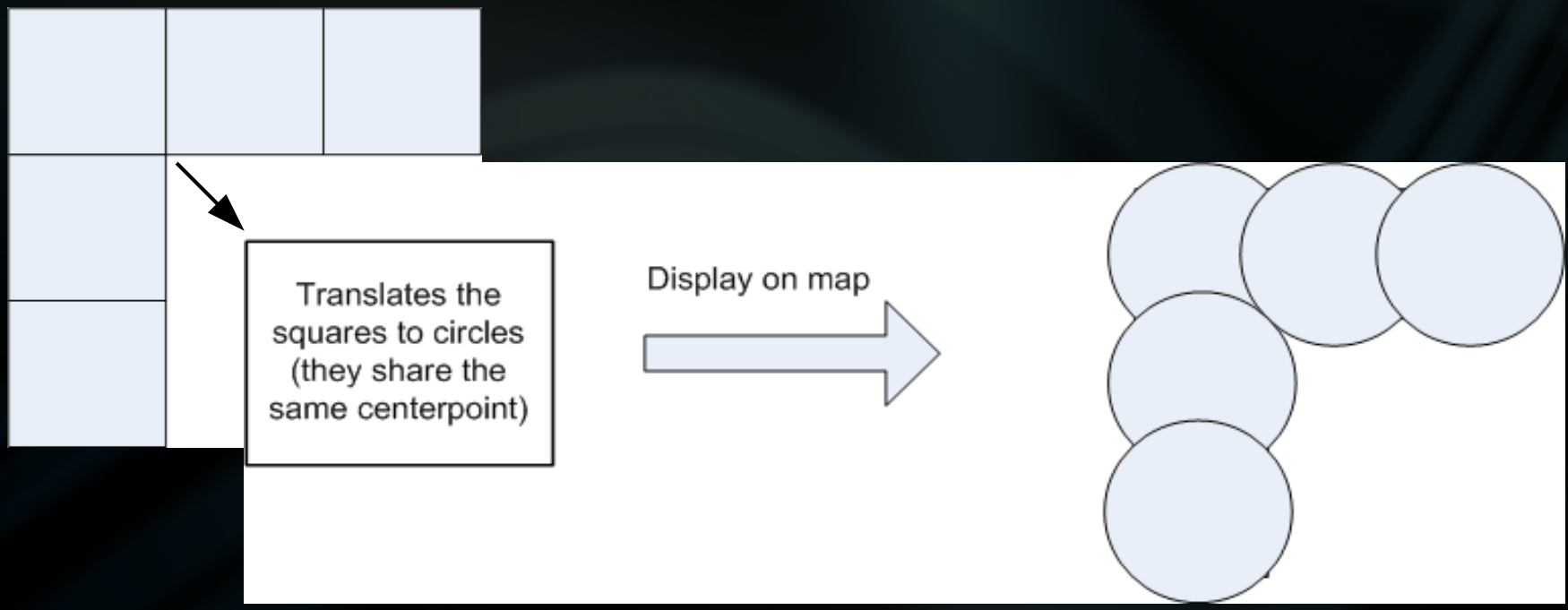
Loop over all the lightnings in the database, and calculate in which cell the lightning belongs



Result: Grid with empty cells where there are no lightnings

Visible Lightnings

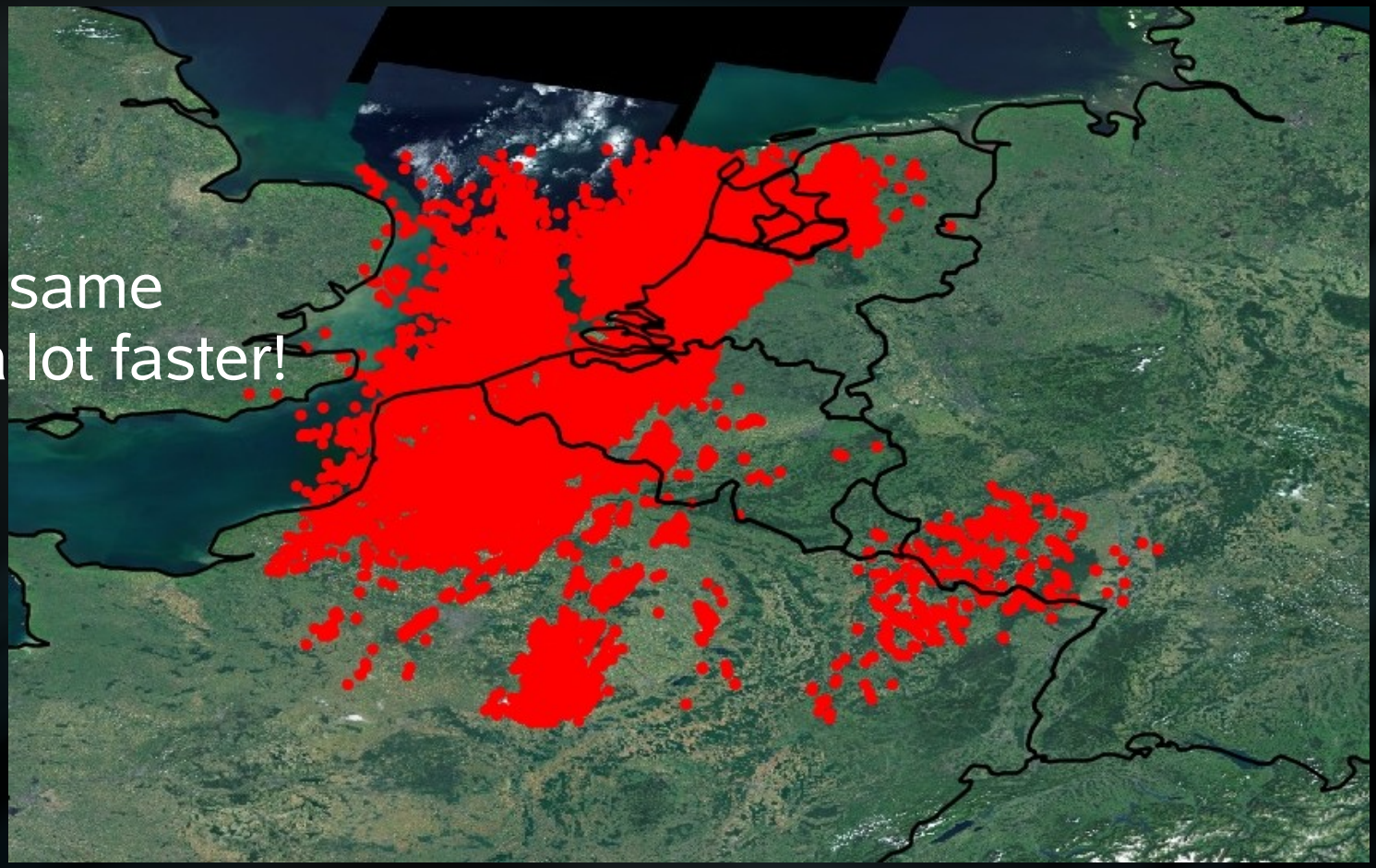
Convert the grid (only the filled cells)



Visible Lightnings

Result:

Looks the same
Renders a lot faster!

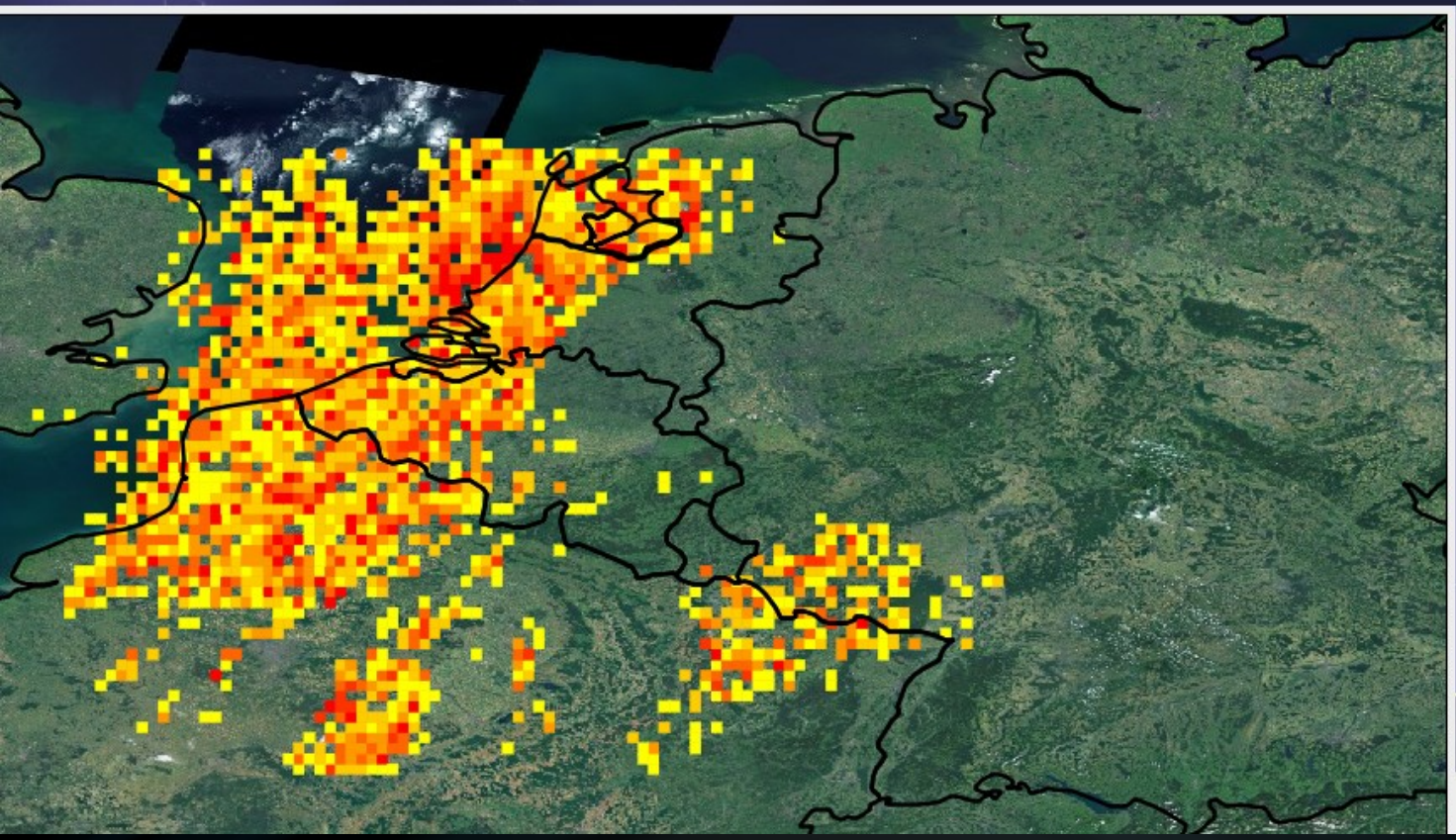


RMI - Safir application

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



RMI - Safir application

Visualisation

Some remarks:

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

Implementation in GeoTools

This algorithm will be implemented in GeoTools (after some more optimalisations). Something similar has already been implemented in uDig (wich is based on GeoTools)

Demo & Questions?